MARCOS ROBERTO SILVA

UMA CONTRIBUIÇÃO AO PROJETO DE REDES DE TRANSPORTE DE CARGA PARCELADA

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

São Paulo

2010

MARCOS ROBERTO SILVA

UMA CONTRIBUIÇÃO AO PROJETO DE REDES DE TRANSPORTE DE CARGA PARCELADA

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

Área de Concentração:
Engenharia de Transportes

Orientador: Prof. Livre-Docente
Cláudio Barbieri da Cunha

São Paulo

2010

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 12 de novembro de 2010.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Silva, Marcos Roberto
Uma contribuição ao projeto de redes de transporte de carga parcelada / M.R. Silva. -- ed.rev. -- São Paulo, 2010.
219 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Transportes.

1. Logística 2. Transportes (Modelagem matemática) I. Uni - versidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Transportes II. t.

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. Cláudio Barbieri da Cunha pelas diretrizes seguras e permanente incentivo.

À minha família pelo estímulo e incansável apoio.

A todos que, direta ou indiretamente, colaboraram na execução desse trabalho.

RESUMO

Esta pesquisa trata do projeto de redes de distribuição de carga parcelada. Mais especificamente são tratados dois tipos de problemas que são comuns no planejamento desse tipo de sistema. O primeiro deles corresponde ao problema estratégico de configuração de redes do tipo *hub-and-spoke*, consistindo na definição simultânea da quantidade e localização de terminais para consolidação de carga (ou *hubs*), e na definição da alocação dos terminais aos *hubs* localizados. Uma vez determinada a configuração da rede, o segundo problema, no nível de decisão tático, corresponde na definição do caminho que cada carga parcelada deve percorrer desde sua origem até alcançar seu terminal de destino, a um mínimo custo, tendo a rede *hub-and-spoke* como um dado de entrada do problema.

Um novo modelo matemático é proposto para representar o problema estratégico de configuração de uma rede *hub-and-spoke,* possuindo uma menor quantidade de variáveis e restrições, ao se comparar com outros modelos matemáticos comumente utilizados para representar o problema. Esse novo modelo matemático permitiu a obtenção de soluções ótimas para problemas em redes com até 100 terminais, sendo apresentada pela primeira vez a solução ótima para problemas utilizados como *benchmark* na literatura. Dado que problemas de grande porte ainda continuam muito difíceis de serem resolvidos, são propostas três variantes de uma heurística simples e eficiente utilizando técnicas de multi-início e busca tabu, bem como uma heurística integrada em dois estágios baseada em busca tabu para solução. Experimentos computacionais utilizando dados tradicionalmente utilizados na literatura para solução de problemas de configuração de redes *hub-and-spoke* (conjuntos de dados CAB e AP), bem como instâncias novas e modificadas, mostraram que a abordagem utilizada para solução do problema possibilitou a obtenção da solução ótima, ou a melhor solução conhecida, para esses problemas em um tempo de processamento muito curto, permitindo assim resolver de forma eficiente problemas de grande porte, nunca antes resolvidos em pesquisas anteriores.

O segundo problema foi motivado por uma aplicação prática de uma empresa de transporte rodoviário de cargas parceladas no Brasil. O problema diz respeito ao planejamento de carregamentos a serem realizados em cada terminal, levando-se em consideração cada carga parcelada que precisa ser transportada, definindo o percurso que cada carga deve percorrer até chegar ao seu destino. É proposto um modelo matemático e, dada a dificuldade para se resolver problemas de tamanho como o encontrado na prática, é proposto também um método de solução utilizando metaheurística busca tabu. Experimentos computacionais realizados mostraram que a heurística proposta pôde efetivamente resolver problemas de tamanho como o encontrado na prática.

Palavras-chave: Projeto de redes. Transporte de cargas. Logística. Algoritmos. Heurísticas.

ABSTRACT

This research deals with problems related to distribution networks for less-than-truckload (LTL) freight transportation. More specifically, we deal with two relevant problems that arise. The first corresponds to the strategic problem of designing and configuring hub-and-spoke networks in terms of simultaneously determining the optimal number of consolidation terminals (hub) nodes, their locations and the allocation of the other terminals (spokes) to the hubs. . Once the network configuration is determined, the second problem, in the tactical level of decision, corresponds to defining the path that each LTL individual freight needs to follow from its origin to reach its destination terminal, at a minimum cost, having a hub-and-spoke network topology as a data entry to the problem.

A new mathematical model is proposed to represent the strategic problem of designing a hub-and-spoke network, with fewer variables and constraints than previous formulations found in the literature This model allowed us to obtain optimal solutions for problems in transportation networks with up to 100 terminals, reporting for the first time the optimal solutions of benchmark problems in the literature. Since this problems still remains too hard to solve for larger instances, we propose we propose three variants of a simple and efficient multi-start tabu search heuristic as well as a two-stage integrated tabu search heuristic to solve it. Computational experiments using typical benchmark problems (CAB and AP data sets) as well as new and modified instances show that our approaches consistently return the optimal or best-known results in very short CPU times, thus allowing the possibility of efficiently solving larger instances of the USAHLP than those found in the literature.

The second problem is motivated by a practical application of a LTL transportation company in Brazil. It deals with the planning of loads to be done at each terminal, taking into account each LTL freight that needs to be transported, defining the path that each good needs to follow to reach its destination. A new mathematical model is proposed, and, since real world problems are very hard to solve, a heuristic based on tabu search is also developed. Computational experiments show that our heuristic can effectively solve real-world instances from a trucking company in Brazil

Keywords: Network design. Less-than-truckload freight transportation. Logistics. Algorithms. Heuristics.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 OBJETIVOS DO TRABALHO

Este trabalho foi estruturado e redigido de forma a constituir uma Tese de Doutorado, e com isso, preencher parte dos requisitos para obtenção do título de Doutor em Engenharia na área de Planejamento e Operação de Transportes do Departamento de Engenharia de Transportes da Escola Politécnica da Universidade de São Paulo.

Esta pesquisa trata do projeto de redes de distribuição de carga parcelada. Mais especificamente são tratados dois tipos de problemas que são comuns no planejamento desse tipo de sistema. O primeiro deles corresponde ao problema estratégico de configuração de redes do tipo *hub-and-spoke*, consistindo na definição simultânea da quantidade e localização de terminais para consolidação de carga (ou *hubs*), e na definição da alocação dos terminais aos *hubs* localizados. Uma vez determinada a configuração da rede, o segundo problema, no nível de decisão tático, corresponde na definição do caminho que cada carga parcelada deve percorrer desde sua origem até alcançar seu terminal de destino, a um mínimo custo, tendo a rede *hub-and-spoke* como um dado de entrada do problema.

Em nível de decisão estratégico, o objetivo é determinar o número de terminais que farão a consolidação de cargas parceladas, a localização de cada um deles, bem como sua respectiva abrangência ou área de atuação, de forma a minimizar a somatória dos custos envolvidos, que englobam custos variáveis de transporte e custos fixos de operação desses terminais.

Esta tese trata mais especificamente do problema de localização de *hubs* não capacitados com alocação única, e neste trabalho são apresentadas quatro heurísticas eficientes para sua solução, superando todos os métodos já desenvolvidos e apresentados na literatura. Foi ainda proposto um novo modelo

matemático para representar o problema, permitindo a obtenção da solução ótima para instâncias com redes de até 100 terminais.

Já em nível de decisão tático, o estudo está centrado na definição do fluxo (percurso ou caminho) que cada carga deve percorrer até chegar ao seu destino, tendo como dado de entrada uma rede de transporte do tipo *hub-and-spoke* com as possíveis ligações entre terminais já pré-definidas. Dado um conjunto de cargas a serem transportadas, deseja-se, portanto, decidir como essas cargas serão movimentadas nessa rede, definindo as possíveis paradas intermediárias, e também no dimensionamento dos veículos necessários em cada percurso.

Esse problema é de grande importância prática, pois o objetivo consiste em definir diferentes formas de consolidação e as rotas a serem operadas, tentando acompanhar a variação na demanda por transporte de carga parcelada que ocorre no Brasil em certas semanas do mês, e em certos meses do ano.

O problema de carregamento de cargas em uma rede de transporte, conhecido na literatura específica como *network loading problem* (NLP), é um problema NP-*hard,* não existindo métodos exatos eficientes para solução de problemas de tamanho igual ao encontrado na prática. Dada a complexidade para solução desse problema, é proposto nesta pesquisa um método de solução utilizando a metaheurística busca tabu, obtendo resultados satisfatórios na solução de um problema real de uma empresa transportadora de carga parcelada no Brasil.

Este trabalho trata, portanto, de dois problemas distintos, um de natureza estratégica, que consiste na difícil tarefa de localização, determinação da quantidade e área de atuação de terminais que operarão como *hubs* e serão responsáveis pela consolidação de cargas parceladas. Em nível de planejamento tático, outro problema estudado foi a determinação do fluxo ou rota que cada carga deve seguir para chegar ao seu destino a um menor custo possível, valendo-se dos benefícios da consolidação.

Parte desta pesquisa é uma continuidade da dissertação de mestrado do autor, finalizada em 2004 (SILVA, 2004), e também faz parte de um esforço de pesquisa

de modelagem de problemas de localização de instalações, e configuração de redes de transporte, que vem sendo desenvolvido no âmbito da Escola Politécnica, do qual se podem citar os trabalhos de Machado (1989), Tondo (1992), Gualda (1995), Hellmuth (2004), Mutarelli (2004), Vallin Filho (2004), Alamo (2005), Cunha (2006), Hamad (2006), Pires (2006) e Romero (2006).

1.2 RELEVÂNCIA DO TEMA

O transporte de cargas é um componente vital na economia. Ele fornece suporte à produção, ao comércio, e a atividades de consumo, assegurando a movimentação eficiente e disponibilizando, em tempo oportuno, matérias primas e produtos acabados. Além disso, o transporte pode representar uma parcela significativa do custo de inúmeros produtos, bem como da despesa nacional de qualquer país.

O setor de transporte de cargas vem alcançando, mundialmente, altos níveis de desempenho em termos de eficiência econômica e qualidade de serviço. Eficiência em decorrência da necessidade de empresas de transportes de buscarem lucro num ambiente de mercado aberto, competitivo e focado no custo. Qualidade devido aos serviços de transporte, que devem se adequar aos altos padrões impostos pelos atuais paradigmas de produção e gestão, tais como redução de estoques, menores *lead-times* de produção, de compras e de distribuição, serviços personalizados e controle de qualidade focada no cliente ao longo da cadeia logística como um todo. Para as empresas de transporte, esses padrões dizem respeito particularmente ao tempo total para entrega e à confiabilidade do serviço, os quais têm, muitas vezes, se traduzido dentro de objetivos como “esteja lá rápido, mas dentro dos limites especificados” ou “oferecer serviços de alta qualidade e desempenho consistente”.

O crescimento econômico experimentado pelo Brasil nos últimos anos teve também um grande impacto no setor de transporte rodoviário de cargas. Um exemplo disso foi o problema enfrentado por diversas empresas transportadoras, que tiveram enormes dificuldades operacionais no segundo semestre de 2009. Terminais superlotados, falta de caminhões, de motoristas e até de ajudantes acabaram causando expressivos atrasos nas entregas.

Benatti (2010), em seu comunicado como Presidente da Associação Nacional do Transporte de Cargas e Logística salientou a necessidade de se recompor os valores de frete, para permitir a cobertura dos custos, a reposição das margens de lucro e assegurar os investimentos necessários para que a demanda de carga possa ser atendida dentro da normalidade. Caso contrário, o transporte rodoviário de cargas pode se transformar em grave ponto de estrangulamento para o alto crescimento econômico do país, previsto para este e os próximos anos.

A demanda por transporte de cargas deriva do inter-relacionamento entre produtores e consumidores e as significativas distâncias que geralmente os separam. Produtores de bens precisam dos serviços de transporte para movimentar matérias primas e produtos intermediários, e para a distribuição dos produtos finais, com o objetivo de atender a demanda. Tais serviços são fornecidos por transportadores (ou transportadoras) tais como empresas de transporte rodoviário, aéreo, ferroviário, de cabotagem, de encomendas expressas e de serviços postais. Embarcadores, que podem ser produtores de bens ou empresas intermediárias (conhecidas como *brokers*), são geradores de demanda para os serviços prestados pelos transportadores.

O poder público em geral é o principal provedor e gestor de infraestrutura: rodovias, e muitas vezes uma parcela significativa da infraestrutura portuária de navegação interior e do transporte ferroviário. O poder público também atua na regulamentação como, por exemplo, o transporte de produtos químicos perigosos, e também cria impostos e tributos para as empresas e a sociedade como um todo.

Ao se examinar o transporte de cargas, pode-se muitas vezes distinguir entre produtores que executam seu próprio transporte com sua frota própria (que então se tornam transportadores da sua própria carga), e transportadoras contratadas, que realizam os serviços de transporte para vários embarcadores, ou seja, empresas que necessitam transportar suas matérias primas ou produtos.

Serviços sob medida para cada cliente tornam-se mais difíceis de serem oferecidos quando as demandas de alguns clientes são atendidas simultaneamente, utilizando

o mesmo veículo ou comboio. Transportadores devem estabelecer rotas regulares e ajustar suas características (paradas intermediárias, frequência, tipo de veículo, capacidade, velocidade etc.) para atender as expectativas de um maior número possível de clientes.

Externamente, os transportadores então definem uma série de rotas, ou serviços, cada uma com suas características operacionais. Solicitações de serviços são, muitas vezes, agrupadas em uma programação que indica os tempos de chegada e de partida e as paradas de cada percurso. Internamente, as transportadoras constroem uma série de regras e políticas que afetam o sistema como um todo e são muitas vezes agrupadas em um plano operacional, cujo objetivo é assegurar que os serviços propostos sejam realizados conforme acordado com o(s) cliente(s) (ou o mais próximo possível), operando de uma maneira racional e eficiente.

A presença de terminais onde as cargas e os veículos são consolidados, agrupados, ou simplesmente movidos de um serviço para outro, caracteriza esse tipo de transporte realizado por transportadores de carga parcelada.

Segundo o sumário executivo do Plano Nacional de Logística & Transportes, apresentado pelos Ministérios dos Transportes e da Defesa no ano de 2007, o transporte rodoviário de cargas é responsável por 58% da produção total de transporte, tendo evoluído significativamente nos últimos anos, em parte devido às exigências dos clientes, mas principalmente em decorrência do aumento da competição entre empresas, com margens cada vez menores.

O segmento de carga parcelada abrange, como o próprio nome já diz, os serviços de transporte rodoviário de cargas cujo volume, para um cliente, uma origem e um destino, não é suficiente para lotar um veículo. Assim, cargas de diversos clientes são agregadas e transportadas conjuntamente. Esse tipo de serviço normalmente engloba as operações de coleta no cliente remetente, transferência de longa distância entre terminais e entrega ao destinatário.

Consequentemente, as empresas de transporte rodoviário dispõem de estruturas de apoio, que são instalações geograficamente distribuídas, geralmente denominadas

“filiais”', nas quais são consolidadas cargas de diferentes origens para destinos convergentes. Na maioria das vezes as localizações destas filiais foram definidas ao longo do tempo, de acordo com a demanda de cargas, e também pela prática e experiência do líder responsável pela empresa.

1.3 ORGANIZAÇÃO DA TESE

Este texto está estruturado e distribuído em Capítulos, sendo que no Capítulo 2 é apresentada uma visão geral do transporte de carga parcelada, incluindo as diferentes formas e particularidades de operação.

O Capítulo 3 corresponde à revisão bibliográfica onde são apresentados inicialmente conceitos e definições básicas para o desenvolvimento dos demais capítulos, e relacionados aos modelos de configuração de redes *hub-and-spoke*, como também no projeto de rede de serviços.

O Capítulo 4 diz respeito ao projeto de uma rede de transporte do tipo *hub-and-spoke*, apresentando o modelo matemático utilizado para representar o problema, alternativas eficientes para solução, métodos esses baseados tanto em métodos exatos como também em heurísticas, e ainda experimentos computacionais realizados para verificar a eficácia das estratégias de solução propostas.

No Capítulo 5 é detalhado o problema do carregamento de cargas parceladas em uma rede de transporte, apresentando também a formulação matemática para representar o problema. Dada a complexidade do problema e dificuldade em se desenvolver um método exato para solução, foi proposta uma heurística baseada em busca tabu. Experimentos computacionais foram conduzidos para verificar a eficácia do método de solução proposto.

Finalmente no Capítulo 6 são apresentadas as conclusões obtidas e considerações finais, incluindo o encaminhamento sugerido para continuidade desta pesquisa.

2 UMA VISÃO GERAL DO TRANSPORTE DE CARGA PARCELADA

Neste Capítulo é apresentada uma visão geral do transporte de carga parcelada, com vistas a um melhor entendimento tanto da revisão bibliográfica que se segue, quanto dos dois problemas tratados nesta tese:

- o problema tratado em nível de decisão estratégico, consistindo na configuração de uma rede do tipo *hub-and-spoke*, isto é, na definição da quantidade e localização de terminais que operarão como pontos para consolidação de cargas parceladas (ou simplesmente terminais de consolidação), e também na definição da designação dos outros terminais da rede aos *hubs* localizados;
- em nível de decisão tático, o problema da definição do carregamento de cargas parceladas em uma rede de transporte pré-definida, especificamente na determinação do percurso ou caminho que cada carga deve percorrer, incluindo possíveis paradas em terminais intermediários, para chegar ao terminal de destino que será responsável pela entrega final dos produtos.

Empresas transportadoras de carga pelo modal rodoviário geralmente se especializam em um dos seguintes tipos de carga:
1) carga completa (*truckload*)
2) carga parcelada (*less-than-truckload*)

A Comissão de Comércio Interestadual dos Estados Unidos (*Interstate Commerce Comission - ICC*) define uma carga parcelada como sendo uma carga com até 10.000 libras de peso, ou 4.536 kg, enquanto que uma carga completa é aquela com peso acima desse valor. O ICC não categoriza as empresas como sendo de carga completa ou parcelada; somente as cargas são classificadas. A mesma definição é encontrada em Kawamura (1999) que distingue a carga fracionada da lotação da mesma forma que o ICC, utilizando os mesmos valores limites de peso como parâmetro para diferenciação.

Uma carga completa é geralmente um carregamento individual desde a sua origem até o destino em um único veículo. Esta carga não necessita nenhuma

movimentação ou classificação intermediária. Um transportador especializado em carga parcelada pode também oferecer serviços de transporte de carga completa. Para fazer um uso econômico do veículo, as cargas parceladas são geralmente consolidadas.

É comum encontrar empresas transportadoras que se especializam em carga parcelada. Uma empresa transportadora especializada no transporte de encomendas e pequenos volumes muitas vezes restringe o peso a ser transportado, sendo que muitas delas limitam esse peso para até 20 kg. Como exemplo o serviço postal norte-americano (USPS), Fedex, UPS e empresas regionais dos EUA como a AB Express.

2.1 OPERANDO COMO UM TRANSPORTADOR DE CARGA COMPLETA

Um transportador rodoviário de carga parcelada realiza o transporte de um despacho desde o seu ponto de origem até o seu ponto de destino. Se não houvesse nenhuma restrição de nível de serviço, a política ideal para despacho de um veículo seria "viaje quando estiver lotado". Sob essa política, a utilização do veículo seria máxima e cada despacho seria enviado diretamente da sua origem até o seu destino.

Utilizando-se esta política, se não houver carga suficiente para completar a capacidade do veículo em um determinado percurso, o veículo permanece na origem aguardando carga suficiente para sua liberação para viagem. Contudo, esta seria uma prática inviável uma vez que não existiria embarcador que aceitaria que seu despacho permanecesse na origem aguardando fluxo suficiente, sem uma data de embarque, e conseqüentemente de entrega, bem definidas.

Dado que o nível de serviço oferecido é de extrema importância para a manutenção e conquista de novos clientes, a implementação de uma estratégia do tipo "viaje quando estiver lotado" parece ser uma péssima idéia, uma vez que não seria

possível garantir o cumprimento do prazo de entrega acordado, o que levaria as empresas a aumentarem seus estoques em trânsito.

Se não há carga suficiente para completar a capacidade de um veículo, mesmo que parcialmente, esse veículo deve ser liberado para viagem a fim de não comprometer o nível de serviço prestado aos clientes. Entretanto, se não há carga suficiente para completar a capacidade de um veículo, a liberação de um veículo parcialmente ocupado é uma forma cara e ineficiente de se operar.

Um despacho corresponde a um embarque único contendo uma origem, um destino e dimensões (peso e volume) definidos, um único cliente remetente e um único cliente destinatário. O conjunto de despachos de diversos clientes, partindo de um mesmo terminal de origem para o mesmo terminal de destino, constitui uma carga, com peso e volume totais correspondentes às somatórias dos pesos e volumes de todos os despachos que a compõe.

Tipicamente, a grande maioria dos despachos possui um tamanho muito pequeno. No caso da Patrus Transportes, que é uma empresa brasileira de transporte rodoviário de cargas parceladas, conforme apresentado na Figura 2.1, mais de 90% dos despachos transportados em março de 2010 ocuparam individualmente menos de 1% da capacidade de uma carreta (sendo uma carreta um veículo tipo composto por um cavalo trator e um semi-reboque, comumente utilizado para transferência de carga entre terminais, possuindo 90 m^3 e 25000 kg de capacidade em volume e peso, respectivamente).

Baseado nesta informação do peso dos despachos, uma das questões que ocorre imediatamente é: Se uma rede de transporte de carga parcelada utilizar uma estratégia de operação similar ao transporte de carga completa, quais seriam os impactos negativos? Se a consolidação não fosse permitida, da mesma forma que no transporte de carga completa, um veículo seria liberado em cada um dos terminais de origem para o terminal de destino somente com a carga disponível e pertencente a esse percurso. O único custo a ser considerado seria o custo de transporte para cada uma dessas rotas diretas.

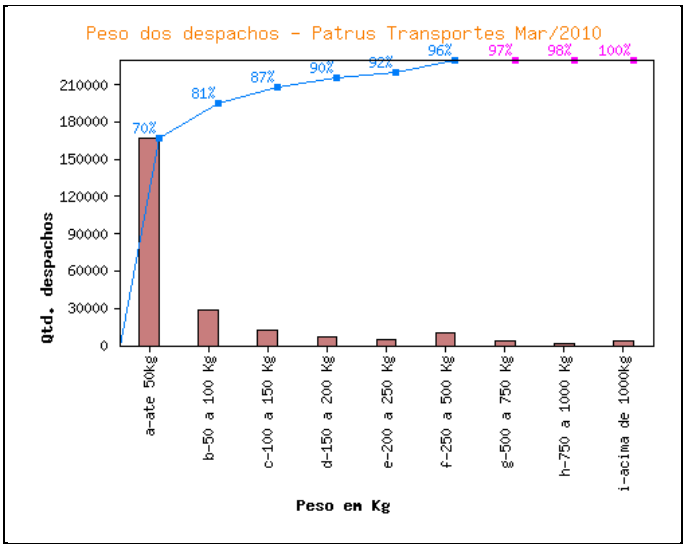

Figura 2.1: Distribuição do peso dos despachos transportados

Fonte: Patrus Transportes - Março de 2010

Baseado nos dados utilizados para o estudo de caso estimou-se que, se fosse utilizada a mesma estratégia de operação de uma empresa transportadora de carga completa (sem consolidação), os custos de transporte seriam em torno de 64 vezes maiores ao se comparar com a operação de consolidação permitida em uma rede contendo terminais operando como *hubs* sendo, portanto, suficiente para justificar a utilização de uma estratégia de consolidação.

Terminais de consolidação (ou *hubs*) são terminais que servem como ponto de transbordo e classificação em sistemas de distribuição entre pontos de oferta e/ou demanda de tráfego. Em vez de servir cada par origem-destino diretamente, *hubs* consolidam cargas parceladas para se valerem da economia de escala. Cargas de diversas origens, tendo em cada uma delas fluxos para destinos diversos, são consolidadas em um *hub*, e então combinadas com outras cargas que possuem destinos convergentes.

É importante salientar que, ao utilizar estratégias de consolidação, independente da escolha do método, resultará sempre em deterioração do nível de serviço. Despachos a serem consolidados são agora enviados a um ou mais terminais de consolidação, aumentando conseqüentemente o tempo de trânsito, pois a cada parada são gastos tempos para manuseio e classificação. De acordo com Braklow *et*

al. (1992), de forma geral, o manuseio em um terminal de consolidação adiciona em torno de um dia no prazo de entrega.

Para aumentar a utilização dos veículos, as cargas são consolidadas de tal forma que a maioria, senão todas, as rotas sejam operadas com veículos tão cheios quanto possível. Desta forma, muitas empresas transportadoras de carga fracionada decidem operar em uma rede do tipo *hub-and-spoke*, com o objetivo de se ter uma operação mais eficiente possível.

2.2 REDES *HUB-AND-SPOKE*

Uma rede de transporte de cargas parceladas é composta por um conjunto de terminais, que são ligados entre si por rotas que são operadas por veículos de tipo e capacidade pré-definidos, e também com uma freqüência pré-estabelecida para garantir o nível de serviço, especialmente no que diz respeito ao prazo de entrega, acordado com os clientes.

Existem dois subconjuntos básicos de terminais em uma rede de transporte de carga parcelada:

1. Terminais para consolidação de carga parcelada, chamados também de terminais de consolidação, ou ainda *hubs*, sendo estes responsáveis por agrupar cargas de diferentes terminais de origem, para terminais de destinos convergentes, que serão responsáveis pela entrega final;
2. Terminais locais, satélites ou fim de linha, que servem como ponto de entrada / saída dos despachos a serem transportados, fazendo a ligação direta entre o remetente ou destinatário dos despachos.

Um terminal de consolidação (ou *hub*), dependendo do caso, pode operar também como um terminal fim de linha, realizando coletas e entregas dentro da sua área de atuação. Já a situação inversa, que seria um terminal fim de linha, operar como um ponto para consolidação de cargas, não é comum e, em termos práticos, inviável do ponto de vista operacional e de custo.

A Figura 2.2 apresenta de forma ilustrativa uma rede do tipo *hub-and-spoke.* O termo *hub-and-spoke* é utilizado para descrever a forma com que a rede de transportes está desenhada. Em uma tradução direta, temos “cubo-e-raio” fazendo uma analogia com as rodas de uma carroça, por exemplo. Um terminal fim de linha é conectado a um terminal de consolidação (ou *hub*) por um *spoke*, representando a ligação entre esses terminais. Todos os terminais de consolidação são conectados entre si, e a conexão entre terminais fim de linha, via de regra, é realizada somente via terminais de consolidação.

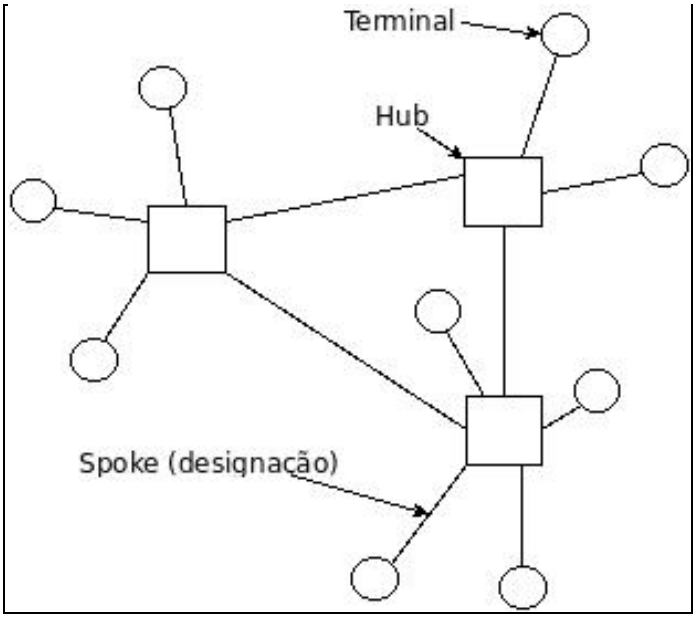

Figura 2.2: Exemplo de rede *hub-and-spoke*

O custo marginal de um despacho em um carregamento é definido como o aumento incremental de custo ao adicionar o mesmo ao carregamento. Se o veículo para a rota em questão possui excesso de capacidade, esse despacho pode então ser carregado neste veículo, e o único incremento no custo seria o de manuseio nos terminais de origem e de destino.

Entretanto, se o veículo está completamente ocupado, ou se ainda não há nenhum veículo designado para a rota, para se carregar um despacho adicional é necessário designar um novo veículo para a rota e nesse caso o custo marginal do despacho não pode ser negligenciado.

Dado que o custo marginal de um despacho é muito pequeno quando em um carregamento cujo veículo possui excesso de capacidade, qualquer política que faça aumentar a ocupação do veículo reduz o custo médio por despacho por viagem. Por esta razão a operação ponto-a-ponto pode ser justificada economicamente para transportadores de carga completa.

Conforme já descrito na Seção anterior, com os terminais enviando cargas para seus *hubs* (ou terminais de consolidação), a configuração da rede de transporte como uma rede do tipo *hub-and-spoke* aumenta o volume de carga transportado nas rotas entre *hubs,* e a economia decorrente desse tipo de operação é significativa, fazendo com que o setor de carga parcelada prefira esse tipo de rede para operação.

O custo para se mover um veículo de um terminal para outro não pode ser negligenciado. Esse custo de transporte pode ser considerado como um custo "fixo", ou melhor, para uma distância definida, o custo de um dado tipo / tamanho de veículo não muda com a quantidade de carga transportada. Considere hipoteticamente a situação em que um novo terminal é adicionado à rede. Em um sistema de transporte ponto-a-ponto, esse terminal deveria ser conectado a todos os outros terminais, incorrendo em custos diretamente proporcionais ao número de terminais existentes na rede.

Por outro lado, em uma rede *hub-and-spoke*, somente duas linhas adicionais deveriam ser operadas: uma ligando o terminal ao *hub*, e outra do *hub* ao novo terminal. É agora óbvio o motivo das empresas de transporte de carga completa utilizar o sistema de transporte ponto-a-ponto. Dado o alto custo marginal de uma carga adicional em uma operação de carga completa, qualquer alternativa diferente da operação com uma rota direta da origem até o destino, aumenta consideravelmente o custo de transporte (STARR; STINCHCOMBE, 1992).

2.3 CONSOLIDAÇÃO TÍPICA BASEADA EM UMA REDE *HUB-AND-SPOKE*

Conforme já descrito na Seção anterior, tipicamente, um despacho é coletado no seu ponto de origem, por exemplo, um cliente, e transferido para um terminal local, conhecido também por terminal satélite ou fim de linha. A rede de transporte é composta por terminais previamente estabelecidos, que servem como ponto de entrada / saída de despachos em uma rede de distribuição. Um terminal coleta cargas de todos os pontos de origem que estão dentro da sua área de controle. A coleta de despachos no seu ponto de origem é conhecida como *coleta local*.

Para minimizar os custos, os despachos que compõem uma carga podem utilizar diferentes rotas para chegar ao seu destino. As cargas em um terminal são divididas basicamente em duas categorias:

1. carga *inbound*, sendo os despachos que devem ser entregues localmente;
2. carga *outbound,* representando as cargas que devem ser entregues fora da região de atendimento do seu terminal de origem.

As cargas *outbound* são consolidadas no terminal e tipicamente transferidas para um terminal de consolidação (ou *hub*). No *hub,* as cargas são classificadas e carregadas em veículos que se encarregarão de transportá-las aos seus respectivos terminais de destino, que serão responsáveis pela entrega ao destinatário final dos despachos.

Entretanto, o fluxo da carga não é tão simples, muitas vezes são realizadas modificações no padrão apresentado acima para o fluxo da carga, visando reduzir o trabalho de manuseio nos *hubs* intermediários, e também melhorar o nível de serviço. Por exemplo, suponha que muitas cargas destinadas ao terminal j chegam ao *hub* i*.* Se um veículo pode ser suficientemente ocupado por estas cargas, então estas podem ser despachadas diretamente para o terminal j*,* eliminando a necessidade de parada no segundo *hub* ao qual estaria ligado o terminal j*.* De forma similar, algumas cargas destinadas a uma mesma região geográfica, originadas no terminal i podem ser carregadas em um único veículo e enviadas diretamente ao segundo *hub*, eliminando a parada no primeiro *hub.* E ainda cargas

podem ser enviadas diretamente do terminal de origem para o seu terminal de destino, desde que o fluxo de cargas seja suficiente para ocupar razoavelmente o veículo nesta rota.

2.4 O EQUIPAMENTO DE TRANSPORTE

Empresas transportadoras de carga parcelada utilizam basicamente dois tipos de veículos para as operações de transferência de cargas entre terminais, que são:

1. *Truck*: veículo com até 12.000 kg de capacidade de carga e 40 a 50 m³ de capacidade em volume, com uma carroceria do tipo baú.
2. *Carreta*: veículo composto por um cavalo trator e um semi-reboque, também com carroceria do tipo baú, tendo o conjunto a capacidade líquida de carga, em peso, de 25.000 kg, e em volume de 90 a 110 m³.

Nesta pesquisa, para efeito de estudo, foram considerados esses dois tipos de veículos, sendo que as ligações entre *hubs* são realizadas utilizando principalmente carretas, e na ligação *hub* - terminal, em alguns casos, é utilizado também trucks.

Esses tipos de veículo foram, portanto utilizados especialmente para as decisões envolvendo a definição de um plano de carregamento para todas as cargas a serem transportadas em uma rede *hub-and-spoke*.

Algumas empresas transportadoras de carga parcelada no Brasil iniciaram a utilização de um veículo do tipo rodotrem especialmente para a transferência de cargas entre *hubs.* Esse veículo ilustrado na Figura 2.3, é composto por um cavalo trator com dois eixos de tração (6x4), e dois semi-reboques, tendo como limite de peso bruto total combinado (PBTC) de 74 toneladas, com comprimento entre 25 e 30 metros, necessitando de AET (Autorização Especial de Trânsito).

Figura 2.3: Exemplo de carreta utilizada como veículo tipo para o estudo
Fonte: Patrus Transportes

2.5 O PROBLEMA DA CONFIGURAÇÃO DE UMA REDE DE TRANSPORTE DE CARGA PARCELADA

Dado que um modelo de rede do tipo *hub-and-spoke* será utilizado para consolidação, a rede precisa ser projetada de tal forma a não comprometer a eficiência da operação como um todo. É proposto nesta pesquisa o projeto de uma rede *hub-and-spoke* para o transporte de carga fracionada, envolvendo duas etapas básicas:

1. Em nível de decisão estratégico, a primeira etapa consiste no projeto da rede *hub-and-spoke* propriamente dita, definindo simultaneamente a quantidade e localização de terminais que operarão como pontos para consolidação de cargas (*hubs*), e também na definição da alocação ou designação dos terminais da rede aos *hubs* definidos. A quantidade de *hubs* a ser localizada é uma variável do problema, e cada terminal deve ser alocado a um único *hub*.

2. Uma vez definida a rede *hub-and-spoke* para o transporte de carga parcelada, a segunda etapa está situada em nível de decisão tático, com o problema da definição de um plano de carregamento, decidindo as rotas a serem operadas, o dimensionamento dos veículos necessários para operação, e na determinação do percurso que cada uma das cargas deve percorrer para chegar ao seu terminal de destino, incluindo as possíveis paradas intermediárias para consolidação;

Diversas outras questões operacionais tais como a programação de veículos e de motoristas, restrições de janelas de horário para coletas e entregas,

reposicionamento de veículos vazios, e o roteamento de veículos para as operações de coleta e entrega são igualmente importantes, mas estão fora do escopo desta pesquisa.

3 REVISÃO BIBLIOGRÁFICA

O problema de configuração de redes *hub-and-spoke*, que consiste na definição da quantidade e localização de terminais de transporte que servirão como pontos para consolidação de cargas, pertence a uma categoria mais geral de problemas do tipo “localização de instalações”.

De acordo com Novaes (1989), o problema da localização de instalações tem sido tratado de modo bastante amplo na literatura, englobando desde simples problemas de localização de uma única instalação, até problemas bastante complexos, englobando diversas instalações, em diversos níveis de uma cadeia produtiva ou de serviços, com fluxos de natureza diversa, problemas conhecidos no ambiente acadêmico como *location-allocation problem* e fluxos *multicommodities*.

De acordo com Gualda (1995), o problema de localização pode ser definido como um problema de alocação espacial de recursos. A hipótese básica da teoria da localização é a de que cada empresa procura escolher a localização que leve à maximização dos lucros da sua atividade.

Estas instalações podem ser fábricas, portos, pontos de venda, armazéns, lojas de varejo e centros de serviço (Ballou, 1999); como também instalações de serviço urbano, incluindo serviços de rotina e de emergência, como postos de correio, pontos de incineração de lixo, instalações de bombeiros e serviços de emergência médica, dentre outros (LARSON; ODONI, 1981).

As decisões de localização, no contexto do planejamento logístico, correspondem à determinação do número, localização e tamanho das instalações a serem usadas (Ballou, 1999).

Silva Leme (1965), apud Gualda (1995), aponta que os fatores locacionais podem ser classificados em fatores aglomerativos, fatores desaglomerativos, e o fator transporte. Os fatores aglomerativos são os que contribuem para agrupar as atividades produtivas em um determinado ponto ou local, sendo que os

desaglomerativos agem no sentido de desagrupar essas atividades, levando à localização das mesmas em mais de um ponto. O fator transporte pode agir tanto num sentido como no outro, dependendo do caso.

Nos problemas em que o fator transporte é predominante, isto é, tem grande peso nas decisões, a resolução de problemas de localização pode ser simplificada através da sua modelagem centrada no fator transporte, e as soluções assim obtidas analisadas com vistas aos demais fatores.

Aspectos básicos sobre a modelagem matemática dos problemas de localização de instalações são apresentados por Gualda (1995), Ballou (1999) e Novaes (1989), entre outros.

Uma resenha que contemplou 54 trabalhos sobre localização é apresentada por Brandeau e Chiu (1989), cobrindo desde o trabalho pioneiro de Weber que, em 1909, modelou o problema de localizar um depósito de distribuição objetivando minimizar a distância total entre o depósito e um conjunto de consumidores distribuídos espacialmente. Essa resenha inclui uma definição abrangente do problema de localização e uma classificação dos diferentes tipos de problemas de localização considerados. O objetivo é localizar instalações (e, talvez, alocar clientes a pontos de prestação de serviços) de forma a otimizar um objetivo espacialmente dependente (implícita ou explicitamente dependente). Critérios típicos utilizados para tal incluem:

- minimização do tempo médio da viagem ou da distância entre os pontos de demanda e os pontos de suprimento;
- minimização do tempo médio de resposta (tempo de viagem mais eventuais esperas para atendimento);
- minimização de uma função de custo da viagem ou do tempo de resposta;
- minimização do máximo tempo de viagem;
- maximização do mínimo tempo de viagem.

Em geral, a função objetivo consiste de termos proporcionais às distâncias (ou tempos) de viagem entre instalações e/ou entre instalações e clientes.

Segundo Gualda (1995), a resolução dos problemas de localização pode ser classificada em dois grupos, a saber:

- Métodos Indutivos, que se baseiam na análise de dados e informações estatísticas, históricas e provenientes de pesquisas de campo (questionários), através do que se busca razões ou indicações quanto à melhor localização para uma dada indústria (ou terminal, no nosso caso);

- Métodos Dedutivos, que consistem no estabelecimento de um modelo representativo da realidade, passível de tratamento matemático, para resolver o problema da localização; dados históricos ou estatísticos são usados para testar os resultados produzidos por esses modelos.

Segundo Crainic (1998), os principais modelos de localização são classificados como segue:

- *Modelos de Cobertura de Conjuntos (Set Covering)*. Localizar instalações nos vértices de uma rede de tal modo que os vértices de demanda são cobertos por uma instalação, isto é, encontram-se dentro de uma dada distância da instalação. A distância de cobertura, geralmente relacionada ao caminho mais curto entre a instalação e cada um dos pontos de demanda, pode ser o mesmo para todos os vértices, ou pode depender de uma instalação específica e pontos de demanda. Em geral, o problema corresponde a minimizar o custo de localização de instalações, sujeito a uma restrição, determinando que todos os vértices sejam cobertos. Caso se considere a situação de um orçamento fixo, então o objetivo pode ser maximizar a demanda coberta pelas instalações.

- *Modelos de Centro*. Localizar *p* instalações nos vértices de uma rede, de tal forma a minimizar a distância máxima entre um ponto de demanda e uma instalação. Para uma revisão detalhada das formulações recomenda-se o trabalho de Handler (1990), apud Crainic (1998).

- *Modelos de Mediana*. Localizar *p* instalações nos vértices de uma rede e alocar demanda a estas instalações de tal forma a minimizar a distância ponderada entre instalações e pontos de demanda. Se as instalações são não capacitadas e *p* é fixo, obtém-se então o tão conhecido problema das *p*-medianas. Desse modo, cada vértice é designado para sua instalação mais próxima. Se *p* é uma variável de decisão e as instalações são não capacitadas, isto define o Problema de Localização de Instalações Não Capacitadas.

Modelos de Cobertura de Conjunto são tipicamente associados à localização de instalações públicas, tais como centros de saúde, agências de correio, bibliotecas, e escolas. Modelos de Centro muitas vezes surgem quando é necessário estabelecer a localização de instalações de emergência tais como estações de bombeiros e estações de ambulância. Modelos de Mediana são diretamente relevantes para o projeto de serviços logísticos e distribuição de cargas.

Dadas as características desse trabalho, a atenção, aqui, está voltada para problemas em que o fator transporte é determinante, com ênfase para a sua modelagem através da utilização de métodos dedutivos. Em particular, está voltada para problemas onde o sistema de transporte envolve a utilização de terminais de transporte de carga parcelada, e a localização de pontos para consolidação de cargas (*hubs*) precisam ser determinados, em configurações do tipo *hub-and-spoke.*

3.1 REDES *HUB-AND-SPOKE* NA LITERATURA

Conforme visto no Capítulo anterior, o paradigma de distribuição *hub-and-spoke* é um sistema de conexões arranjadas como uma roda de carroça, no qual todo o tráfego é realizado via seus raios (ou *spokes*), conectado ao *hub* (cubo) no centro. Esse modelo é comumente usado na indústria, em particular no transporte pelos modais rodoviário e aéreo, inclusive de cargas, e também em redes de telecomunicações, bem como na área de computação, especificamente em programação paralela e distribuída.

O termo *hub-and-spoke* é utilizado para descrever a forma com que a rede de transportes está desenhada. Em uma tradução direta, temos “cubo-e-raio” fazendo uma analogia com as rodas de uma carroça, por exemplo. De acordo com Aykin (1995), o desenvolvimento de redes do tipo *hub-and-spoke* é uma das inovações mais importantes na indústria, e reconhecida como a sétima melhor idéia na série “Grandes Idéias da Década de 80 em Marketing” pela Associação Americana de Marketing.

Como exemplo de benefício direto em uma rede do tipo *hub-and-spoke*, pode-se exemplificar que, para que um grafo não direcionado com n nós esteja completamente conectado, são necessários $n(n-1)/2$ arcos. Se o grafo for direcionado, a quantidade de arcos necessária é $n(n-1)$.

Se no mesmo grafo for adicionado um nó que operará como um *hub*, a quantidade de arcos necessários para ligar todos os nós, agora utilizando o nó *hub* como ponto intermediário, é de somente *n* arcos. Na Figura 3.1 é apresentado o exemplo de uma rede com seis nós, ilustrando as alternativas de ligação com e sem *hub* intermediário. Na rede ponto-a-ponto, são necessários 15 arcos para ligar todos os nós. Já na rede em que existe um *hub*, a quantidade de arcos necessários para ligar todos os nós é de seis arcos.

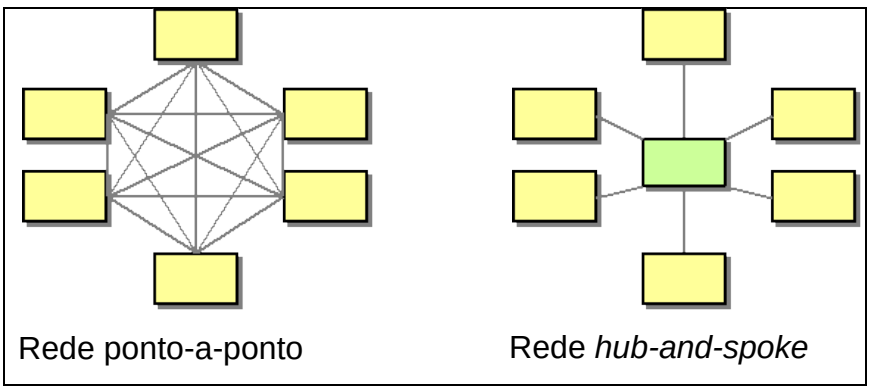

Figura 3.1: Exemplo de rede ponto a ponto e *hub-and-spoke*[1]

[1] Ilustrações extraídas do artigo de Gregor Hohpe com título “Hub and Spoke [or] Zen and the Art of Message Broker Maintenance”,
disponível em: http://www.eaipatterns.com/ramblings/03_hubandspoke.html
Acesso: 20 mai. 2010.

A empresa americana de encomendas expressas pelo modal aéreo, Federal Express, é um dos casos de maior sucesso na utilização de redes do tipo *hub-and-spoke.* Segundo Chan e Ponder (1979), um dos principais fatores de sucesso da Federal Express desde sua fundação em 1973, foi justamente a utilização de uma rede do tipo *hub-and-spoke*, centralizando as operações de consolidação e classificação de pacotes, inicialmente, em um único *hub* localizado em Memphis,TN.

O problema de localização de *hubs* (ou *hub location problem*) consiste em definir terminais de carga que operarão como *hubs*, e na alocação de terminais de ponta (ou fim de linha) de demanda a esses *hubs*, definindo a rota para cada fluxo entre terminais de origem e destino. Há dois tipos básicos de rede *hub-and-spoke*: com *alocação única* ou com *alocação múltipla*. Na rede *hub-and-spoke* com alocação única cada terminal é alocado a um único *hub* e todo o fluxo enviado ou recebido de outros terminais é direcionado exclusivamente a esse *hub*. Já no caso da alocação múltipla, um terminal pode estar alocado a um ou mais *hubs*, ou seja, um terminal pode enviar ou receber cargas por mais de um *hub*.

Alguns trabalhos encontrados na literatura estudam somente o aspecto relacionado à alocação dos terminais aos *hubs*. Como exemplo, o trabalho de Kuby e Gray (1993), que analisou o caso da empresa Federal Express no transporte aéreo de encomendas expressas nos Estados Unidos, em que a quantidade e localização dos *hubs* já estavam pré-definidas.

Contudo, dado que a alocação ótima dos terminais é afetada pela localização dos *hubs*, e também a localização ótima de *hubs* é afetada pelas decisões de alocação, os problemas de localização e alocação devem ser considerados conjuntamente no projeto de uma rede *hub-and-spoke*.

Os trabalhos encontrados na literatura relativos ao problema de configuração de redes *hub-and-spoke* têm como premissas: (i) uma rede completa com arcos ligando todos os pares de *hubs*; (ii) que existe economia de escala no custo unitário de transporte nas ligações entre *hubs* representada por um fator de desconto α ao

utilizar as conexões entre *hubs;* (iii) e que nenhum serviço direto (ligando dois terminais não-*hub*) é permitido.

Outras variantes do problema, como por exemplo o problema de localização de *hubs* no espaço contínuo, ou no plano, não fazem parte do escopo desta pesquisa. Diversos trabalhos estudaram esse tipo de problema, como por exemplo em Aykin (1988, 1995), Campbell (1990), O'Kelly e Miller (1991), e Aykin e Brown (1992).

Alumur e Kara (2008) fizeram uma resenha que contemplou mais de 100 artigos relacionados ao problema de localização de *hubs,* sendo que a quantidade de trabalhos publicados têm crescido a cada ano, especialmente na última década.

Com base nos trabalhos publicados na literatura, problemas de localização de *hubs*, ou de configuração de redes do tipo *hub-and-spoke*, podem ser classificados nos seguintes tipos:

- **USApHMP**: *uncapacitated single allocation p-hub median problem*, consistindo na localização de *p hubs*, sendo que cada terminal da rede deve ser alocado a um único *hub*. Para esse problema, a quantidade "*p"* de *hubs* a ser localizada já está pré-definida, sendo um dado de entrada para solução desse problema. Nesse caso os *hubs* não possuem nenhuma restrição de capacidade, e o custo fixo para abertura de instalações que operarão como *hubs* é desconsiderado, dado que o número de *hubs* a ser selecionado é dado *a priori*.
- **UMApHMP**: *uncapacitated multiple allocation p-hub median problem*, similar ao USApHMP, sendo que nesse caso um terminal não-*hub* pode estar alocado a mais de um *hub*.
- **USAHLP**: *uncapacitated single allocation hub location problem*, problema de localização de *hubs* não capacitado com alocação única. Similar ao USApHMP, sendo que a quantidade de *hubs* (*p*) a serem localizados não é mais um dado de entrada, mas sim uma das decisões a ser definidas na solução do problema. Nesse caso, existem custos fixos associados à abertura de *hubs*.
- **UMAHLP**: *uncapacitated multiple allocation hub location problem*, similar ao USAHLP, permitindo alocação múltipla, ou seja, um terminal não-*hub* pode

estar alocado a mais de um *hub*.

- **CSAHLP**: *capacitated single allocation hub location problem*, similar ao USAHLP, mas considerando restrições de capacidade, no que diz respeito à quantidade de carga movimentada em cada um dos *hubs.*
- **CMAHLP**: *capacitated multiple allocation hub location problem*, diferenciando-se do CSAHLP por permitir a alocação múltipla, um terminal não-*hub* pode estar alocado a mais de um *hub*.
- ***p-Hub Center Problem***: nesse caso a configuração de uma rede *hub-and-spoke* é um problema *"minimax"* similar ao problema de localização de *p* centros de distribuição, podendo ser de um dos três tipos básicos:
    1. a minimização do custo máximo para qualquer par origem - destino;
    2. a minimização do custo máximo no transporte:
        2.1. terminal de origem $\rightarrow$ hub;
        2.2. entre hubs;
        2.3. hub $\rightarrow$ terminal de destino;
    3. similar ao *vertex center* descrito em Hakimi (1965), no qual o conjunto de *hubs* minimiza o custo máximo para movimento entre terminal de origem $\rightarrow$ hub e hub $\rightarrow$ terminal de destino.
- ***Hub Covering Problem***: problema de configuração de uma rede do tipo *hub-and-spoke,* em que terminais de demanda são considerados cobertos se estiverem dentro de uma distância específica do *hub*. São definidos três tipos básicos para cobertura. O par origem destino (i, j) é coberto pelos *hubs* k e m se:
    - o custo de i para j via *hubs* k e m não exceder a um valor específico;
    - o custo para cada arco no caminho de i para j via *hubs* k e m não exceder a um valor específico;
    - Se o custo para cada arco terminal de origem-*hub* e *hub*-terminal de destino não exceder um valor específico, sendo que esse valor pode ser distinto para cada um desses arcos.

Como mencionado anteriormente, o foco da presente pesquisa é na solução do USAHLP, que incorpora aspectos importantes de um problema prático de uma transportadora de carga parcelada, como por exemplo a definição da quantidade e

localização de *hubs* no modelo, e também na apropriação de custos fixos ao se definir terminais que servirão como pontos para consolidação e transbordo de cargas parceladas. Desta forma, portanto, uma ênfase maior foi dada na revisão bibliográfica específica para esse tipo de problema.

A pesquisa relacionada à localização de *hubs* iniciou-se com o trabalho pioneiro de O'Kelly (1986) estudando o problema de localização de um ou dois *hubs,* apresentando aplicações em redes de transporte de passageiros e de encomendas expressas pelo modal aéreo. Talvez Goldman (1969) seja o primeiro trabalho sobre o problema de localização de *hubs*.

Entretanto, O'Kelly (1987) é reconhecido como o primeiro a apresentar a formulação matemática para o problema de localização de *hubs* estudando uma rede de transporte aéreo de passageiros. Sua formulação matemática é referenciada como o problema de localização de *p hubs* com alocação única e sem restrições de capacidade (*uncapacitated single allocation p-hub median problem*, ou USApHMP).

3.1.1 Literatura específica referente ao USAHLP

O'Kelly (1992) introduziu o problema de localização de *hubs* não capacitado com alocação única (USAHLP), incorporando os custos fixos na função objetivo, e fazendo com que a quantidade e localização dos *hubs* seja uma variável de decisão do modelo matemático. O problema foi modelado como um problema de programação inteira quadrática e foi uma generalização do USApHMP.

Segundo o autor, seja N o conjunto de nós da rede e W_{ij} o fluxo entre os terminais i e $j,\ (i,j) \in N,\ C_{ij}$ o custo de transporte por unidade de fluxo entre i e $j,\ (i,j) \in N,\ F_j$ o custo fixo ao se estabelecer um *hub* no terminal $j,\ j \in N.$ Define-se X_{ik} como sendo igual a 1 se o terminal i for alocado ao *hub* $k,\ (i,k) \in N$, e zero caso contrário; X_{kk} assume valor 1 se o terminal $k,\ k \in N$ for um *hub* e zero caso contrário. Assim, modelo matemático proposto por O'Kelly (1992) para representar o USAHLP foi:

Minimizar

$$\sum_i \sum_j W_{ij} \left(\sum_k X_{ik} C_{ik} + \sum_m X_{jm} C_{jm} + \alpha \sum_k \sum_m X_{ik} X_{jm} C_{km} \right) + \sum_j X_{jj} F_j \tag{3.1}$$

sujeito às restrições:

$$\sum_k X_{ik} = 1, \quad \forall i, \tag{3.2}$$

$$X_{ij} \leq X_{jj}, \quad \forall i, j \tag{3.3}$$

$$X_{ik} \in \{0, 1\}, \quad \forall i, k \tag{3.4}$$

A função objetivo (3.1) possui três parcelas principais de custo, sendo a primeira delas relativa ao custo de transporte na ligação terminal $\leftrightarrow$ *hub*, a segunda parcela diz respeito ao custo de transporte na ligação entre *hubs*, e a última parcela apropria o custo fixo ao definir um terminal com um ponto para consolidação de cargas, ou *hub*. O parâmetro α no terceiro termo representa o fator de economia de escala na ligação entre *hubs*; o custo do fluxo entre *hubs* deve ser menor que o custo original dado que as instalações definidas como *hub* concentram fluxo, portanto $0 \leq \alpha < 1$. É importante ressaltar que a função objetivo é quadrática devido ao fato de que o desconto na ligação entre *hubs* é um produto das decisões de alocação.

As restrições (3.2) impõem que cada terminal seja alocado a exatamente um *hub*. Já as restrições (3.3) asseguram que um terminal i seja alocado a outro terminal j somente se esse for um *hub.*

Campbell (1994) propôs novas formulações matemáticas para o problema de configuração de redes *hub-and-spoke*, incluindo para o USAHLP, sendo o primeiro trabalho a modelar matematicamente o problema de forma linear, abrindo as primeiras perspectivas e estudos para solução desse tipo de problema utilizando-se métodos exatos.

Segundo o autor, X_{ijkm} corresponde à fração do fluxo do terminal i para o terminal j que é enviado via *hubs* k e m, nesta ordem, e $C_{ijkm} = C_{ik} + C_{mj} + \alpha C_{km}$ o custo de

transporte de i para j. Utilizando-se os parâmetros e variáveis já definidos anteriormente no trabalho de O'Kelly (1992), tem-se que o modelo matemático de Campbell (1994) para representar o USAHLP foi:

$$\text{Min} \quad \sum_i \sum_j \sum_k \sum_m W_{ij} X_{ijkm} C_{ijkm} + \sum_k F_k X_{kk} \tag{3.5}$$

Sujeito às restrições:

$$\sum_k \sum_m X_{ijkm} = 1, \quad \forall i, j, \tag{3.6}$$

$$X_{ijkm} \leq X_{kk}, \quad \forall i, j, k, m, \tag{3.7}$$

$$X_{ijkm} \leq X_{mm}, \quad \forall i, j, k, m, \tag{3.8}$$

$$X_{ik} \in \{0, 1\}, \quad \forall i, k, \tag{3.9}$$

$$0 \leq X_{ijkm} \leq 1 \quad \forall i, j, k, m. \tag{3.10}$$

A função objetivo (3.5) visa minimizar o custo variável de transporte e custo fixo, sendo o custo variável associado ao fluxo entre terminais e *hubs,* e o custo fixo associado à definição da localização de *hubs*. As restrições (3.6) visam assegurar que um terminal i seja designado a um único *hub*. Já as restrições (3.7) e (3.8), da mesma forma que em (3.3), asseguram que um terminal deva ser alocado exclusivamente a um *hub*.

Baseado em uma verificação de que a formulação proposta por Campbell (1994) resultava em soluções altamente fracionadas, Skorin-Kapov, D., Skorin-Kapov, J. e O'Kelly (1996) propuseram uma nova formulação matemática para o USAHLP. Utilizando-se a mesma nomenclatura de parâmetros e variáveis das formulações apresentadas anteriormente, têm-se que o modelo matemático de Skorin-Kapov, D., Skorin-Kapov, J. e O'Kelly (1996) para representar o USAHLP foi:

$$\text{Min} \quad \sum_i \sum_j \sum_k \sum_m W_{ij} X_{ijkm} C_{ijkm} + \sum_k F_k X_{kk} \tag{3.11}$$

Sujeito às restrições:

$$\sum_k X_{ik} = 1, \quad \forall i, \tag{3.12}$$

$$X_{ik} \leq X_{kk}, \ \ \forall i,k, \tag{3.13}$$

$$\sum_{m} X_{ijkm} = X_{ik} \ \ \forall i,j,k, \tag{3.14}$$

$$\sum_{k} X_{ijkm} = X_{jm} \ \ \forall i,j,m, \tag{3.15}$$

$$X_{ijkm} \geq 0, \ \ \forall i,j,k,m, \tag{3.16}$$

$$X_{ik} \in \{0,1\} \ \ \forall i,k. \tag{3.17}$$

A formulação matemática compreende $(n^4 + n^2)$ variáveis, das quais n^2 binárias. Os autores, ao resolver o problema, ainda mostraram que, ao se utilizar esta formulação matemática relaxando-se o conjunto de restrições representado pela equação (3.17), foram capazes de encontrar uma grande quantidade de soluções inteiras utilizando-se os dados do *Civil Aeronautics Board data set* (CAB). Para as instâncias nas quais não foi possível obter a solução ótima diretamente do problema relaxado, o valor da função objetivo resultante ficou menos de 1% da solução ótima do problema.

Abdinnour-Helm e Venkataramanan (1998) propuseram uma nova formulação matemática inteira baseada na idéia de fluxos *multicommodity* em redes.

Sejam os parâmetros e variáveis de decisão:

- w_{kl} : a quantidade de carga a ser levada do terminal k ao terminal l, $\forall\ k = 1,\ldots,n,\ l = 1,\ldots,n$;
- d_{ij} : a distância do terminal i ao terminal j, $\forall\ i = 1,\ldots,n,\ j = 1,\ldots,n$;
- f_i : custo fixo se o terminal i se tornar um *hub*, $\forall\ i = 1,\ldots,n$;
- n : quantidade total de terminais da rede;
- M : um número inteiro positivo grande;
- (kl) : índice para denotar o fluxo de k para l (*commodity* kl), onde k é a origem e l o destino;
- e_{ij} : arco do terminal i ao terminal j;
- $x_{ij(kl)}$: variável binária assumindo valor igual a 1 se o arco e_{ij} é utilizado para transportar a *commodity* (ou carga) (kl), $\forall\ k = 1,\ldots,n,\ l = 1,\ldots,n$ e $k \neq 1$, ou 0 caso contrário $\forall\ i = 1,\ldots,n,\ j = 1,\ldots,n$ e $i \neq j$;

- y_i : variável binária assumindo valor igual a 1 se o terminal i se tornar um *hub*, ou 0 caso contrário $\forall\ i = 1, \ldots, n$;
- z_{ij} : variável binária assumindo valor igual a 1 se o terminal i é alocado ao terminal j, ou 0 caso contrário $\forall\ i = 1, \ldots, n,\ j = 1, \ldots, n$ e $i \neq j$;

A formulação matemática para representar o USAHLP proposta por Abdinnour-Helm e Venkataramanan (1998) foi:

$$Min\ \ \mathcal{Z} = \sum_{k=1}^{n} \sum_{\substack{l=1 \\ l \neq k}}^{n} w_{kl} \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij(kl)} \left(1 - y_i\, y_j + \alpha\, y_i\, y_j\right) \right) + \sum_{i=1}^{n} f_i y_i \qquad (3.18)$$

Sujeito às restrições:

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} z_{ij} + y_i = 1, \quad \forall\, i = 1, \ldots, n, \qquad (3.19)$$

$$M y_i \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} z_{ji}, \quad \forall i = 1, \ldots, n, \qquad (3.20)$$

$$\sum_{\substack{j=1 \\ j \neq k}}^{n} x_{kj(kl)} = 1, \quad \forall\, (kl)\ k = 1, \ldots, n,\ l = 1, \ldots, n\ , k \neq l \qquad (3.21)$$

$$\sum_{\substack{j=1 \\ j \neq l}}^{n} x_{jl(kl)} = 1, \quad \forall\, (kl)\ k = 1, \ldots, n,\ l = 1, \ldots, n\ , k \neq l \qquad (3.22)$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij(kl)} - \sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ji(kl)} = 0, \forall j = 1, \ldots, n\ , j \neq k, j \neq l,$$
$$\forall\, (kl)\ k = 1, \ldots, n,\ l = 1, \ldots, n\ , k \neq l, \qquad (3.23)$$

$$x_{ij(kl)} \leq y_i \cdot y_j + z_{ij} + z_{ji}\ \ \forall\ i = 1, \ldots, n, \forall\ j = 1, \ldots, n\ , j \neq i,$$
$$\forall\ (kl)\ k = 1, \ldots, n,\ l = 1, \ldots, n\ , k \neq l, \qquad (3.24)$$

$$x_{ij(kl)} \in (0, 1) \quad \forall i = 1, \ldots, n,\ \forall j = 1, \ldots, n\ , j \neq i,$$
$$\forall\ (kl)\ k = 1, \ldots, n,\ l = 1, \ldots, n\ , k \neq l, \qquad (3.25)$$

$$y_i \in (0, 1) \quad \forall i = 1, \ldots, n, \qquad (3.26)$$

$$z_{ij} \in (0,1) \quad \forall i = 1,\ldots,n,\ \forall j = 1,\ldots,n\ , j \neq i. \tag{3.27}$$

A função objetivo (3.18) minimiza o custo total, que é a soma dos custos de transporte e custo fixo para a localização de *hubs*. A restrição (3.19) assegura que cada terminal i é, ou será designado a, um *hub*. Já as restrições (3.20) asseguram que o terminal i se torne um *hub* se houver pelo menos um outro terminal designado a i. As restrições de (3.21) a (3.23) garantem a condição de se encontrar o caminho mais curto para cada par de terminais (kl): a *commodity* (kl) deixa o terminal de origem k (restrição (3.21)); chega ao terminal de destino l (restrição (3.22)); e satisfaz a restrição de conservação do fluxo a cada terminal intermediário no caminho de k para l (restrição (3.23)).

As restrições (3.24) asseguram que, se o arco e_{ij} é utilizado para o transporte da *commodity* (kl), então os terminais i e j são ambos *hubs*, ou um é designado ao outro. As restrições de (3.25) a (3.27) garantem que as variáveis do problema assumam valores 0 ou 1.

Dado que o algoritmo B&B necessitou uma quantidade considerável de tempo, os autores também propuseram uma heurística baseada em algoritmos genéticos (AG) objetivando encontrar soluções eficientes de forma rápida, sendo esse trabalho a primeira tentativa de solução do USAHLP utilizando uma metaheuristica.

Nos experimentos computacionais apresentados, além do conjunto de dados CAB, foram também resolvidos problemas gerados aleatoriamente com 10, 15, 50 e 80 terminais. O algoritmo B&B pôde encontrar a solução ótima para problemas com até 25 terminais do CAB, e ainda sendo necessário mais de 24 horas de processamento para sua convergência. Para os problemas com 50 e 80 terminais, o B&B não conseguiu convergir para a solução ótima após 72 horas de processamento.

O AG de Abdinnour-Helm e Venkataramanan (1998) conseguiu obter soluções aproximadas de forma rápida, obtendo a convergência do AG para o problema de 80 terminais em menos de 8 segundos. Resolvendo os problemas tradicionalmente estudados, utilizando o conjunto de dados CAB, pôde-se obter um tempo de

processamento inferior a um segundo e com um desvio médio de 0,4% da solução ótima. Para as instâncias geradas aleatoriamente, a qualidade das soluções obtidas com o AG foram comparadas somente com relação ao percentual de desvio em relação ao limitante inferior do B&B, dado que o algoritmo B&B não pôde encontrar a solução ótima para os problemas com mais de 15 terminais (desse conjunto de instâncias).

Abdinnour-Helm (1998) propôs uma nova heurística híbrida baseada em algoritmos genéticos (AG) e busca tabu (BT) para solução do USAHLP. A heurística, denominada pelo autor de GATS (*Genetic Algorithm and Tabu Search*), consistiu em dividir o problema original em duas partes: a primeira parte consistiu na definição da quantidade e localização de *hubs*, e a segunda parte buscou resolver o problema de alocação dos terminais aos *hubs* localizados na primeira etapa.

Para a solução do problema da definição da quantidade e localização dos *hubs*, o autor utilizou AG, e para solução da segunda parte do problema, que consistiu na definição da alocação dos terminais aos *hubs* localizados, foi utilizada a busca tabu. Os resultados obtidos foram comparados com o AG de Abdinnour-Helm e Venkataramanan (1998) utilizando o conjunto de dados CAB. Os resultados obtidos com os experimentos computacionais, utilizando a busca tabu em combinação com o AG, foram melhores ao se comparar os problemas resolvidos unicamente com o AG.

A deficiência encontrada na heurística desenvolvida por Abdinnour-Helm e Venkataramanan (1998) foi devido à busca tabu ter sido executada somente no melhor indivíduo de cada geração do AG, tendo esse indivíduo, um *fitness* (valor da função objetivo) baseado na alocação dos terminais não-*hub* aos *hubs* mais próximos. Em suma, para alguns problemas, a busca tabu foi executada em uma solução que não continha a quantidade e localização correta de *hubs*.

Labbé e Yaman (2004) modelaram matematicamente o USAHLP utilizando variáveis de fluxo *multicommodity*, de forma similar ao trabalho de Ernst e Krishnamoorthy (1999) para o CSAHLP. Projetando estas variáveis de fluxo, conseguiram determinar

alguns raios extremos dos cones de projeção, conseguindo derivar algumas inequações correspondentes que definiam faces do poliedro do USAHLP.

Labbé e Yaman (2004) projetaram as variáveis de fluxo utilizando dois métodos distintos, sendo o primeiro deles baseado no trabalho de Mirchandani (2000), produzindo inequações conhecidas como *inequações métricas*. O segundo método de projeção das variáveis de fluxo foi baseado no trabalho de Rardin e Wolsey (1993) e consistiu na substituição das restrições de fluxo pelas restrições de corte correspondentes, para em seguida realizar a projeção das variáveis.

Topcuoglu et al. (2005) propuseram um novo AG para o USAHLP, apresentando desempenho superior em relação à heurística híbrida GATS de Abdinnour-Helm (1998), tanto no que diz respeito tanto na qualidade da solução final obtida, como também no tempo computacional necessário para se resolver os problemas utilizando os dados do CAB e do AP.

O AG desenvolvido pelos autores utilizou um cromossomo contendo dois vetores, sendo o primeiro deles com a informação da quantidade e localização de *hubs*, e no segundo vetor a alocação dos terminais não-*hub* aos *hubs* localizados. As operações tradicionais do AG, como por exemplo as operações de cruzamento, foram realizadas em ambos os vetores para simultaneamente explorar as duas partes do problema (localização / alocação).

O desempenho do AG de Topcuoglu et al. (2005) foi ligeiramente superior no que diz respeito à qualidade da solução final obtida, quando comparado com o GATS de Abdinnour-Helm (1998), mas não conseguindo ainda encontrar a solução ótima para todos os problemas estudados. Como exemplo, o AG de Topcuoglu et al. (2005) obteve melhores soluções em apenas 4 dos 80 problemas resolvidos com o conjunto de dados CAB.

Tanto o AG de Topcuoglu et al. (2005) como o GATS de Abdinnour-Helm (1998) possuem uma deficiência no método para geração da quantidade e localização de *hubs*, pois para alguns problemas, o AG não pôde obter a quantidade e localizações ótimas para os *hubs*. Como exemplo o problema do CAB contendo 25 terminais,

custo fixo igual a 100 e fator de desconto $\alpha = 1.0$, os autores reportam um custo de 1559,19 com dois hubs localizados nos terminais 7 e 19. A solução ótima desse problema possui custo de 1556,63 com três hubs localizados nos terminais 4, 8 e 20.

Cunha e Silva (2007) propuseram um novo AG combinado com uma heurística conhecida como Metropolis (METROPOLIS et al., 1953), superando os resultados obtidos nos trabalhos de Abdinnour-Helm (1998) e Abdinnour-Helm e Venkataramanan (1998).

A heurística híbrida, descrita em Cunha e Silva (2007), consistiu na utilização do AG para resolver a parte locacional do problema, ou seja, na definição da quantidade e localização de *hubs*, da mesma forma que em Abdinnour-Helm (1998) e Topcuoglu et al. (2005). A parte alocacional do problema, que define a alocação dos terminais aos *hubs*, foi resolvida utilizando-se uma heurística de busca local incorporada com o critério de Metropolis (METROPOLIS et al., 1953).

Uma diferença crucial, ao se comparar o trabalho de Cunha e Silva (2007) com o trabalho de Abdinnour-Helm (1998), diz respeito à busca local combinada com a heurística Metropolis, que foi utilizada em todos os novos indivíduos gerados pelo AG e não somente nos melhores indivíduos de cada geração. Com isso pôde-se obter um desempenho significativamente superior, sendo evidenciado nos experimentos computacionais apresentados.

Adicionalmente, Cunha e Silva (2007) resolveram o problema prático de uma empresa transportadora de carga parcelada no Brasil. Para representar corretamente a realidade do problema, o fator de desconto α na ligação entre *hubs* não foi fixo, mas sim dependente do volume de carga na ligação entre os *hubs*. Com isso, a função objetivo deixou de ser linear, inviabilizando a utilização direta de *solvers*, como o CPLEX, por exemplo, para obtenção da solução do problema.

Outra heurística híbrida para esse problema foi proposta por Chen (2007), que utilizou *Simulated Annealing (*SA), lista tabu e procedimentos de melhoria, para resolver o USAHLP, superando os resultados obtidos em Topcuoglu et al. (2005)

tanto em termos da qualidade da solução final obtida, como também no tempo de processamento necessário para encontrar as soluções.

No trabalho de Chen (2007) foi desenvolvido um mecanismo para definição de um limitante superior para a definição da quantidade de *hubs* a serem localizados, e criou um mecanismo baseado nos fluxos originados e destinados em cada terminal, e na distância relativa de um terminal aos outros terminais da rede, como critério para a seleção dos terminais que serão inicialmente escolhidos como *hubs*.

Na solução da parte alocacional do problema, inicialmente os terminais foram alocados ao *hub* mais próximo para em seguida testar diferentes alocações, permitindo que uma nova alocação seja aceita se esta melhorar o valor da função objetivo ou analisar, utilizando o critério da metaheurística Metropolis, se um movimento que piorasse o valor da função objetivo seria aceito.

Os experimentos computacionais de Chen (2007) demonstraram que a heurística desenvolvida teve um desempenho superior ao se comparar com a heurística apresentada no trabalho de Topcuoglu et al. (2005), especialmente no que diz respeito à qualidade das soluções finais obtidas para alguns problemas do conjunto de dados CAB, e também com relação aos tempos de processamento necessários para se resolver problemas de maior porte do conjunto de dados AP.

3.1.2 Outros problemas de projeto de redes do tipo *hub-and-spoke*

Nesta seção são apresentados os principais trabalhos desenvolvidos para os demais problemas que envolvem a localização de *hubs*, nas suas principais variantes. O objetivo aqui não é o de se esgotar a literatura relativa ao tema, mas sim dar uma direção do estado-da-arte no que diz respeito aos modelos matemáticos e aos métodos de solução desenvolvidos para cada tipo de problema. Maiores detalhes de cada um dos problemas, bem como uma revisão completa da literatura relativa ao projeto de redes *hub-and-spoke* podem ser encontrados em Alumur e Kara (2008).

3.1.2.1 USApHMP

No USApHMP, ao contrário do USAHLP, o número de *hubs* a serem localizados é pré-determinado, portanto a função objetivo não considera a parcela de custos fixos, uma vez que o número de *hubs* é fixo.

No que diz respeito ao número de variáveis e restrições, Ebery (2000) fornece a melhor formulação matemática para o problema. Contudo, a melhor formulação matemática, no aspecto relativo ao tempo de processamento necessário para a obtenção das soluções ótimas, é a apresentada em Ernst e Krishnamoorthy (1996).

O método exato considerado o mais eficiente foi proposto por Ernst e Krishnamoorthy (1998b), e consistiu na utilização do algoritmo do caminho mínimo combinado com algoritmo *branch-and-bound*, sendo que o maior problema resolvido obtendo a solução ótima foi o problema com 100 terminais.

A heurística que conseguiu obter melhores soluções para o USApHMP é baseada em relaxação lagrangiana apresentada em Pirkul e Schilling (1998). No que diz respeito às metaheurísticas desenvolvidas para o USApHMP, destacam-se a busca tabu desenvolvida por Skorin-Kapov,D e Skorin-Kapov, J (1994); e *simulated annealing* apresentada em Ernst e Krishnamoorthy (1996).

Maiores detalhes sobre os métodos de solução já desenvolvidos para o USApHMP podem ser encontrados em Klincewicz (1991, 1992), Sohn e Park (1997, 1998, 2000) e Abdinnour-Helm (2001).

3.1.2.2 UMApHMP

O UMApHMP, da mesma forma que o USApHMP, consiste em definir uma quantidade fixa e pré-determinada de *hubs* a serem localizados, sendo que para esse problema é permitida a alocação múltipla, ou seja, um terminal não-*hub* pode enviar e receber cargas por mais de um *hub*.

A formulação matemática que utiliza menor tempo de processamento para a obtenção de soluções ótimas para o UMApHMP foi proposta por Ernst e Krishnamoorthy (1998a), que utilizaram as mesmas idéias já desenvolvidas para o USApHLP, descritas em Ernst e Krishnamoorthy (1996).

O melhor método para solução do UMApHMP foi apresentado em Ernst e Krishnamoorthy (1998b), que consistiu na obtenção de limitantes inferiores para o algoritmo *branch-and-bound,* resolvendo o problema do caminho mínimo, em vez de se resolver o problema de programação linear resultante da relaxação das restrições de integralidade das variáveis do modelo matemático.

Os autores ainda destacam que, se a quantidade e localização de *hubs* for conhecida de antemão, o UMApHMP pode ser resolvido de forma trivial, utilizando-se o algoritmo do caminho mínimo sucessivo.

Outros trabalhos também trataram do UMApHMP, destacando Campbell (1992, 1996), Sasaki, Suzuki e Drezner (1999) e Boland et al. (2004).

3.1.2.3 UMAHLP

O UMAHLP é um problema similar ao USAHLP, diferenciando-se somente no que diz respeito à alocação dos terminais não-*hub* aos *hubs* localizados, que nesse caso permite que um terminal envie e receba cargas por mais de um *hub.*

Mayer e Wagner (2002) desenvolveram um novo método de solução do UMAHLP baseado no algoritmo *branch-and-bound*, denominado HubLocator. Foram resolvidos os problemas utilizando os conjuntos de dados CAB e AP, e os resultados obtidos com o HubLocator foram comparados com o CPLEX utilizando uma adaptação da formulação matemática proposta por Ernst e Krishnamoorthy (1998a).

O HubLocator se mostrou um método eficiente para solução do UMAHLP apesar do fato de nem sempre superar o tempo de processamento obtido ao se resolver o

problema utilizando o CPLEX, podendo também indiretamente confirmar a eficácia da formulação matemática proposta por Ernst e Krishnamoorthy (1998a).

A heurística mais eficiente, no que diz respeito ao tempo computacional necessário para obtenção de soluções, foi proposta por Cánovas, García e Marín (2007), que desenvolveram uma heurística baseada na técnica de ascensão dual. Foram realizados experimentos computacionais com os conjuntos de dados CAB e AP, sendo capazes de encontrar a solução ótima para problemas com até 120 terminais.

Para um maior aprofundamento, sugerem-se os trabalhos de Klincewicz (1996), Boland et al. (2004), Hamacher et al. (2004) e Marín, Cánovas e Landete (2006).

3.1.2.4 CSAHLP

O CSAHLP é muito similar ao USAHLP, diferenciando-se somente pelo fato dos terminais de consolidação possuirem restrições de capacidade, especificamente no que diz respeito à quantidade de carga a ser classificada nos *hubs*.

A formulação matemática mais eficiente, no que diz respeito ao tempo necessário para obtenção de soluções ótimas, foi proposta por Ernst e Krishtamoorthy (1999), utilizando o conceito de fluxos *multicommodity* para representar as variáveis de decisão do modelo.

A melhor estratégia para solução do CSAHLP foi também proposta em Ernst e Krishnamoorthy (1999), que utilizaram *simulated annealing* e uma busca local aleatória para a obtenção de soluções aproximadas para os problemas. As soluções obtidas com essas heurísticas foram utilizadas como limitante superior para a obtenção da solução ótima utilizando o algoritmo *branch-and-bound*. Os autores resolveram problemas do conjunto de dados AP com até 200 terminais, tendo esse conjunto de dados custos fixos e capacidades de dois tipos: *loose* e *tight*.

Para um melhor detalhamento sobre o CSAHLP, recomendam-se os trabalhos de Aykin (1994, 1995); Labbé, Yaman e Gourdin (2005) e Costa, Captivo e Climaco (2007).

3.1.2.5 CMAHLP

O CMAHLP é similar ao UMAHLP, diferenciando-se somente por permitir que um terminal não-*hub* seja designado a mais de um *hub*. É apropriado um custo fixo quando um terminal é escolhido para operar como um *hub*, possuindo também restrições de capacidade no que diz respeito à quantidade de carga a ser classificada.

O modelo matemático mais eficiente, do ponto de vista do tempo necessário para obtenção de soluções ótimas, para solução do CMAHLP foi proposta por Marín (2005). Marín (2005) apresenta os resultados obtidos dos experimentos computacionais realizados em problemas criados, utilizando dados do conjunto de dados AP, para redes com até 40 terminais.

A melhor estratégia para solução do CMAHLP foi proposta por Ebery et al. (2000), que desenvolveram uma heurística baseada no algoritmo do caminho mínimo. As soluções obtidas com a heurística foram utilizadas como limitantes superiores para o *branch-and-bound.*

Outras referências podem ser encontradas em Sasaki e Fukushima (2003) e Boland et al. (2004).

3.1.2.6 Problemas de centro

Campbell (1994) foi o primeiro trabalho a pesquisar e discutir o problema chamado de *p-hub center problem.* O melhor modelo matemático para representar o problema, no que diz respeito à quantidade de variáveis e restrições, é creditado ao trabalho de Ernst et al. (2002).

Ernst et al. (2002) estudaram o problema com alocação única e múltipla, desenvolvendo heurísticas para solução de ambos os problemas. O método proposto para solução do problema com alocação múltipla é considerado ainda a melhor estratégia para solução. A abordagem utilizada para solução do problema com alocação múltipla foi similar à utilizada em Ernst e Krishnamoorthy (1998b), utilizando o algoritmo do caminho mínimo sucessivo em conjunto com o algoritmo *branch-and-bound*.

Meyer, Ernst e Krishnamoorthy (2009) estudaram a versão com alocação única do problema e propuseram uma estratégia de solução em duas fases, utilizando conjuntamente a metaheurística colônia de formigas e B&B, sendo esse o método mais eficiente para solução do problema.

Para um maior aprofundamento, sugerem-se os trabalhos de Kara e Tansel (2000), Pamuk e Sepil (2001), Baumgartner (2003), Hamacher e Meyer (2006), Gavriliouk e Hamacher (2006) e Campbell, Lowe e Zhang (2007).

3.1.2.7 Problemas de cobertura

Campbell (1994) foi também o primeiro a apresentar na literatura o problema conhecido como *hub covering problem*, propondo o primeiro modelo matemático de programação inteira para representar o problema.

Ernst et al. (2005) apresentaram a melhor formulação matemática, tendo sucesso na obtenção de soluções ótimas para o problema. A melhor heurística para solução do *hub covering problem* foi proposta por Calik et al. (2009) que desenvolveram uma heurística baseada em busca tabu para solução do problema, apresentando testes computacionais com dados derivados do conjunto de dados CAB, e também de uma rede da Turquia contendo 81 nós.

Outras referencias podem ser encontradas em Kara e Tansel (2003), Wagner (2004) e Hamacher e Meyer (2006).

3.2 LITERATURA RELATIVA AO PROBLEMA DE CARREGAMENTO DE CARGAS EM UMA REDE DE TRANSPORTE

O problema do carregamento de cargas em uma rede de transporte tem sido largamente estudado na literatura, e conhecido com o nome de *network loading problem* (NLP). O NLP é considerado uma variante do problema do projeto de redes capacitado com múltiplas *commodities* (*capacitated network design problem*), e foi estudado pela primeira vez por Magnanti, Mirchandani e Vachani (1993).

O NLP consiste na definição dos arcos ou ligações a serem utilizados, dentre um conjunto de ligações possíveis entre nós ou terminais de uma rede, de tal forma a transportar todas as commodities (ou cargas parceladas) desde sua origem até seu destino, a um menor custo possível. Custo esse composto exclusivamente de um valor fixo proporcional à quantidade de recursos (no caso do transporte de carga parcelada, veículos) utilizados em uma determinada ligação.

O NLP pode surgir em diversas aplicações e contextos. Um exemplo é no setor de telecomunicações, em que pode representar o projeto de redes privadas que utilizam instalações de transmissão digital (chamados circuitos T1), para enviar o tráfego de voz e dados entre diferentes locais. No setor de transporte de cargas, estas instalações podem representar veículos de tamanho fixo, e uma ligeira variação desse problema pode representar o problema conhecido como *load plan:* a designação de veículos a rotas e o carregamento das cargas nos veículos, como em Powell e Sheffi (1983) (MAGNANTI; MIRCHANDANI; VACHANI, 1993).

O NLP modela situações em que o custo variável associado aos fluxos é igual a zero, e instalações de capacidade fixa estão disponíveis para o transporte do fluxo. Magnanti, Mirchandani e Vachani (1993) estudaram o problema de configuração de uma rede de telecomunicações, consistindo em determinar o número de instalações de transmissão digital a serem abertas em cada arco da rede de tal forma a atender toda a demanda, desde sua origem até o seu destino, a um mínimo custo. Segundo os autores, o NLP é fortemente NP-*hard* e há pouca esperança de desenvolvimento de algoritmos teoricamente eficientes para sua solução.

Magnanti, Mirchandani e Vachani (1993), derivaram uma família de faces de um envoltório convexo para dois subproblemas principais do NLP, e puderam caracterizar de forma completa o envoltório convexo de soluções viáveis do problema, fortalecendo o modelo matemático de programação inteira utilizado para representar o NLP, definindo uma classe exponencial de inequações válidas, chamadas *inequações de capacidade residual*.

Segundo os autores, N corresponde ao o conjunto de nós da rede em estudo, K o conjunto de *commodities*, e A o conjunto de arcos candidatos para a definição da rede.

Para cada $k \in K$, seja R_k a quantidade de fluxo requerida para a *commodity* k a ser transportada do seu ponto de origem, definido como sendo $O(k)$, para o seu ponto de destino $D(k)$. O modelo contém dois tipos de variáveis, um conjunto de variáveis discretas representando escolha discreta (estrutural) de projeto da rede, e outro conjunto de variáveis contínuas modelando as decisões de fluxo.

x_{ij} uma variável inteira não negativa indicando se o arco $(i,j) \in A$ é utilizado e também quantos recursos serão alocados no arco (i,j) para atender toda a demanda; se o arco não for utilizado então $x_{ij} = 0$. f_{ij}^k uma variável contínua representando a quantidade da *commodity* k enviada pelo arco (i,j) para chegar ao seu destino, K a capacidade dos arcos da rede, e a_{ij} o custo unitário ao se abrir um arco (i,j) na solução.
O modelo matemático proposto por Magnanti, Mirchandani e Vachani (1993) para representar o NLP foi:

$$\text{Min} \sum_{(i,j)\in A} a_{ij}x_{ij}, \tag{3.28}$$

sujeito às restrições:

$$\sum_{\{j\in N:(i,j)\in A\}} f_{ij}^k - \sum_{\{j\in N:(j,i)\in A\}} f_{ji}^k = \begin{cases} R_k, & se\ i = O(k) \\ -R_k, & se\ i = D(k) \\ 0, & \text{caso contrario} \end{cases} \quad \forall i \in N,\ \forall k \in K, \tag{3.29}$$

$$\sum_{k\in K}(f_{ij}^k + f_{ji}^k) \leq \mathcal{K}x_{ij} \ \ \forall\,(i,j) \in A, \tag{3.30}$$

$$x_{ij} \geq 0,\ inteiro, \quad \forall (i,j) \in A, \tag{3.31}$$

$$f_{ij}^k,\ f_{ji}^k \geq 0, \ \ \forall (i,j) \in A,\ \forall k \in K. \tag{3.32}$$

A função objetivo (3.28) visa a minimizar o custo associado ao se alocar um recurso (veículo) na ligação de *i* para *j*. As restrições de conservação de fluxo para cada uma das *commodities* em cada nó são representadas pela equação (3.29). A restrição de capacidade (3.30) visa garantir que o fluxo total em um arco não pode exceder a sua capacidade.

Já Magnanti, Mirchandani e Vachani (1995) estudaram o problema do carregamento em uma rede capacitada (NLP) com dois tipos distintos de recursos a serem alocados nos arcos, nesse caso dois tipos diferentes de cabos em uma rede de telecomunicações; considerando duas abordagens para solução do problema de programação inteira mista: uma estratégia baseada em relaxação lagrangiana, e outra baseada na definição de planos de corte utilizando três tipos de inequações válidas.

Os dois tipos de recursos estudados pelos autores possuíam capacidades distintas, sendo a primeira delas considerada de baixa capacidade e a segunda de alta capacidade.

Segundo os autores, sejam os parâmetros:

- a_{ij} : custo unitário do recurso de baixa capacidade no arco $(i,j) \in A$;
- b_{ij} : custo unitário do recurso de alta capacidade no arco $(i,j) \in A$;
- $\mathcal{K}$: capacidade unitária do recurso de baixa capacidade;
- $\mathcal{C}$: capacidade unitária do recurso de alta capacidade.

Utilizando a mesma nomenclatura de parâmetros e variáveis das formulações apresentadas anteriormente, têm-se que o modelo matemático proposto por Magnanti, Mirchandani e Vachani (1995) para representar o NLP com dois tipos diferentes de recursos foi:

$$\text{Min} \sum_{(i,j)\in A} (a_{ij}x_{ij} + b_{ij}y_{ij}), \tag{3.33}$$

sujeito às restrições:

$$\sum_{\{j\in N:(i,j)\in A\}} f_{ij}^k - \sum_{\{j\in N:(j,i)\in A\}} f_{ji}^k = \begin{cases} R_k, & se\ i = O(k) \\ -R_k, & se\ i = D(k) \\ 0, & \text{caso contrario} \end{cases} \quad \forall i \in N,\ \forall k \in K, \tag{3.34}$$

$$\sum_{k\in K} (f_{ij}^k + f_{ji}^k) \leq \mathcal{K}x_{ij} + \mathcal{C}y_{ij} \ \ \forall\, (i,j) \in A, \tag{3.35}$$

$$x_{ij},\ y_{ij} \geq 0,\ inteiro, \quad \forall (i,j) \in A, \tag{3.36}$$

$$f_{ij}^k,\ f_{ji}^k \geq 0, \ \ \forall (i,j) \in A,\ \forall k \in K. \tag{3.37}$$

Esta formulação matemática visa minimizar o custo dos recursos alocados nos arcos, assumindo que é suficiente instalar somente capacidade suficiente para atender a demanda, sem a necessidade de se oferecer capacidade extra, sendo esta premissa válida também para o transporte de cargas.

A função objetivo (3.33) visa minimizar o custo para operação. As restrições de conservação de fluxo para cada uma das commodities em cada nó são representadas pela equação (3.34). A restrição de capacidade (3.35) visa garantir que o fluxo total em um arco não pode exceder a capacidade alocada no mesmo.

Magnanti, Mirchandani e Vachani (1995) mostraram que a solução obtida ao se resolver o problema de programação linear, com as inequações desenvolvidas, foi tão boa quanto a solução obtida do problema de programação inteira mista resolvido utilizando relaxação lagrangiana (medido pelo *gap* dos valores das funções objetivo), concluindo que os planos de corte desenvolvidos se mostraram robustos, podendo ser uma ferramenta eficaz para a solução de problemas de tamanho real, comuns em redes de telecomunicações.

Bienstock e Günlük (1996) estudaram o problema de expansão capacitado (*capacity expansion problem*) e estenderam dois tipos de inequações válidas apresentadas em Magnanti, Mirchandani e Vachani (1995). De um modo geral, a abordagem utilizada pelos autores se mostrou eficaz quando aplicado aos problemas teste utilizados, reduzindo consideravelmente o *gap* entre a solução do problema de programação linear e a solução inteira do problema. Dessa forma, com o *gap* reduzido, a solução inteira do problema pôde ser obtida em poucos segundos, utilizando o CPLEX. As redes testadas possuíam de 15 a 16 nós, e de 22 a 49 arcos.

Barahona (1996) trata do NLP em redes de telecomunicações e estuda o problema com e sem *bifurcação*. O autor define que o problema é *não bifurcado* se o fluxo para cada *commodity* deve seguir um único caminho até chegar ao seu destino. O problema foi resolvido utilizando-se uma relaxação baseada nos fluxos *multicommodity*, e a partir da solução do problema bifurcado, derivou também a solução do problema que não permite bifurcação dos fluxos. Para testar o método de solução proposto, Barahona (1996) utilizou dados de redes variando de 10 a 64 nós, e de 646 a 2016 *commodities*.

O modelo matemático utilizado por Barahona (1996) para representar o NLP que permite bifurcação nos fluxos corresponde ao mesmo proposto por Magnanti, Mirchandani e Vachani (1993). Utilizando a mesma nomenclatura de parâmetros e variáveis das formulações apresentadas anteriormente, o modelo matemático utilizado por Barahona (1996) para representar o NLP que não permite bifurcação dos fluxos foi:

$$\text{Min} \sum_{(i,j)\in A} a_{ij}x_{ij}, \tag{3.38}$$

sujeito às restrições:

$$\sum_{\{j\in N:(i,j)\in A\}} f_{ij}^k - \sum_{\{j\in N:(j,i)\in A\}} f_{ji}^k = \begin{cases} 1, & se\ i = O(k) \\ -1, & se\ i = D(k) \\ 0, & \text{caso contrario} \end{cases} \quad \forall i \in N,\ \forall k \in K, \tag{3.39}$$

$$\sum_{k\in K} R_k(f_{ij}^k + f_{ji}^k) \leq \mathcal{K}x_{ij} \ \ \forall\,(i,j) \in A, \tag{3.40}$$

$$x_{ij} \geq 0,\ inteiro, \quad \forall (i,j) \in A, \tag{3.41}$$

$$f_{ij}^k,\ f_{ji}^k \in \{0,1\}, \ \ \forall (i,j) \in A,\ \forall k \in K. \tag{3.42}$$

Segundo o autor, esse problema é NP-*hard* e possui a rede de Steiner como um caso especial, possuindo uma dificuldade extra ao envolver fluxos *multicommodity* inteiros.

O foco do trabalho de Mirchandani (2000) foi na solução do NLP capacitado, com dois tipos de recursos a serem alocados nos arcos, da mesma forma que em Magnanti, Mirchandani e Vachani (1995), e considerou o problema com uma e múltiplas *commodities*. O autor desenvolveu formulações matemáticas equivalentes em um espaço dimensional menor, projetando as variáveis de fluxo e estudando as propriedades do poliedro para os cones de projeção correspondentes. E ainda caracterizou algumas classes de faces definindo inequações para esse poliedro de espaço dimensional menor.

No trabalho de Berger et al. (2000) foi estudado o NLP não bifurcado, em que a demanda para qualquer nó da rede deve ser satisfeita através de um único caminho. Foi desenvolvida uma heurística baseada em busca tabu para solução do problema, cujo desempenho foi comparado com duas outras heurísticas desenvolvidas: *1-opt* e *2-opt.*

Berger et al. (2000) utilizaram um modelo matemático baseado em caminhos (*path based*), equivalente ao modelo proposto por Magnanti, Mirchandani e Vachani (1995) para dois tipos de recursos, para representar o problema não bifurcado, e ainda generalizaram o modelo para representar problemas com mais de dois tipos de recursos.

Ainda segundo Berger et al. (2000), $G = \{V, E\}$ corresponde a um grafo não direcionado onde V seja o conjunto de nós da rede com cardinalidade n, E o conjunto de arcos de cardinalidade m, e $D \subset V$ o subconjunto de nós de demanda com cardinalidade r.

Utilizando a seguinte notação:

- T : conjunto de tipos de recursos;
- P_i : conjunto de caminhos partindo do terminal central até o terminal de demanda i;
- d_i : demanda no nó i;
- c_t : custo unitário do recurso do tipo t (por unidade de comprimento);
- w_t : capacidade do recurso do tipo t;
- l_e : comprimento do arco e;
- w_e : capacidade inicial no arco e;
- δ_{ej} : igual a 1 se o arco e está no caminho j, ou 0 caso contrário,
- x_{ij} : variável binária que recebe valor igual a 1 se o caminho $j \in P_i$ é utilizado para servir a demanda do nó i, ou 0 caso contrário,
- y_{te} : variável estrutural inteira positiva igual à quantidade de recursos do tipo t utilizados no arco e.

O modelo proposto por Berger et al (2000) foi:

$$Min \quad \sum_{e \in E} l_e \left(\sum_{t \in T} c_t y_{te} \right) \tag{3.43}$$

Sujeito às restrições:

$$\sum_{i \in D} \left(\sum_{j \in P_i} x_{ij} \delta_{ej} \right) d_i \leq w_e + \sum_{t \in T} w_t y_{te}, \quad \forall e \in E, \tag{3.44}$$

$$\sum_{j \in P_i} x_{ij} = 1, \quad \forall i \in D, \tag{3.45}$$

$$x_{ij} \in 0, 1, \quad \forall i \in D, \; \forall j \in P_i, \tag{3.46}$$

$$y_{te} \geq 0, \text{inteiro}, \quad \forall t \in T, \; \forall e \in E. \tag{3.47}$$

Nesse modelo, o primeiro conjunto de restrições (3.44) visa assegurar a existência de capacidade suficiente na rede para comportar o fluxo em cada arco. O segundo conjunto de restrições (3.45) requer fluxos não bifurcados, isto é, um caminho único do nó central até cada nó de demanda. O objetivo é minimizar o custo de instalação de qualquer capacidade adicional para satisfazer a demanda em cada terminal.

Berger et al. (2000) realizaram experimentos computacionais com redes possuindo 50, 100 e 200 nós. Os resultados obtidos mostraram que a busca tabu obteve um desempenho superior no que diz respeito à qualidade da solução final obtida, mas necessitou em alguns casos oito vezes mais tempo de processamento para a obtenção das soluções, sendo um método de solução competitivo especialmente para instâncias maiores, acima de 100 nós.

Gendron, Potvin e Soriano (2002) propuseram uma heurística em duas etapas para solução do NLP não bifurcado. A primeira etapa da heurística consistiu na construção de uma solução inicial que em seguida foi melhorada, utilizando um

mecanismo de busca local e busca tabu. O modelo matemático utilizado por Gendron, Potvin e Soriano (2002) para representar o NLP não bifurcado foi o mesmo utilizado por Berger et al. (2000).

O mecanismo para geração de soluções iniciais proposto em Gendron, Potvin e Soriano (2002) foi também utilizado para reiniciar a heurística a partir de uma nova solução, visando diversificar a busca em diferentes regiões do espaço de soluções viáveis do problema. Quatro estratégias diferentes de diversificação foram testadas em problemas com 200 e 500 nós. Nenhuma comparação foi realizada para se verificar o quão distante as soluções obtidas estavam da solução ótima dos problemas testados.

Avella, Mattia e Sassano (2007) introduziram uma nova classe de inequações métricas justas para o NLP, que caracterizaram completamente o envoltório convexo de soluções inteiras viáveis para o problema, apresentando algoritmos de separação para as inequações métricas justas e algoritmos de plano de corte.

Os experimentos computacionais realizados por Avella, Mattia e Sassano (2007) mostraram que, para problemas de pequeno porte, com redes de até 64 nós, os autores puderam obter a solução ótima em 22 dos 35 problemas resolvidos.

Bartolini e Mingozzi (2009) estudaram a versão não bifurcada do NLP incorporando na função objetivo os custos relativos aos fluxos das *commodities.* Nesse trabalho os autores desenvolveram um método exato e quatro heurísticas diferentes para solução do problema. A primeira heurística consistiu em gerar uma solução inicial viável, que em seguida era melhorada até que uma condição necessária para otimalidade fosse satisfeita. Duas outras heurísticas foram baseadas na enumeração parcial do espaço de soluções, e a última heurística consistiu em aplicar busca tabu em diferentes soluções iniciais viáveis.

Os experimentos realizados por Bartolini e Mingozzi (2009) foram realizados em duas classes distintas de problemas. A primeira delas, chamada pelos autores de classe “A” consistiu na solução de cinco problemas diferentes, tendo de 70 a 93 *commodities*, de 18 a 29 nós, e de 29 a 61 arcos. A segunda classe de problemas

(classe “B”), consistiu na solução de um único problema com 435 *commodities*, 30 nós e 435 arcos.

Os resultados obtidos pelos autores evidenciaram a eficácia das heurísticas desenvolvidas especialmente no problema da classe “B”, de maior porte, tendo um resultado entre 14 e 15% superior ao se comparar com duas horas de processamento do CPLEX (versão 10.1). Já nos problemas da classe “A”, o CPLEX obteve resultados 1 a 3 % melhores que as heurísticas propostas, com uma hora de processamento.

No trabalho de Babonneau e Vial (2010) os autores utilizam uma abordagem inovadora para solução do NLP. O método de solução desenvolvido pelos mesmos pode ser considerado uma versão modificada do algoritmo de decomposição de Benders. O problema mestre consistiu em um problema de programação inteira puro, e o subproblema em um problema de otimização contínua responsável por testar se as capacidades inteiras geradas pelo problema mestre foram suficientes para suportar o fluxo para atender a demanda.

A diferença fundamental no algoritmo desenvolvido por esses autores, quando comparado com o esquema clássico de particionamento de Benders, foi de que o problema mestre não consistiu em buscar a melhor solução inteira do problema relaxado. Em vez disso, a busca foi por uma solução inteira melhor, próxima à melhor solução inteira gerada ao longo da execução do algoritmo, e o subproblema se resumiu a um problema de fluxo *multicommodity* não linear que foi resolvido utilizando um pacote de otimização de equações não diferenciáveis baseado em um método conhecido como ACCPM (*Analytic Center Cutting Plane Method).*

Os experimentos computacionais realizados mostraram que a abordagem utilizada pelos autores para solução do NLP se mostrou eficiente, especialmente para problemas de maior porte, com mais de 700 *commodities*, superando todos os resultados obtidos por outros autores para esta classe de problemas.

Altin, Yaman e Pinar (2010) estudaram o NLP em que as demandas entre os pares de nós da rede são assumidas como não sendo conhecidas de antemão. Os autores

propuseram uma nova formulação matemática mais compacta para o problema, e após um estudo de poliedros baseados na projeção das variáveis de fluxo do problema, resolveram o NLP utilizando *branch-and-cut* (B&C). Os experimentos computacionais realizados mostraram que o B&C foi eficaz na solução de problemas com até dois tipos de diferentes de recursos disponíveis para serem alocados nos arcos.

3.2.1 O problema de projeto de uma rede de serviços (PPRS)

O problema de roteamento de cargas em uma rede de transporte está fortemente ligado ao problema de projeto de uma rede de serviços (ou *service network design problem*). O projeto de uma rede de serviços consiste em decidir as características (freqüência, número de paradas intermediárias etc.) das rotas a serem operadas, a designação do tráfego ao longo destas rotas, as regras de operação de cada terminal, e possivelmente a realocação de veículos vazios e contêineres (CRAINIC, 2000).

O PPRS é considerado mais abrangente que o NLP, por levar em conta outros fatores, sendo o foco principal a definição da freqüência dos serviços a serem oferecidos (CRAINIC, 2000), sendo que o NLP está focado principalmente no atendimento das demandas por transporte e na definição do fluxo que cada carga deve percorrer para chegar até o seu destino, devendo para tal selecionar os arcos a serem abertos.

Crainic (2000) trata especificamente do problema de projeto da rede de serviços, apresentando uma revisão do estado-da-arte dos esforços para a modelagem matemática e solução do PPRS. É apresentada também uma nova classificação para o problema, incluindo os modelos matemáticos com suas diversas variantes.

O PPRS é um problema *NP-hard* sendo geralmente representado como um problema de programação inteira mista de otimização em uma rede capacitada com múltiplas *commodities*. Os modelos matemáticos usualmente utilizados para representar o PPRS possuem uma relaxação muito fraca, ou melhor, a solução obtida resolvendo-se o problema relaxando-se as restrições de integralidade das

variáveis possui uma grande quantidade de valores fracionários e muito distante da solução inteira do problema, dificultando assim todos os métodos de solução baseados em *branch-and-bound*. Dessa forma, a grande maioria dos métodos de solução propostos são baseados em heurísticas, como por exemplo, em Armacost, Barnhart e Ware (2002), Barnhart, Jin e Vance (2000), Barnhart e Schneur (1996), Kim et al. (1999), Crainic, Ferland e Rousseau (1984), Crainic e Rousseau (1986), Crainic e Roy (1988), Grunert e Sebastian (2000), Powell e Sheffi (1983), Powell (1986) e Powell e Sheffi (1989).

No trabalho de Powell e Sheffi (1983) é proposto um modelo matemático para representar o PPRS como um problema de projeto de redes com custo fixo, com restrições de nível de serviço representadas heuristicamente através de um conjunto de freqüências mínimas nos arcos. Foi também apresentada uma heurística para solução do PPRS consistindo basicamente na inclusão e exclusão de arcos da rede de forma ordenada, juntamente com uma heurística de busca local para melhoria.

Crainic, Ferland e Rousseau (1984) examinaram o problema de roteamento do tráfego de cargas, programação dos serviços de transporte, e na alocação de tarefas para pátios de classificação em uma rede de transporte ferroviário. Foi descrito um modelo geral de otimização que levou em conta as interações entre estas atividades, e desenvolveram estratégias globais para seu gerenciamento em um horizonte de planejamento de médio prazo, propondo também uma heurística para solução.

Em Crainic e Rousseau (1986) foi estudado o problema do transporte de cargas que ocorre quando a mesma autoridade controla e planeja tanto o fornecimento de serviços de transporte, como também o roteamento da carga. É proposta uma estratégia para solução do problema baseada nos princípios de decomposição do problema e geração de colunas.

Powell (1986) utiliza um processo de melhorias locais para solução do PPRS, consistindo na inclusão e exclusão de arcos na solução. O nome “melhoria local” vem do fato de que as alterações da rede foram analisadas item a item, ou seja, foi analisada individualmente a contribuição que um determinado arco possa trazer, caso este venha a ser cancelado ou inserido da rede.

Crainic e Roy (1988) desenvolveram uma plataforma genérica para modelagem do problema de planejamento tático do transporte de cargas, utilizando técnicas de programação matemática. A abordagem desses autores consistiu na geração, avaliação e seleção de estratégias de operação considerando globalmente a rede de serviços de uma transportadora, e o roteamento das cargas através desta rede.

Em Powell e Sheffi (1989) foi descrito o projeto e a implementação de um sistema interativo de otimização para o roteamento das cargas a serem transportadas em uma rede de transporte de carga parcelada. Os autores propuseram uma estratégia interativa para solução do problema, que foi decomposto em problemas individuais mais simples de serem resolvidos, sendo aplicada com sucesso em uma grande empresa de transporte de carga parcelada dos Estados Unidos.

Barnhart e Schneur (1996) propõem a utilização do método de geração de colunas para solução do PPRS para o problema do transporte de cargas expressas pelo modal aéreo, sendo capazes de encontrar soluções aproximadas para uma frota fixa de aeronaves ou para uma frota de tamanho não especificado.

O foco da pesquisa apresentada em Kim et al. (1999) foi na modelagem e solução do PPRS em larga escala, envolvendo o transporte de cargas expressas. O objetivo foi definir a movimentação de encomendas das suas origens até os seus destinos, em uma dada janela de atendimento, muitas vezes curta, limitado pelas capacidades de classificação nos terminais intermediários, e um número finito de aeronaves e caminhões para entrega. Em síntese, os autores propuseram um método de solução baseado na técnica de geração de colunas e linhas para a obtenção de soluções de um problema de grande porte do mundo real.

Barnhart, Jin e Vance (2000) formularam o problema de blocagem de vagões ferroviários como um NDP, com restrições de fluxo e de grau máximo nos nós, e propuseram uma heurística baseada em relaxação lagrangiana para solução, decompondo o problema de programação linear inteira mista em dois subproblemas mais simples de serem resolvidos.

Grunert e Sebastian (2000) apresentaram modelos de planejamento para operações de transporte de longo curso para empresas de transporte de carga expressa, para os diferentes problemas, nos diferentes níveis de decisão, que surgem nesse segmento, tratando também do PPRS. É apresentada a forma com que esses modelos foram construídos e como estes se relacionam com outros modelos freqüentemente apresentados na literatura.

Armacost, Barnhart e Ware (2002) apresentaram uma nova abordagem para solução do PPRS para o serviço de cargas expressas, e desenvolveram uma nova relaxação para o modelo matemático, e com esse modelo puderam resolver o PPRS utilizando os dados fornecidos por uma empresa de transporte de encomendas expressas dos Estados Unidos, a UPS, demonstrando um ganho potencial na redução de custos de transporte.

Inspirado no trabalho de Powell (1986), Hellmuth (2004) utilizou um método interativo para resolver o problema de planejamento da estrutura operacional de uma empresa de transporte de carga parcelada, envolvendo a solução do PPRS, juntamente com o reposicionamento de veículos vazios decorrentes do desbalanceamento entre o total de veículos expedidos e recebidos nos terminais.

Wieberneit (2008) faz um amplo levantamento dos trabalhos realizados sobre o PPRS, abordando as diferenças práticas para os diferentes níveis de planejamento estudados, apresentando ainda uma formulação matemática genérica para o problema. Esta representação matemática genérica para o problema é comparada com cinco problemas de planejamento tático encontrados na literatura:

1. o problema de transporte de carga expressa;
2. o problema de configuração de uma rede de transporte aéreo de encomendas postais;
3. a operação de transporte de carga parcelada na America do Norte;
4. a operação de transporte de carga parcelada na Europa;
5. a operação de transporte multimodal de carga parcelada na Europa.

A decomposição do PPRS em um problema de designação e outro de programação de veículos a rotas foi recentemente estudada por Teypaz (2010). O problema

original foi dividido em três etapas principais: construção da rede, roteamento das cargas e criação da programação dos veículos. Cada etapa do problema foi resolvida utilizando heurísticas, programação inteira mista, e técnicas de programação restrita.

3.2.2 Reposicionamento de veículos vazios

Uma característica comum de qualquer sistema de transporte de carga parcelada é a necessidade de movimentação de veículos vazios devido ao desbalanceamento que existe nos fluxos de negócios, o qual resulta em discrepâncias entre o fornecimento e a demanda por veículos nos terminais do sistema. Para corrigir estas diferenças, veículos precisam ser reposicionados, de tal forma que estejam prontos para atender a demanda em períodos futuros.

Em um nível um pouco mais agregado, o balanceamento de veículos vazios faz parte também do projeto da rede de serviços, contudo a decisão da quantidade e localização de veículos a serem enviados é muito complicada pela numerosa quantidade de possibilidades de movimento e as incertezas futuras de demanda e fornecimento.

A busca por uma estratégia mais econômica é um problema por si só muito significante, possuindo modelagens matemáticas e estratégias de solução específicas. Devido às estas especificidades, esta pesquisa não contempla o estudo do reposicionamento de veículos vazios. Uma descrição mais detalhada desse problema pode ser vista nos trabalhos de Dejax e Crainic (1987), Powell, Jaillet e Odoni (1995), Cordeau, Toth e Vigo (1998) e Crainic (1998), entre outros.

3.3 CONCLUSÕES DO CAPÍTULO

Este capítulo tratou da revisão bibliográfica relativa ao projeto de redes de transporte de carga parcelada, analisando dois problemas comuns encontrados neste setor, em diferentes níveis de decisão: estratégico e tático.

Em nível de decisão estratégico, o problema consiste em: dado um conjunto de terminais de uma empresa transportadora de carga parcelada, o objetivo é definir dentre esse conjunto, quantos e quais desses terminais operarão como terminais de consolidação, ou *hubs*, e ainda na definição da alocação dos terminais não-*hub* aos *hubs* localizados.

Em nível de decisão tático, o problema de carregamento de carga parcelada consiste em, a partir de uma rede *hub-and-spoke* criada, definir as rotas a serem operadas para o transporte de cargas parceladas entre terminais, as possíveis paradas intermediárias em terminais de consolidação, e também no dimensionamento dos veículos envolvidos para a operação.

O problema estratégico de configuração de uma rede do tipo *hub-and-spoke* foi modelado como o problema conhecido na literatura por *uncapacitated single allocation hub location problem* (USAHLP), ou problema de localização de *hubs* não capacitado com alocação única. Nesse problema, a quantidade e localização de terminais de consolidação devem ser determinadas, simultaneamente com a alocação dos terminais não-*hub* aos *hubs* localizados. É apropriado um custo fixo ao se definir um terminal como um ponto para consolidação de cargas, e não há restrição de capacidade no volume de carga movimentado nos *hubs*.

Apesar de o USAHLP ter sido relativamente bem estudado na literatura, o problema ainda carece de um método de solução exato eficiente, e também de uma heurística com desempenho superior aos já estudados na literatura, pois os problemas do mundo real muitas vezes são de tamanho e complexidade superiores aos já estudados.

Até o presente momento, todos os autores deduziram a solução “ótima” do USAHLP tendo como ponto de partida a solução ótima do USApHMP apresentada em Skorin-Kapov,D., Skorin-Kapov, J, O'Kelly (1996). Como exemplo os trabalhos de Abdinnour-Helm e Venkataramanan (1998), Abdinnour-Helm (1998) e Topcuoglu et al. (2005).

A premissa nesses trabalhos foi que a quantidade de *hubs* obtida com os métodos de solução propostos estava correta, e correspondia com a quantidade ótima de *hubs.* Com isso, alguns enganos foram cometidos, ao se comparar o resultado final obtido com as heurísticas desenvolvidas com a solução dita “ótima” para esses problemas.

Deve-se notar ainda que os trabalhos apresentados na literatura consideram os mesmos conjuntos de dados: o conjunto de dados CAB possuindo uma quantidade máxima de 25 terminais na rede, e o conjunto de dados AP com até 200 terminais, e somente um único trabalho apresentou um estudo de caso analisando o problema real de uma empresa transportadora de carga parcelada, que foi em Cunha e Silva (2007).

Já em nível de decisão tático, o problema do carregamento de carga parcelada em uma rede de transporte foi modelado como um problema denominado na literatura de *network loading problem* (NLP). As raras aplicações práticas apresentadas na literatura para o NLP são todas relacionadas ao projeto de redes de telecomunicações, ou tratam o problema somente do ponto de vista matemático, sem apresentar uma aplicação prática para o mesmo.

O NLP é um problema *NP-hard* e todos os trabalhos apresentados na literatura, que se propuseram a tentar resolver problemas de grande porte desenvolveram uma estratégia para solução baseada em heurísticas, destacando a busca tabu. Para avaliar a qualidade das soluções obtidas com as heurísticas, diversos autores, como por exemplo, Bartolini e Mingozzi (2009), utilizaram o pacote de otimização CPLEX, fixando um tempo limite para processamento, e comparando a solução obtida do pacote com a solução obtida pela heurística, no mesmo tempo de processamento.

4 PROJETO DE REDES *HUB-AND-SPOKE* – MODELAGEM E HEURÍSTICAS

No projeto de redes *hub-and-spoke,* diferentes aspectos podem ser incluídos na definição do problema, especialmente no que diz respeito a:

1. quantidade de *hubs* a serem localizados;
2. alocação dos nós aos *hubs;*
3. capacidade dos *hubs.*

Nesta pesquisa é tratado o caso especial em que os *hubs* não possuem restrição de capacidade no fluxo a ser movimentado; a quantidade *p* de *hubs* a serem localizados deve ser definida, não sendo um dado de entrada para o problema; e os nós não-*hub* devem ser alocados a um único *hub*. Esse problema é conhecido na literatura como o problema de localização de *hubs* não capacitado com alocação única (*Uncapacitated Single Allocation Hub Location Problem* ou USAHLP).

Dada uma rede de transporte de carga parcelada composta por nós ligados entre si, o USAHLP consiste na definição simultânea da quantidade e localização de nós que operarão como ponto para consolidação de cargas (ou *hubs*), e na definição da alocação ou designação dos nós não-*hub* aos *hubs* localizados. Os nós de consolidação não possuem restrição de capacidade, ou seja, não há limite na quantidade de carga classificada nos *hubs*, e são apropriados custos fixos ao se selecionar um nó da rede como um *hub.* Adicionalmente, há economias de escala nos custos unitários de transporte nas ligações entre *hubs* em decorrência do maior fluxo consolidado.

Neste capítulo é proposto um novo modelo matemático para representar o USAHLP, e são descritas quatro heurísticas para sua solução. São apresentados ainda os resultados obtidos com os experimentos computacionais realizados utilizando dados comumente utilizados na literatura para *benchmark*, e também com novas instâncias geradas compostas por redes de até 400 nós, sendo a maior rede utilizada até o momento para solução do USAHLP.

4.1 MODELO MATEMÁTICO PARA O PROJETO DE REDES *HUB-AND-SPOKE*

Conforme visto no capítulo anterior, mais especificamente na Seção 3.1, o problema mais geral de configuração de redes *hub-and-spoke* foi bastante estudado na literatura, em suas diferentes variações, em especial o USAHLP, em que o número de *hubs* não é pré-definido, e o USApHMP, em que a quantidade *p* de *hubs* a serem localizados é dada de antemão.

Tendo como ponto de partida a formulação matemática desenvolvida por Ernst e Krishnamoorthy (1999) para o CSAHLP (isto é, o USAHLP com restrições de capacidade no fluxo movimentado nos *hubs*), é proposto neste trabalho um modelo matemático similar para representar o USAHLP. Esta formulação possui $(n^3 + n^2)$ variáveis, sendo destas n^2 binárias, representando um grande avanço ao se comparar a formulação matemática utilizada em O'Kelly (1992) com uma função objetivo quadrática, Campbell (1996) e Skorin-Kapov, D., Skorin-Kapov, J. e O'Kelly (1996) com um modelo matemático linear, porém com $(n^4 + n^2)$ variáveis. O modelo matemático aqui proposto reduz tanto a quantidade de variáveis como de restrições por um fator de aproximadamente n.

Sejam os parâmetros:

- α : fator de desconto na ligação entre *hubs*;
- χ : custo por unidade de fluxo na operação de coleta (no transporte nó de origem $\rightarrow$ *hub*);
- δ : custo por unidade de fluxo na operação de distribuição (no transporte *hub* $\rightarrow$ nó de destino);
- W_{ij} : quantidade de carga com origem no nó i e destino no nó j;
- $O_i = \sum_j W_{ij}$ a somatória dos fluxos originados no nó i;
- $D_i = \sum_j W_{ji}$ a somatória dos fluxos destinados ao nó i;
- C_{ij} o custo por unidade de fluxo entre os nós i e j;
- F_k custo fixo ao se estabelecer um *hub* no nó k.

e variáveis de decisão:

- Y_{kl}^{i} : quantidade de carga com origem no nó i (isto é, o tráfego partindo do nó i), que é enviado via *hubs* k e l;
- X_{ik} : variável binária que recebe o valor 1 se o nó i é alocado ao *hub* localizado no nó k, ou 0 caso contrário.
- $X_{kk} = 1$: se o nó *k* é definido como *hub*, ou 0 caso contrário.

O modelo matemático utilizado para representar o USAHLP foi:

$$Min \sum_{i} \sum_{k} C_{ik} X_{ik} (\chi O_i + \delta D_i) + \sum_{i} \sum_{k} \sum_{l} \alpha C_{kl} Y_{kl}^{i} + \sum_{k} F_k X_{kk} \quad (4.1)$$

Sujeito às restrições

$$\sum_{k} X_{ik} = 1, \ \ \forall i \in N, \quad (4.2)$$

$$X_{ik} \leq X_{kk}, \ \ \forall i, k \in N, \quad (4.3)$$

$$\sum_{l} Y_{kl}^{i} - \sum_{l} Y_{lk}^{i} = O_i X_{ik} - \sum_{j} W_{ij} X_{jk}, \ \ \forall i, k \in N, \quad (4.4)$$

$$X_{ik} \in \{0, 1\}, \ \ \forall i, k \in N, \quad (4.5)$$

$$Y_{kl}^{i} \geq 0, \ \ \forall i, k, l \in N. \quad (4.6)$$

Nesse modelo matemático, a função objetivo (4.1) visa minimizar os custos de coleta, transferência e distribuição, e os custos fixos associados à abertura de *hubs.* As restrições (4.2) impõem que cada nó *i* seja designado a exatamente um único *hub*, enquanto as restrições (4.3) asseguram que um nó seja alocado a um *hub k* somente se esse for um *hub*. As restrições (4.4) representam as equações de divergência para cada *commodity i* no nó *k* em um grafo completo, em que a oferta e a demanda nos nós são determinadas pelas variáveis de alocação X_{ik}. Em outras palavras, para um dado nó qualquer *i*, seja $Y_{kl}^{i} > 0$ ou $Y_{lk}^{i} > 0$, mas não ambos simultaneamente (para $k \neq l$) dado que cada nó *i* é designado a somente um *hub k*, como imposto pelas restrições (4.2).

Mais especificamente, a variável Y_{kl}^{i} não permite mais o acompanhamento do fluxo entre pares de nós separadamente. Esta abordagem não é intuitiva, e algumas explicações adicionais são apropriadas, dado que esta questão não é explicada em detalhes em Ernst e Krishnamoorthy (1999).

Deve-se destacar que nenhum dos trabalhos encontrados na literatura considerou esta formulação para o USAHLP, e sim a sua equivalente quadrática proposta por O'Kelly (1992). Esse modelo matemático nunca foi utilizado previamente para representar o USAHLP, outro ponto importante a ser destacado é que todas as soluções ótimas reportadas na literatura basearam-se na solução do USApHMP (*Uncapacitated Single Allocation p-Hub Median Problem*), em que o número de *hubs* é fixo, variando-se o número de *hubs p.*

4.2 ESTRATÉGIA PROPOSTA PARA SOLUÇÃO DO USAHLP

Nesta Seção são descritas duas abordagens principais para solução do USAHLP. A primeira delas é baseada na obtenção de soluções ótimas para o problema, e a segunda abordagem foi centrada no desenvolvimento de heurísticas, especialmente para problemas de grande porte (redes com mais de 100 nós). A primeira heurística utilizou a técnica de *multi-start*, ou multi-início, com procedimentos de melhoria utilizando busca tabu. A segunda heurística desenvolvida utilizou exclusivamente busca tabu para obtenção das soluções.

4.2.1 Heurísticas multi-início para solução do USAHLP

Em heurísticas multi-início, um número de diferentes soluções iniciais são geradas e em seguida melhoradas utilizando-se algum método de busca local. Heurísticas multi-início podem ser aplicadas a problemas em que soluções iniciais são facilmente construídas, mas heurísticas de melhoria não são capazes de explorar a vizinhança de soluções de uma maneira eficiente com o objetivo de se alcançar melhores soluções, dado que podem ser necessários vários movimentos, muitas vezes complexos, para poder explorar de forma razoável esta vizinhança de soluções dentro do espaço de soluções viáveis do problema (MARTÍ, 2002).

Algumas vezes o problema é altamente restrito, tornando-se difícil definir vizinhanças que mantenham a viabilidade; por outro lado, existem problemas que possuem restrições fracas, resultando em um espaço de soluções e vizinhança muito grandes para serem exploradas de maneira eficiente.

Uma alternativa para superar estas dificuldades, tentando evitar um ótimo local e também alcançar diversificação, é reiniciar a busca partindo de uma nova solução, depois que a vizinhança da solução corrente tenha sido extensivamente explorada.

Ainda, estratégias multi-início podem ser utilizadas para guiar a construção de novas soluções em um horizonte de longo prazo do processo de busca. O mecanismo de reinício pode ser sobreposto com muitos métodos de busca diferentes.

Em geral, o método multi-início apresenta duas fases básicas: a primeira no qual a solução é gerada, e a segunda em que a solução é tipicamente melhorada. Desta forma, cada iteração completa produz uma solução, e a heurística termina quando o critério de parada foi alcançado. A melhor solução encontrada em todas as iterações é a saída da heurística (MARTÍ, 2002).

A heurística multi-início combina uma estratégia de diversificação dada pela fase de construção com a intensificação fornecida pela fase de melhoria, resultando em um método simples e apropriado para resolver problemas difíceis de otimização (MARTÍ e VEGA, 2003).

A estratégia multi-início foi utilizada para definir a quantidade e localização de *hubs* (problema locacional), enquanto que a busca tabu foi utilizada para a definição da designação dos nós aos *hubs* localizados (problema alocacional). A motivação para a utilização da estratégia multi-início foi devido ao fato de se verificar que soluções viáveis para a parte locacional do problema, que consiste na definição da quantidade e localização de *hubs*, podem ser geradas facilmente, sem a necessidade de estratégias complexas, como as já utilizadas na literatura baseadas basicamente em algoritmos genéticos, consumindo um grande tempo de processamento que, conforme os resultados obtidos apresentados nas seções a seguir, se mostraram

desnecessários para se resolver de forma eficiente a primeira parte da solução do problema.

Como no USAHLP os *hubs* não possuem restrição de capacidade, e qualquer nó da rede é um candidato a se tornar um *hub*, nós que possuem um grande fluxo *inbound* e *outbound* e que estão com uma localização relativa boa, ao se comparar com os outros nós da rede, possuem uma maior chance de serem escolhidos para operarem como *hubs*.

Muitos autores na literatura geraram soluções iniciais para a parte alocacional do problema, que define a alocação dos nós não-*hub* aos *hubs* localizados, utilizando a estratégia de alocação do nó ao *hub* mais próximo introduzida por O'Kelly (1987). Contudo, a alocação exclusivamente com base na distância nem sempre resulta na alocação ótima para o problema por não levar em conta o fator de desconto na ligação entre *hubs* (α). A busca tabu desenvolvida neste trabalho para resolver a parte alocacional do problema, parte de uma solução inicial gerada utilizando uma estratégia de alocação dos nós ao *hub* mais próximo. Esta solução inicial é então melhorada, caso possível, utilizando a busca tabu.

Na Figura 4.1 é apresentado um exemplo de rede *hub-and-spoke* que corresponde à solução do problema do conjunto de dados do CAB contendo 15 nós, fator de desconto (α) na ligação entre *hubs* igual a 0,2, e custo fixo igual a 100. A solução ótima para esse problema possui 5 *hubs* localizados nos nós 3, 4, 7, 12 e 14 e estão sinalizados na cor verde na figura. A ligação entre *hubs* está representada por arcos na cor vermelha. Para esse exemplo, se a alocação for realizada baseando-se exclusivamente na distância, ou seja, um nó é alocado ao *hub* mais próximo, o custo da solução é de 1.183,12. Se for alterada a alocação do nó 1 do *hub* 14, que é o *hub* mais próximo de 1, para o *hub* 4, a solução final obtida, que é a solução ótima para o problema, possui um custo de 1.030,07.

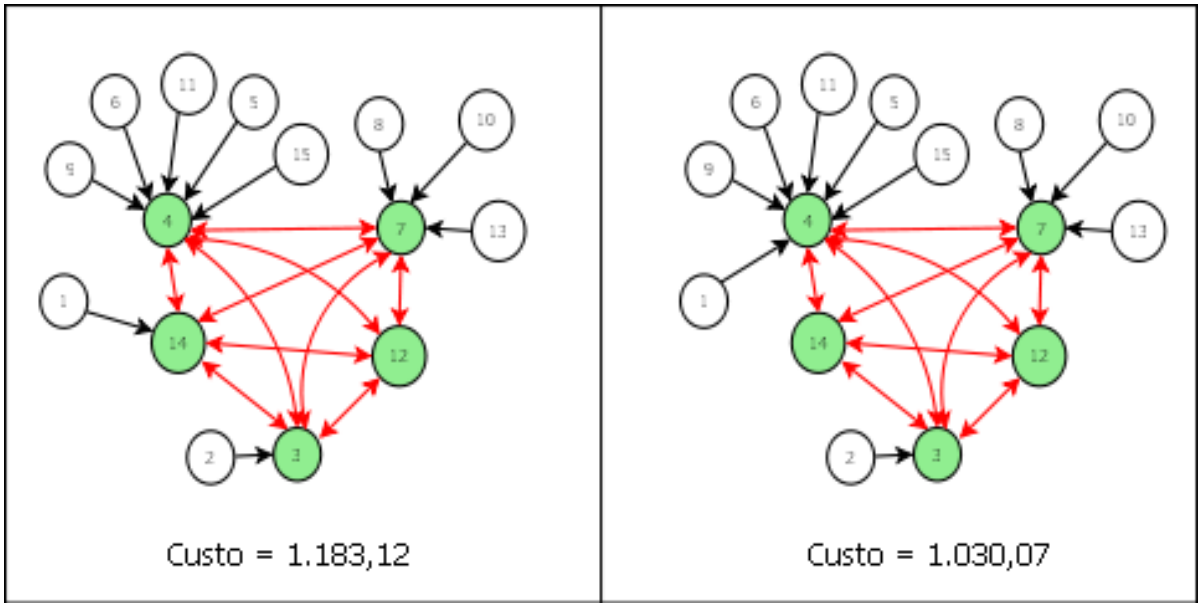

Figura 4.1: Exemplo de alocação dos nós aos *hubs*

São propostas três variantes da heurística multi-início, chamadas MSTS-1, MSTS-2 e MSTS-3, para resolver o USAHLP. (MSTS: *multi-start tabu search*). Estas três heurísticas seguem o mesmo esqueleto básico apresentado na Figura 4.2.

Heurística *MSTS*
Início
  Enquanto o número total de sementes não for alcançado **Faça**
    Enquanto o número de soluções geradas para cada semente não for alcançado **Faça**
      Gere uma solução inicial aleatória em termos dos selecionados
      Faça a alocação dos terminais não-*hub* ao *hub* mais próximo
      Execute a BT para tentar melhorar a alocação dos terminais aos *hubs*
      Se a solução explorada supera a melhor solução corrente **Então**
        Atualize a melhor solução
    Fim enquanto
  Fim enquanto
Fim

Figura 4.2: Esqueleto básico das heurísticas MSTS para o USAHLP

As principais entradas para as heurísticas são:

- O número total de sementes, ou rodadas independentes, a serem realizadas para cada problema. Cada semente é um número inteiro utilizado como entrada para o gerador de números aleatórios.

- O número total de soluções a serem exploradas a cada rodada independente (ou semente). Para cada solução são geradas aleatoriamente a quantidade e localização dos *hubs* de acordo com a probabilidade de cada nó se tornar um *hub*.

A principal diferença entre as três heurísticas propostas é no método para obtenção de uma solução inicial viável, como detalhado a seguir na Seção 4.2.1.1. Para cada solução inicial gerada é então aplicada uma heurística de melhoria baseada em busca tabu, com o objetivo de se melhorar cada nova solução gerada. Os detalhes desta heurística de melhoria, baseada em busca tabu, são apresentados mais adiante na Seção 4.2.1.2.

Todas as três variantes propostas derivam-se do fato de que uma vez que a quantidade e localização de *hubs* estão ambas definidas, uma boa solução em termos da alocação dos nós não-*hub* aos *hubs,* pode ser encontrada utilizando o critério de alocação ao *hub* mais próximo proposta pela primeira vez em O'Kelly (1987).

4.2.1.1 Geração de soluções iniciais

O conjunto de soluções iniciais é um fator importante que interfere no desempenho da heurística multi-início. Esse conjunto deve conter soluções de boa qualidade e ao mesmo tempo diversas, procurando levar a busca o mais próximo possível do ótimo global do problema. Uma vantagem da heurística proposta, e também das características do problema, é que soluções iniciais viáveis podem ser facilmente obtidas, com tempos de processamento muito reduzidos.

Para o MSTS-1, um conjunto de soluções iniciais é gerado baseado em uma probabilidade p fixa de um nó se tornar *hub*. Em outras palavras, a probabilidade p é a mesma para todos os nós da rede, e esse valor é um parâmetro de entrada para a heurística. Para cada solução gerada, o conjunto de *hubs* selecionados é dado como segue: para cada nó $i \in N$, um número r_i entre 0 e 1 é aleatoriamente

gerado. Se $r_i \leq p$, o nó i é selecionado como *hub*; caso contrário esse nó não se torna um *hub*.

No MSTS-2, a probabilidade de cada nó se tornar um *hub* não é a mesma para todos os nós, mas depende do fluxo total originado e destinado ao nó i. A hipótese é a de que nós com um grande tráfego enviado e recebido possuem uma maior probabilidade de se tornarem *hubs*, ou seja, a probabilidade de um nó *i* ser escolhido como *hub* é proporcional à parcela de fluxo total movimentado no nó em relação ao fluxo total. Usando os parâmetros já definidos, têm-se:

$$p(i) = \frac{(O_i + D_i)}{\sum_i (O_i + D_i)}\ , \quad \forall\ i \in N \tag{4.7}$$

De forma análoga ao MSTS-1, um número r_i no intervalo $[0, 1]$ é aleatoriamente gerado e se $r_i \leq p(i)$, o nó i é selecionado como *hub*.

Finalmente, no caso do MSTS-3 a probabilidade $p(i)$ do nó i se tornar um *hub* depende não só do tráfego total originado e destinado ao nó i, mas também é afetado por um fator de penalidade, baseado na localização geográfica relativa (ou espacial) do nó i, medida em termos da distância de todos os outros nós, como mostrado abaixo:

$$p(i) = \frac{(O_i + D_i)}{\sum_i (O_i + D_i)} \times \left(1 - \frac{\aleph_i}{\sum_i \aleph_i}\right) \quad \forall\ i \in N \tag{4.8}$$

onde $\aleph_i = \sum_j (d_{ij} + d_{ji})$

Essa penalidade baseada na distância objetiva reduzir a probabilidade $p(i)$, levando-se em conta a localização relativa do nó i em relação aos nós restantes. Em outras palavras, um nó *i*, mesmo que gere e atraia muita carga, tem o seu potencial para se tornar um *hub* diminuído caso sua posição relativa aos demais nós não seja tão favorável. Foi assegurado também que, nesse caso, $p(i)$ não seja inferior a um dado $p(i)_{min}$, expressado como uma fração da probabilidade do nó i se tornar um *hub,* considerada no MSTS-2, e dada pela seguinte expressão:

$$p(i)_{min} = \frac{(O_i + D_i)}{\sum_i (O_i + D_i)} \times \varphi \qquad (4.9)$$

onde φ é uma constante não negativa assumindo valor entre 0 e 1.

Dado que as três variantes da heurística multi-início propostas são baseadas em números gerados aleatoriamente, com o objetivo de se assegurar a diversidade das soluções, decidiu-se substituir a função para geração de números aleatórios fornecida pela linguagem de programação C (*rand ()*), por outra função chamada *Mersenne Twister* (MT) (Matsumoto e Nishimura, 1998).

Experimentos preliminares realizados mostraram que o MT proporciona melhores soluções utilizando um menor tempo de processamento quando comparado com a função *rand ()* embutida na linguagem de programação C. E também, a utilização do MT possibilita que as heurísticas apresentem o mesmo desempenho em diferentes plataformas, compiladores e sistemas operacionais (por exemplo Windows, Linux, etc.), diferente da função *rand ()*, que apresentou diferentes resultados em compiladores Windows e Linux.

4.2.1.2 Uma heurística de melhoria baseada em busca tabu

Nas três variantes da heurística MSTS desenvolvidas, cada solução inicial distinta gerada foi melhorada por um procedimento de busca local baseado em busca tabu. A busca tabu é um procedimento de busca local que utiliza estruturas de memória para guiar os movimentos de uma solução viável para outra, com o objetivo de se explorar regiões do espaço de busca que poderiam não ser explorados, tentando escapar do ótimo local. Os princípios fundamentais da busca tabu são apresentados em detalhes em Glover (1986, 1989).

Foi implementado um procedimento simples, porém eficaz, de movimento para buscar na vizinhança de uma dada solução para o USAHLP: um movimento chamado *shift-move* (ABDINNOUR-HELM, 1998), consistindo na troca da

designação de um único nó não-*hub* de um *hub* para outro. E ainda, a heurística de melhoria busca tabu proposta utilizou somente memória de curto prazo, e consistiu em restringir os nós candidatos para troca de alocação.

Experimentos computacionais realizados utilizando-se também o mecanismo de memória de longo prazo da busca tabu não apresentou melhorias significativas nas soluções finais obtidas, aumentando significativamente o tempo de processamento, sendo ainda um mecanismo difícil de ser calibrado. A implementação realizada consistiu em gravar as alocações realizadas e utilizar esta informação para a geração de soluções diferentes das já exploradas,

Verificou-se que apenas com estruturas e movimentos de curto prazo foi possível convergir para alocações ótimas, em tempos de processamento muito mais reduzidos, de forma compatível com os requerimentos de uma heurística multi-início, em que um número elevado de soluções são geradas, e posteriormente melhoradas, em termos da alocação dos nós não-*hub* aos *hubs*.

O foco desta busca em vizinhança na parte alocacional do problema, teve como objetivo superar as deficiências da alocação baseada na distância (O'KELLY, 1987) quando as soluções iniciais são geradas; esta deficiência é devido ao fato de que a alocação dos nós não-*hub* baseada exclusivamente na distância, não leva em consideração as economias de escala no tráfego entre *hubs*. Esta abordagem se mostrou um procedimento de melhoria simples, eficaz e fácil de ser implementado, em conjunto com a heurística multi-início*,* permitindo um grande número global de iterações, utilizando um curto tempo de processamento.

Alternativamente, poderiam ter sido consideradas outras opções de projeto de heurística multi-início, como por exemplo, uma rotina de busca local mais complexa, que poderia melhorar significativamente uma menor quantidade de soluções geradas; conseqüentemente seria necessário estender a busca para uma vizinhança maior, objetivando incluir movimentos que permitiriam também modificações na quantidade e localização dos *hubs*. Esta abordagem não somente aumentaria a complexidade na exploração da vizinhança de soluções, como também seria mais difícil de ser implementada, pois a mudança da localização de um *hub* implica a

realocação de todos os nós não-*hub*, no cálculo dos novos fluxos, e em uma busca tabu também para a parte locacional do problema, possivelmente requerendo sofisticados mecanismos adaptativos de memória de longo prazo, possivelmente fazendo com que a busca seja mais demorada e difícil de ser calibrada e controlada.

Dada a facilidade de se gerar boas soluções iniciais para o USAHLP, como já apresentado, os procedimentos MSTS visaram superar a dificuldade de se melhorar soluções, especialmente no que diz respeito à quantidade e localização de *hubs*, gerando soluções iniciais boas e diversificadas de uma maneira muito rápida e direta.

4.2.2 Uma busca tabu integrada para solução do USAHLP

Conforme apresentado na Seção 4.2.1 anterior, as heurísticas multi-início propostas para o USAHLP utilizaram a busca tabu para a parte alocacional do problema. Em outras palavras, uma vez que a quantidade e localização de *hubs* sejam determinadas, a busca tabu permite alcançar, de forma eficaz, soluções de alta qualidade no que diz respeito à designação dos nós aos *hubs* selecionados.

Essa abordagem guarda similaridades com a heurística híbrida desenvolvida por Abdinnour-Helm (1998), que utilizou AG e busca tabu. Em ambos os casos estas heurísticas utilizam primeiramente uma fase de diversificação (AG e multi-início), possibilitando gerar um número de soluções diversas tal que englobe a solução ótima em termos de quais nós são *hubs*.

A heurística integrada baseada em busca tabu para resolver o USAHLP, denominada nesse trabalho de HubTS, consistiu em dois estágios da busca tabu aplicados simultaneamente para determinar a quantidade e localização dos *hubs* e na definição dos *spokes* que farão a ligação aos nós não-*hub* da rede. A heurística HubTS foi inspirada na busca tabu de Skorin-Kapov, D. e Skorin-Kapov, J. (1994) para o USApHMP, e também na geração de vizinhanças de soluções para localização de *hubs* e na lista tabu para o USAHLP desenvolvido por Chen (2007).

Diferentemente da busca tabu desenvolvida por Skorin-Kapov, D. e Skorin-Kapov, J. (1994) para o USApHMP, no caso do USAHLP, a quantidade de *hubs* não é dada de antemão, mas sim esta quantidade de *hubs* deve ser determinada. É interessante notar que, no USAHLP, similar a outros problemas de localização, surge um *trade - off* entre os custos fixos de estabelecimento de *hubs* versus o custo de transporte. Em outras palavras, conforme o número de *hubs* selecionados aumenta, os custos de transporte, incluindo os custos de coleta e distribuição (devido à maior proximidade dos nós não-*hub*), bem como os custos de transferência (devido às economias de escala) tendem a diminuir.

Nesse contexto, a idéia central é que a heurística HubTS possibilita efetivamente determinar soluções de boa qualidade para o USAHLP em termos da localização dos *hubs* e da alocação dos nós não-*hub*. Tal afirmação baseia-se no fato de que para as instâncias do CAB e AP, o custo total resultante (dado pela soma dos custos de coleta, distribuição, transferência entre *hubs* e custos fixos), como uma função do número de *hubs* selecionados (p), exibe um comportamento estritamente convexo, no qual o mínimo global pode ser encontrado.

Esse comportamento estritamente convexo foi evidenciado em todos os problemas resolvidos utilizando-se os dados do CAB e do AP, contudo não é possível afirmar com exatidão que esse comportamento será o mesmo para qualquer instância do problema.

Um exemplo desse comportamento convexo pode ser visto na Figura 4.3, em que são apresentadas soluções para uma instância do conjunto de dados CAB contendo 15 nós, fator de desconto na ligação entre *hubs* $\alpha = 0,2$, e custo fixo igual a 100. Esse problema foi resolvido variando-se a quantidade de *hubs* p.de 1 até 11. A solução ótima para esse problema corresponde a uma configuração com cinco *hubs*, com custo total de 1.030,07 e pode ser determinada variando-se o número de nós *hubs* p.

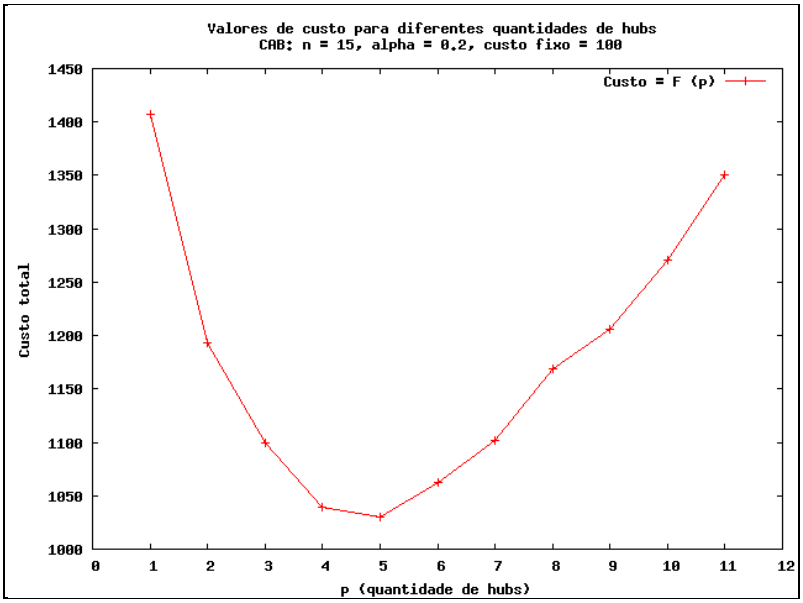

Figura 4.3: Custo da solução para diferentes números de *hubs* selecionados (*p*)

Em linhas gerais, a heurística HubTS pode ser brevemente descrita da seguinte forma: iniciando com uma configuração de rede com somente um *hub* ($p = 1$), e adicionando um *hub* por vez (isto é, fazendo $p = p + 1$), o objetivo é encontrar a melhor configuração correspondente para cada valor de p, cessando o procedimento quando o custo total, quando comparado com o custo correspondente do problema com $p - 1$ *hubs*, tiver aumentado.

Em outras palavras, a heurística HubTS desenvolvida é aplicada de forma iterativa para determinar, para diferentes e consecutivos valores de p, a melhor configuração correspondente, em termos das melhores p localizações dos *hubs* e a designação dos nós não-*hub*. O critério de parada é quando o custo total para um dado valor p de *hubs* é maior que o valor correspondente para $p - 1$.

O esqueleto básico para a heurística HubTS é mostrado na Figura 4.4.

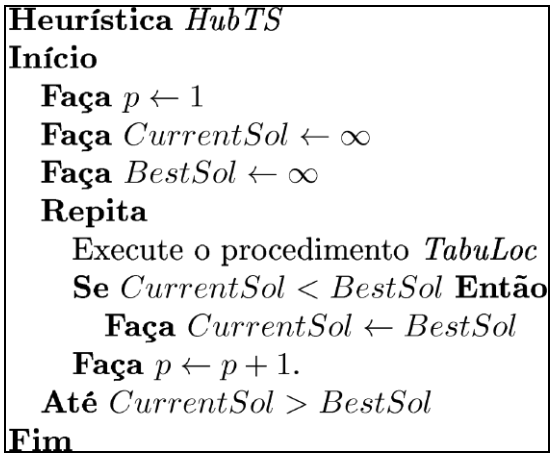

Figura 4.4: Esqueleto básico da heurística HubTS para o USAHLP.

Para cada valor de p, a melhor solução é determinada pelo procedimento TabuLoc (Figura 4.5). Uma vez que uma solução inicial viável é obtida, a busca tabu locacional tenta melhorar a solução corrente, examinando todas as possíveis possibilidades de substituição de um *hub* selecionado por um nó não-*hub*. A heurística TabuAlloc (Figura 4.6) é então aplicada para cada troca de alocação no qual o custo resultante da realocação dos nós não-*hub,* no que diz respeito à alocação ao *hub* mais próximo, não piore a solução acima de um determinado valor limite dado por $(1 + \lambda)$, onde λ é um valor pequeno não negativo.

A justificativa para não aplicar o procedimento TabuAlloc para todo movimento de troca tem a finalidade de tentar evitar que o TabuAlloc desperdiçasse tempo tentando melhorar soluções não promissoras em termos de localização de *hubs*. Em outras palavras, somente soluções promissoras, isto é, todas para as quais o custo não se deteriore acima de um valor limite, em relação à melhor solução corrente conhecida, são aceitos para o TabuAlloc tentar melhorar a realocação dos nós não-*hub* (inicialmente alocados ao *hub* mais próximo) ao novo conjunto p de *hubs* selecionados.

Dessa forma, uma iteração completa do TabuLoc consiste na avaliação de todas as trocas únicas de localização do atual conjunto de *hubs*. Em cada avaliação de troca, o TabuAlloc foi aplicado em soluções promissoras para tentar melhorar a alocação de cada nó não-*hub* (diferente do seu *hub* mais próximo). Esse procedimento foi

realizado examinando-se todas as trocas individuais únicas admissíveis (movimento não é tabu). A melhor alocação obtida, depois de um determinado número de iterações, foi utilizada na avaliação da troca de *hubs* corrente.

```
Heurística TabuLoc
Início
  Selecione os p primeiros terminais em ordem decrescente da quant. fluxo inbound
    e outbound como hubs
  Faça alocação dos terminais aos hubs mais próximos
  Calcule o custo total da solução
  Armazene a solução corrente como melhor solução
  Enquanto o número máximo de iterações tabu não for alcançado Faça
    Para cada troca única admissível entre um terminal não-hub e um hub Faça
      Realoque cada terminal não-hub ao seu novo hub mais próximo
      Determine o custo da solução corrente CurrentCost
      Se CurrentCost ≤ (1 + λ) × BestCost Então
        Execute TabuAlloc
      Fim Se
    Fim Para cada
    Realize a melhor troca de hubs permitida
    Se a solução em estudo melhora a melhor solução corrente Então
      Grave a solução em estudo como sendo a melhor solução corrente
    Fim Se
    Atualize a lista tabu alocacional
  Fim Enquanto
Fim
```

Figura 4.5: Esqueleto básico do procedimento TabuLoc para o USAHLP.

O conjunto de movimentos tabu foi gravado em duas listas tabu separadas. No TabuLoc um nó não-*hub* que substitui um *hub* é incluído na lista tabu e não pode ser substituído durante um determinado número de iterações (*tenure*). De forma análoga, no TabuAlloc um nó não-*hub* alocado a um novo *hub* é feito tabu e assim não pode ser realocado para outro *hub* por um dado número de iterações. Em ambos os casos o critério de aspiração permite que um movimento envolvendo um nó na lista tabu seja realizado se este melhorar a melhor solução corrente.

Diferente da busca tabu desenvolvida por Skorin-Kapov, D. e Skorin-Kapov, J. (1994), a heurística HubTS não utiliza mecanismo de memória de longo prazo para reiniciar a busca partindo de uma solução inicial diferente. Os experimentos realizados mostraram que esse procedimento não foi necessário, dado que não foi

alcançada nenhuma melhoria relevante em termos da qualidade da solução final obtida, fazendo somente com que a heurística HubTS ficasse mais lenta.

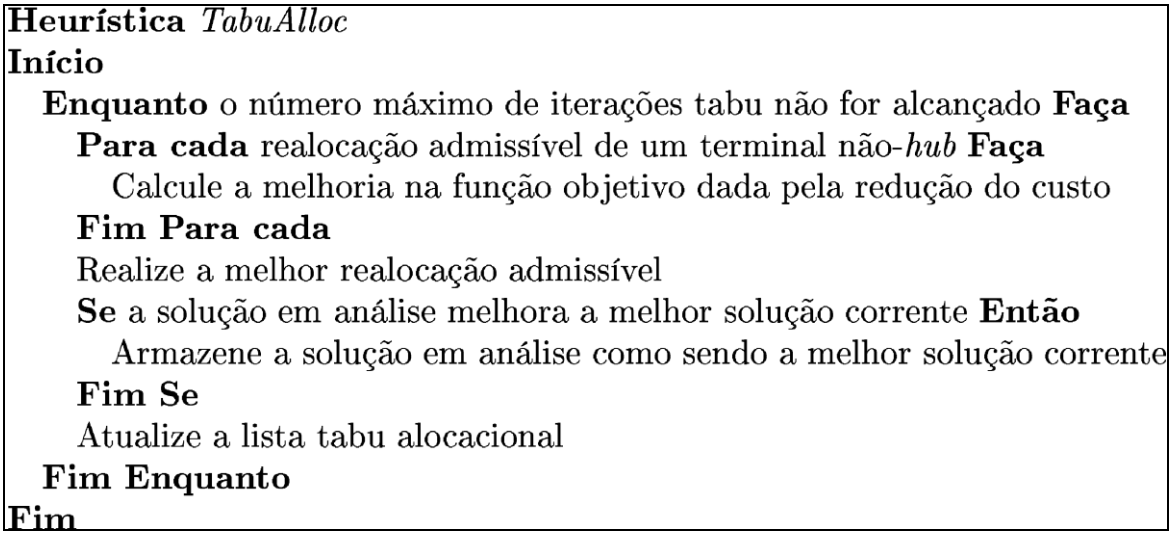

Figura 4.6: Descrição básica do procedimento TabuAlloc.

4.3 EXPERIMENTOS COMPUTACIONAIS

Todos os experimentos para resolução do USAHLP foram realizados em um computador Pentium IV 3.0 Ghz PC, com 1GB de memória RAM e sistema operacional Windows XP Professional. Todas as heurísticas foram codificadas utilizando a linguagem de programação C. Foi utilizando também o pacote comercial *ILOG CPLEX Concert Technology*, versão 9.1, para determinar a solução ótima para o USAHLP, considerando a formulação apresentada na Seção 4.1, representada pelas equações (4.1) a (4.6).

O conjunto de dados CAB (*Civil Aeronautics Board*) foi introduzido por O'Kelly (1987) e desde então tem sido comumente utilizado para comparar heurísticas para vários problemas de localização de *hubs*. São dados baseados no tráfego aéreo de passageiros entre 25 cidades dos Estados Unidos em 1970. Estas 25 cidades fazem parte de um conjunto de dados maior consistindo no fluxo entre 100 cidades. O subconjunto de 25 cidades foi escolhido por representar 51% do fluxo observado. A

distância entre as cidades satisfaz a igualdade triangular, e os fluxos entre qualquer par de cidades são simétricos, isto é, $W_{ij} = W_{ji}$.

Outro conjunto de dados utilizado na literatura é o do correio australiano conhecido como *Australian Post data set* (AP), utilizado pela primeira vez por Ernst e Krishnamoorthy (1996); e foi baseado na entrega de encomendas postais dos correios em Sidney, Australia, consistindo de 200 nós, representando na verdade distritos postais. A principal diferença entre o CAB e o AP, além da quantidade de nós da rede, é que no caso do AP a matriz de fluxo não é simétrica, e também o fluxo para um mesmo distrito postal é diferente de zero ($W_{ii} \neq 0$).

Os resultados obtidos nos experimentos computacionais realizados para solução do USAHLP, utilizando os dados tanto do conjunto CAB, como também do conjunto AP, são apresentados nas Seções 4.3.1 e 4.3.2, respectivamente.

Analisando os resultados obtidos com os experimentos computacionais realizados com o conjunto de dados AP, verificou-se que a quantidade de *hubs* localizados foi muito pequena, contendo entre 1 ou 2 *hubs* localizados para a maioria das instâncias. Com isso, foi considerada também uma versão modificada dos dados do AP, no qual fatores de redução são aplicados nos custos fixos ao se estabelecer *hubs*, objetivando aumentar o número de *hubs* na solução ótima, para se testar o desempenho dos métodos de solução propostos em problemas de maior dificuldade para solução, dado que a quantidade de *hubs* a serem localizados foi maior ao se comparar com o conjunto de dados AP original. Adicionalmente, foi desenvolvido um novo conjunto de quatro novas instâncias de grande porte, baseado no conjunto AP, compreendido por problemas com 300 e 400 nós. Esses novos experimentos estão detalhados na Seção 4.5.

Um conjunto de experimentos preliminares foi conduzido a fim de se definirem os parâmetros de controle para as heurísticas multi-início. Os resultados indicaram que, utilizando os seguintes parâmetros, torna-se possível encontrar soluções de alta qualidade em um curto tempo de CPU:

- número de sementes (rodadas independentes) igual a 50;

- 50 soluções diferentes para cada semente, totalizando 2.500 soluções geradas e exploradas;
- período tabu (*tabu tenure*) igual a 5 iterações;
- quantidade de iterações da busca tabu igual a 10 iterações.

Esses parâmetros da busca tabu são similares aos utilizados por Abdinnour-Helm (1998). No MSTS-3, φ, parâmetro utilizado para se calcular a probabilidade mínima de um nó se tornar *hub*, assumiu os valores $0,2$ e $0,5$ para os dados do CAB e do AP, respectivamente.

Os mesmos experimentos foram aplicados a fim de determinar o melhor conjunto de parâmetros para o HubTS. A duração tabu (*tabu tenure*) foi fixada em três e cinco iterações para o TabuLoc e TabuAlloc, respectivamente, enquanto que o número máximo de iterações da busca tabu foi limitado a cinco iterações para o TabuLoc, e dez iterações para o TabuAlloc.

Conforme explicitado na seção anterior, o procedimento TabuAlloc é executado somente se o custo da nova localização de *hubs* não for pior que 5% da melhor solução corrente (ou $\lambda = 0,05$). Os experimentos realizados indicaram os parâmetros de duração tabu, e quantidade total de iterações, mesmo que ligeiramente menores daqueles utilizados por outros autores (Skorin-Kapov,D e Skorin-Kapov, J. 1994; Abdinnour-Helm, 1998), possibilitaram a obtenção de soluções de alta qualidade em curtos tempos de processamento. Esses parâmetros foram mantidos fixos para todos os problemas, independente de seu tamanho.

4.3.1 Resultados obtidos com o conjunto de dados CAB

O experimento consistiu em resolver o CAB em subconjuntos de 10, 15, 20 e o conjunto de dados completo, com 25 nós. O fator de desconto α nas ligações entre *hubs* foi fixado e considerado em quatro níveis: 1,0; 0,8; 0,6; 0,4 e 0,2, e mantendo $\chi = \delta = 1$. O custo fixo F_k utilizado ao se estabelecer um *hub* foi 100, 150, 200 e 250, e igual para todos os nós.

Na Figura 4.7 é apresentado o trecho do código fonte na linguagem C++ para representar o modelo matemático apresentado nas equações de (4.1) a (4.6), utilizando o pacote ILOG CPLEX, versão 9.1.

```
// Variaveis do modelo matematico
          IloNumVarArray2 Z(env,n);
          IloNumVarArray3 Y(env,n);
          for (int i=0; i<n; i++)
          {
                 Z[i] = IloNumVarArray(env, n, 0, 1, ILOBOOL);
                 Y[i] = IloNumVarArray2(env,n);
                 for (int k=0; k<n; k++)
                 {
                        Z[i][k].setName(str);
                        Y[i][k]=IloNumVarArray(env, n, 0, IloInfinity, ILOFLOAT);
                        for (int l=0; l<n; l++)
                             Y[i][k][l].setName(str);
                 }
          }
       // criando o modelo
          IloModel model(env);

       // parcelas da FO
          IloExpr cColDist(env), cTransf(env), cFixo(env);
       // restricoes

          // loops para construir FO e restricoes simultaneamente
          for (int k=0; k<n; k++)
          {
                 cFixo+=fixo[k]*Z[k][k];
                 IloExpr constr1(env);
                 for (int i=0; i<n; i++)
                 {
                        cColDist+=dist[i][k]*Z[i][k]*(gama*O[i]+delta*D[i]);
                        model.add(Z[i][k] <= Z[k][k]); // adicionando restricao 2
                        constr1+=Z[k][i]; // construindo restricao 1 Zik=1
                        IloExpr constr3l(env);
                        IloExpr constr3r(env);
                        for (int l=0; l<n; l++)
                        {
                               cTransf+=alpha*dist[k][l]*Y[i][k][l];
                               constr3l+=(Y[i][k][l] - Y[i][l][k]);
                               constr3r+=fluxo[i][l]*Z[l][k];
                        }
                        model.add(constr3l==O[i]*Z[i][k]-constr3r);
                        constr3l.end();
                        constr3r.end();
                 }
                 model.add(constr1 == 1);
                 constr1.end();
          }
          model.add(IloMinimize(env,(((cColDist+cTransf)/divisor)+cFixo)));
          IloCplex cplex(model);
          cplex.solve();
```

Figura 4.7: Trecho de código em C++ - modelo matemático (CPLEX)

Como mencionado anteriormente, as soluções ótimas do CAB nunca tinham sido publicadas anteriormente. As melhores soluções apresentadas na literatura em trabalhos anteriores para o USAHLP foram derivadas da solução ótima do USApHMP, variando a quantidade p de *hubs*.

A formulação apresentada neste trabalho permitiu confirmar alguns desses resultados prévios reportados na literatura como ótimos, bem como verificar que outras soluções apresentadas como ótimas na literatura para alguns problemas, não estavam corretas. Adicionalmente, duas instâncias nas quais a solução ótima não era conhecida puderam ser obtidas. Estas soluções corrigidas, e novas soluções ótimas, estão destacadas na Tabela 4.1.

Tabela 4.1: Instâncias do CAB: soluções ótimas novas ou corrigidas

n	α	f_k	OptSol	Valor anterior	Referência
15	0,2	100	1030,07	nunca apresentado antes	
	0,6	150	1443,97	1456,66	Abdinnour-Helm (1998) Cunha e Silva (2007)
20	0,2	100	967,74	nunca apresentado antes	
25	1,0	100	1556,63	1559,19	Topcuoglu et al. (2005)

As soluções ótimas para os problemas do CAB obtidas com as formulações (4.1) a (4.6) são apresentados na Tabela 4.2. Para cada problema, n representa o número de nós e α o fator de desconto na ligação entre *hubs*; os valores ótimos da função objetivo estão listados na coluna com título “OptSol”, juntamente com o respectivo conjunto de *hubs* selecionados e tempos de processamento (em segundos) para alcançar a otimalidade.

Os resultados computacionais das três variantes da heurística multi-início para os dados do CAB com n =10, 15, 20 e 25 nós são apresentados nas Tabelas 4.3 e Tabela 4.4. A notação utilizada nestas tabelas foi:

- BestSol - diferença percentual entre o valor da solução ótima e a melhor solução encontrada pela heurística;
- HubProb - somente para a heurística MSTS-1: probabilidade fixa p de um nó se tornar *hub*;
- CPUt - tempo total em segundos necessário para realizar todas as rodadas independentes (nesse caso, tempo total de CPU para 50 rodadas independentes, para cada problema resolvido);
- AvgSol - diferença percentual entre a média das soluções obtidas e a solução ótima;

- SRun - percentual de rodadas independentes no qual a solução ótima foi encontrada.

Os resultados mostraram que todas as três variantes da heurística MSTS obtiveram a solução ótima para todas as instâncias do CAB com tempos de processamento muito reduzidos. Foi possível também evidenciar que o MSTS-3 foi aproximadamente quatro vezes mais rápido que o MSTS-1 e aproximadamente duas vezes mais rápido que o MSTS-2.

As diferenças no tempo de processamento podem ser explicadas pelo número de *hubs* gerados em cada solução explorada; em outras palavras, quanto maior a quantidade de *hubs* selecionados em cada solução, maior é o tempo necessário para alocar os nós não-*hub* e para a busca tabu melhorar esta solução.

Esses tempos são também muito menores do que os tempos necessários para obtenção e prova da solução ótima utilizando o CPLEX, conforme mostrado na Tabela 4.2. A variante MSTS-3 de um modo geral apresentou um desempenho um pouco melhor em termos do número de vezes (isto é, rodadas independentes) em que as soluções ótimas foram alcançadas. Para algumas instâncias, no caso do MSTS-3, a diferença percentual da solução média obtida baseada em todas as soluções geradas e a solução ótima, denotada por “AvgSol” foi ligeiramente maior que o MSTS-2; entretanto, isto não necessariamente significa um desempenho inferior. Em vez disso, pode ser visto como o resultado de um procedimento de busca mais diversificado permitiu obter soluções de melhor qualidade em relação à variante MSTS-2.

Tabela 4.2: Soluções ótimas: conjunto de dados CAB

		$n = 10$			$n = 15$			$n = 20$			$n = 25$		
α	f_k	OptSol	Hubs	CPUt(s)	OptSol	Hubs	CPUt(s)	OptSol	Hubs	CPUt(s)	OptSol	Hubs	CPUt(s)
0,2	100	791,93	4,6,7	0,172	1030,07	3,4,7,12,14	0,265	967,74	4,7,12,14,17	0,407	1029,63	4,12,17,24	1,328
	150	915,99	7,9	0,016	1239,77	4,7,12,14	0,187	1174,53	4,12,17	0,703	1217,34	4,12,17	2,609
	200	1015,99	7,9	0,016	1381,28	4,12	0,140	1324,53	4,12,17	0,579	1367,34	4,12,17	2,047
	250	1115,99	7,9	0,032	1481,28	4,12	0,078	1474,53	4,12,17	1,266	1500,90	12,20	1,547
0,4	100	867,91	4,6,7	0,141	1179,71	4,7,12,14	0,547	1127,09	1,4,12,17	1,625	1187,51	1,4,12,17	5,578
	150	974,30	7,9	0,031	1355,09	4,7,12	0,532	1297,76	4,12,17	1,484	1351,69	4,12,18	5,000
	200	1074,30	7,9	0,046	1462,62	4,12	0,172	1442,56	4,17	2,031	1501,62	12,20	5,672
	250	1174,30	7,9	0,047	1556,66	4	0,109	1542,56	4,17	1,157	1601,62	12,20	3,140
0,6	100	932,62	7,9	0,172	1309,92	4,7,12	1,438	1269,15	1,4,12,17	4,985	1333,56	2,4,12	12,672
	150	1032,62	7,9	0,047	1443,97	4,12	1,094	1406,04	4,17	3,235	1483,56	2,4,12	9,015
	200	1131,05	4	0,032	1506,66	4	0,047	1506,04	4,17	3,063	1601,20	12,20	7,547
	250	1181,05	4	0,016	1556,66	4	0,047	1570,91	6	0,187	1701,20	12,20	5125
0,8	100	990,94	7,9	0,140	1390,76	4,11	1,157	1369,52	4,17	15,641	1458,83	2,4,12	33,359
	150	1081,05	4	0,047	1456,66	4	0,063	1469,52	4,17	4,688	1594,08	12,20	20,468
	200	1131,05	4	0,031	1506,66	4	0,047	1520,91	6	0,188	1690,57	5	11,531
	250	1181,05	4	0,016	1556,66	4	0,046	1570,91	6	0,125	1740,57	5	0,562
1,0	100	1031,05	4	0,156	1406,66	4	0,078	1410,07	4,20	5,328	1556,63	4,8,20	40,109
	150	1181,05	4	0,016	1456,66	4	0,062	1470,91	6	0,250	1640,57	5	13,688
	200	1131,05	4	0,016	1506,66	4	0,046	1520,91	6	0,141	1690,57	5	0,640
	250	1181,05	4	0,031	1556,66	4	0,047	1570,91	6	0,172	1740,57	5	0,406

Tabela 4.3: Heurísticas MSTS – conjunto de dados CAB ($n = 10, 15$)

CAB data set				MSTS-1					MSTS-2				MSTS-3			
n	α	f_k	OptSol	HubProb	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun
10	0,2	100	791,93	0,10	0,00	0,031	5,02	6,00	0,00	0,078	1,88	22,00	0,00	0,046	5,24	8,00
		150	915,99	0,10	0,00	0,015	7,88	12,00	0,00	0,077	2,81	18,00	0,00	0,046	3,33	22,00
		200	1015,99	0,10	0,00	0,015	6,44	12,00	0,00	0,078	3,69	18,00	0,00	0,015	3,13	22,00
		250	1115,99	0,10	0,00	0,015	4,08	12,00	0,00	0,078	3,16	18,00	0,00	0,031	2,73	22,00
10	0,4	100	867,91	0,10	0,00	0,046	3,71	6,00	0,00	0,078	1,86	22,00	0,00	0,031	3,09	6,00
		150	974,30	0,10	0,00	0,015	6,08	12,00	0,00	0,093	3,08	18,00	0,00	0,031	2,68	22,00
		200	1074,30	0,10	0,00	0,031	3,74	12,00	0,00	0,077	2,75	18,00	0,00	0,046	2,34	22,00
		250	1174,30	0,10	0,00	0,015	0,57	12,00	0,00	0,077	0,45	18,00	0,00	0,031	0,43	22,00
10	0,6	100	932,62	0,10	0,00	0,016	5,33	12,00	0,00	0,077	1,82	18,00	0,00	0,031	2,03	22,00
		150	1032,62	0,10	0,00	0,015	3,26	12,00	0,00	0,077	2,22	18,00	0,00	0,031	1,85	22,00
		200	1131,05	0,10	0,00	0,015	0,08	92,00	0,00	0,062	0,00	98,00	0,00	0,015	0,00	98,00
		250	1181,05	0,10	0,00	0,031	0,07	92,00	0,00	0,077	0,00	98,00	0,00	0,031	0,00	98,00
10	0,8	100	990,94	0,10	0,00	0,015	2,64	12,00	0,00	0,077	1,19	18,00	0,00	0,031	0,94	22,00
		150	1081,05	0,10	0,00	0,015	0,08	92,00	0,00	0,077	0,00	98,00	0,00	0,016	0,00	98,00
		200	1131,05	0,10	0,00	0,031	0,08	92,00	0,00	0,062	0,00	98,00	0,00	0,031	0,00	98,00
		250	1181,05	0,10	0,00	0,031	0,07	92,00	0,00	0,077	0,00	98,00	0,00	0,031	0,00	98,00
10	1,0	100	1031,05	0,10	0,00	0,015	0,09	92,00	0,00	0,077	0,00	98,00	0,00	0,031	0,00	98,00
		150	1081,05	0,10	0,00	0,031	0,08	92,00	0,00	0,077	0,00	98,00	0,00	0,031	0,00	98,00
		200	1131,05	0,10	0,00	0,015	0,08	92,00	0,00	0,062	0,00	98,00	0,00	0,031	0,00	98,00
		250	1181,05	0,10	0,00	0,015	0,07	92,00	0,00	0,077	0,00	98,00	0,00	0,015	0,00	98,00
15	0,2	100	1030,07	0,15	0,00	0,250	11,59	2,00	0,00	0,201	7,81	2,00	0,00	0,078	15,18	2,00
		150	1239,77	0,15	0,00	0,250	5,89	2,00	0,00	0,186	2,69	2,00	0,00	0,078	5,19	2,00
		200	1381,28	0,15	0,00	0,234	4,25	16,00	0,00	0,186	1,03	62,00	0,00	0,078	1,90	64,00
		250	1481,28	0,15	0,00	0,234	4,24	16,00	0,00	0,186	1,44	62,00	0,00	0,062	1,55	64,00
15	0,4	100	1179,71	0,15	0,00	0,234	7,16	2,00	0,00	0,201	4,64	2,00	0,00	0,078	7,53	2,00
		150	1355,09	0,15	0,00	0,234	3,74	4,00	0,00	0,186	1,08	16,00	0,00	0,062	1,98	2,00
		200	1462,62	0,15	0,00	0,234	3,15	16,00	0,00	0,186	1,03	62,00	0,00	0,062	0,97	64,00
		250	1556,66	0,15	0,00	0,234	1,48	62,00	0,00	0,201	0,00	98,00	0,00	0,078	0,00	98,00
15	0,6	100	1309,92	0,15	0,00	0,234	3,81	4,00	0,00	0,186	2,03	16,00	0,00	0,062	3,07	2,00
		150	1443,97	0,15	0,00	0,234	1,49	16,00	0,00	0,186	0,31	62,00	0,00	0,062	0,29	64,00
		200	1506,66	0,15	0,00	0,234	1,79	62,00	0,00	0,186	0,00	98,00	0,00	0,062	0,00	98,00
		250	1556,66	0,15	0,00	0,250	1,95	62,00	0,00	0,201	0,00	98,00	0,00	0,063	0,00	98,00
15	0,8	100	1390,76	0,15	0,00	0,235	1,68	12,00	0,00	0,186	0,83	24,00	0,00	0,078	0,71	34,00
		150	1456,66	0,15	0,00	0,234	1,86	62,00	0,00	0,186	0,00	98,00	0,00	0,062	0,00	98,00
		200	1506,66	0,15	0,00	0,234	2,08	62,00	0,00	0,186	0,00	98,00	0,00	0,063	0,00	98,00
		250	1556,66	0,15	0,00	0,234	2,06	62,00	0,00	0,186	0,00	98,00	0,00	0,078	0,00	98,00
15	1,0	100	1406,66	0,15	0,00	0,235	1,71	62,00	0,00	0,186	0,00	98,00	0,00	0,078	0,00	98,00
		150	1456,66	0,15	0,00	0,234	2,17	62,00	0,00	0,201	0,00	98,00	0,00	0,062	0,00	98,00
		200	1506,66	0,15	0,00	0,234	2,13	62,00	0,00	0,201	0,00	98,00	0,00	0,062	0,00	98,00
		250	1556,66	0,15	0,00	0,234	2,06	62,00	0,00	0,186	0,00	98,00	0,00	0,078	0,00	98,00
			Média		0,00	0,129	2,89	41,45	0,00	0,133	1,20	59,45	0,00	0,050	1,65	59,35

Tabela 4.4: Heurísticas MSTS – conjunto de dados CAB ($n = 20, 25$)

CAB data set				MSTS-1					MSTS-2				MSTS-3			
n	α	f_k	OptSol	HubProb	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun
20	0,2	100	967,74	0,20	0,00	0,828	12,85	2,00	0,00	0,496	9,94	2,00	0,00	0,203	17,80	2,00
		150	1174,53	0,15	0,00	0,796	8,18	2,00	0,00	0,481	5,23	8,00	0,00	0,218	8,03	6,00
		200	1324,53	0,15	0,00	0,781	6,20	2,00	0,00	0,450	3,77	8,00	0,00	0,218	4,19	6,00
		250	1474,53	0,15	0,00	0,781	2,96	2,00	0,00	0,496	1,35	8,00	0,00	0,218	0,66	6,00
20	0,4	100	1127,09	0,20	0,00	0,828	7,71	2,00	0,00	0,449	5,37	6,00	0,00	0,187	8,61	2,00
		150	1297,76	0,15	0,00	0,781	5,61	2,00	0,00	0,480	3,06	8,00	0,00	0,218	3,55	6,00
		200	1442,56	0,15	0,00	0,781	3,14	8,00	0,00	0,434	1,13	66,00	0,00	0,187	0,75	72,00
		250	1542,56	0,15	0,00	0,796	2,20	8,00	0,00	0,542	0,76	66,00	0,00	0,187	0,39	72,00
20	0,6	100	1269,15	0,20	0,00	0,828	4,23	2,00	0,00	0,434	2,21	6,00	0,00	0,203	2,80	2,00
		150	1406,04	0,15	0,00	0,796	2,98	8,00	0,00	0,449	0,98	66,00	0,00	0,203	0,26	86,00
		200	1506,04	0,15	0,00	0,796	1,79	8,00	0,00	0,434	0,49	66,00	0,00	0,203	0,10	86,00
		250	1570,91	0,15	0,00	0,796	1,74	30,00	0,00	0,465	0,53	36,00	0,00	0,203	0,08	74,00
20	0,8	100	1369,52	0,15	0,00	0,796	2,10	8,00	0,00	0,434	0,64	66,00	0,00	0,218	0,14	86,00
		150	1469,52	0,15	0,00	0,796	1,13	8,00	0,00	0,573	0,20	66,00	0,00	0,203	0,02	86,00
		200	1520,91	0,15	0,00	0,796	1,92	30,00	0,00	0,496	0,58	36,00	0,00	0,234	0,08	74,00
		250	1570,91	0,15	0,00	0,796	2,38	30,00	0,00	0,418	0,62	36,00	0,00	0,218	0,08	74,00
20	1,0	100	1410,07	0,15	0,00	0,796	1,64	6,00	0,00	0,418	0,93	14,00	0,00	0,203	0,83	10,00
		150	1470,91	0,15	0,00	0,781	1,87	30,00	0,00	0,418	0,61	36,00	0,00	0,234	0,08	74,00
		200	1520,91	0,15	0,00	0,781	2,40	30,00	0,00	0,403	0,64	36,00	0,00	0,218	0,08	74,00
		250	1570,91	0,15	0,00	0,796	2,61	30,00	0,00	0,418	0,62	36,00	0,00	0,218	0,08	74,00
25	0,2	100	1029,63	0,20	0,00	2,140	10,11	2,00	0,00	0,791	8,83	4,00	0,00	0,375	20,18	2,00
		150	1217,34	0,15	0,00	2,110	7,35	2,00	0,00	0,775	5,37	20,00	0,00	0,406	13,41	4,00
		200	1367,34	0,15	0,00	2,125	7,16	2,00	0,00	0,899	4,26	20,00	0,00	0,406	10,07	4,00
		250	1500,90	0,15	0,00	2,109	6,66	2,00	0,00	0,868	3,53	4,00	0,00	0,375	7,17	2,00
25	0,4	100	1187,51	0,15	0,00	2,203	7,80	2,00	0,00	0,899	4,64	4,00	0,00	0,437	10,75	2,00
		150	1351,69	0,15	0,00	2,156	6,45	4,00	0,00	0,806	4,32	6,00	0,00	0,406	9,53	4,00
		200	1501,62	0,15	0,00	2,125	5,36	2,00	0,00	0,806	2,86	4,00	0,00	0,390	5,90	2,00
		250	1601,62	0,15	0,00	2,125	6,00	2,00	0,00	0,791	3,99	4,00	0,00	0,375	5,85	2,00
25	0,6	100	1333,56	0,15	0,00	2,250	6,08	2,00	0,00	0,837	3,64	2,00	0,00	0,406	8,85	2,00
		150	1483,56	0,15	0,00	2,234	5,16	2,00	0,00	0,837	2,74	2,00	0,00	0,421	5,62	2,00
		200	1601,20	0,15	0,00	2,187	4,46	2,00	0,00	0,791	3,05	4,00	0,00	0,375	4,02	2,00
		250	1701,20	0,15	0,00	2,344	3,64	2,00	0,00	0,791	2,75	4,00	0,00	0,390	2,87	2,00
25	0,8	100	1458,83	0,15	0,00	2,296	4,58	2,00	0,00	0,868	2,83	2,00	0,00	0,406	4,77	2,00
		150	1594,08	0,15	0,00	2,140	2,79	2,00	0,00	0,791	1,79	4,00	0,00	0,390	2,19	2,00
		200	1690,57	0,15	0,00	2,125	1,92	10,00	0,00	0,807	0,99	24,00	0,00	0,437	0,59	50,00
		250	1740,57	0,15	0,00	2,125	2,87	10,00	0,00	0,791	1,16	24,00	0,00	0,437	0,57	50,00
25	1,0	100	1556,63	0,15	0,00	2,140	2,22	4,00	0,00	0,853	1,64	2,00	0,00	0,406	1,79	2,00
		150	1640,57	0,15	0,00	2,203	2,20	10,00	0,00	0,791	1,12	24,00	0,00	0,484	0,61	50,00
		200	1690,57	0,15	0,00	2,109	3,15	10,00	0,00	0,791	1,19	24,00	0,00	0,406	0,59	50,00
		250	1740,57	0,15	0,00	2,109	3,84	10,00	0,00	0,806	1,16	24,00	0,00	0,391	0,57	50,00
			Média		0,00	1,482	4,39	8,3	0,00	0,639	2,54	21,95	0,00	0,307	4,11	31,45

Nas Tabelas 4.5 e 4.6 são apresentados os resultados para a heurística HubTS; essas tabelas também permitem a comparação com a variante MSTS-3, que foi a que apresentou o melhor desempenho dentre as três variantes da heurística multi-início.

Analisando-se os resultados apresentados nas Tabelas 4.5 e 4.6, pôde-se verificar que a heurística HubTS também alcançou a solução ótima em todos os problemas resolvidos do conjunto de dados CAB, porém requerendo tempos de processamento inferiores ao se comparar com a heurística MSTS-3 (em média três vezes mais rápido).

Os resultados também evidenciaram que, conforme a quantidade de *hubs* obtidos cresce, a heurística HubTS não se comporta tão bem no que diz respeito ao tempo de processamento, quando o custo fixo ao se estabelecer um *hub* está no menor valor (100).

Isto pode ser explicado pelo fato de que, quanto maior o número de *hubs* na solução ótima correspondente, maior esforço computacional é necessário para o HubTS determinar o correto número de *hubs* p e para a busca tabu melhorar a alocação dos nós não-*hub* na presença de uma quantidade maior de *hubs* selecionados.

Foi ainda realizada a comparação das duas melhores heurísticas propostas neste trabalho, HubTS e MSTS-3, com a heurística SATLUHLP proposta por Chen (2007). De acordo com o autor, SATLUHLP (uma heurística híbrida baseada em *simulated annealing*, busca tabu e procedimentos de melhoria), supera as heurísticas previamente desenvolvidas apresentadas na literatura e fornece os melhores resultados conhecidos com menores tempos de processamento.

Dado que as heurísticas HubTS, MSTS-3 e SATLUHLP foram capazes de encontrar a solução ótima em todos os 80 problemas do conjunto CAB, a comparação foi centrada somente no que diz respeito ao desempenho computacional, medido em termos do tempo total de processamento.

Apesar de não garantir a mesma igualdade de condições, foram resolvidos todos os problemas do conjunto de dados CAB utilizando o HubTS e o MSTS-3 em um computador com a mesma configuração utilizada em Chen (2007), isto é, um PC Pentium-IV 1,6 Ghz. Os tempos apresentados para a heurística SATLUHLP foram obtidos diretamente do artigo do autor. A comparação pode ser vista na Tabela 4.7.

Os resultados claramente evidenciaram que ambas as heurísticas HubTS e MSTS-3 superaram o SATLUHLP em termos do tempo de processamento em todos os problemas, e que a heurística HubTS foi, em média, mais de seis vezes mais rápida que o SATLUHLP.

4.3.2 Resultados obtidos com o conjunto de dados AP

O conjunto de dados denominado AP (*Australian Post*) *data set* foi derivado de uma aplicação real de uma rede de entregas postais (ERNST; KRISHNAMOORTY, 1996), possuindo 200 nós, que representam distritos postais contendo suas coordenadas, volumes de tráfego, bem como χ, α e δ. Os valores dos custos de coleta (nó de origem $\rightarrow$ *hub*), transferência entre *hubs* (*hub* $\leftrightarrow$ *hub*), e de distribuição (*hub* $\rightarrow$ nó de destino) são constantes e dados por $\chi = 3$, $\alpha = 0,75$ e $\delta = 2$, respectivamente.

Em outras palavras, no AP o custo unitário de coleta (de um nó de origem para um *hub*) é maior que o custo de distribuição (dos *hubs* para os nós de destino), que por sua vez é maior que o custo de transferência entre *hubs*. Os volumes de tráfego não são simétricos (isto é, $W_{ij} \neq W_{ji}$), e adicionalmente encomendas postais podem possuir a sua origem e destino no mesmo distrito postal (isto é, $W_{ii} \neq 0$).

O conjunto AP também considera dois tipos de custos fixos nos *hubs*: *tight* e *loose*. De acordo com Ernst e Krishnamoorthy (1999), problemas *tight* são mais difíceis de serem resolvidos, dado que possuem maiores custos fixos em nós com grande fluxo, tornando assim mais difícil para esses nós com alto volume de tráfego serem escolhidos como *hubs*, ao contrário dos problemas *loose* que possuem custos fixos menores, possibilitando a localização de uma maior quantidade de *hubs.*

Tabela 4.5: Resultados para o HubTS – conjunto de dados CAB ($n = 10, 15$)

				HubTS		MSTS-3						HubTS		MSTS-3	
n	α	f_k	OptSol	BestSol	CPUt(s)	BestSol	CPUt(s)	n	α	f_k	OptSol	BestSol	CPUt(s)	BestSol	CPUt(s)
10	0,2	100	791,93	0,00	0,015	0,00	0,046	15	0,2	100	1030,07	0,00	0,109	0,00	0,078
		150	915,99	0,00	0,000	0,00	0,046			150	1239,77	0,00	0,062	0,00	0,078
		200	1015,99	0,00	0,000	0,00	0,015			200	1381,28	0,00	0,000	0,00	0,078
		250	1115,99	0,00	0,000	0,00	0,031			250	1481,28	0,00	0,015	0,00	0,062
10	0,4	100	867,91	0,00	0,000	0,00	0,031	15	0,4	100	1179,71	0,00	0,062	0,00	0,078
		150	974,30	0,00	0,000	0,00	0,031			150	1355,09	0,00	0,031	0,00	0,062
		200	1074,30	0,00	0,000	0,00	0,046			200	1462,62	0,00	0,015	0,00	0,062
		250	1174,30	0,00	0,000	0,00	0,031			250	1556,66	0,00	0,015	0,00	0,078
10	0,6	100	932,62	0,00	0,000	0,00	0,031	15	0,6	100	1309,92	0,00	0,031	0,00	0,062
		150	1032,62	0,00	0,000	0,00	0,031			150	1443,97	0,00	0,000	0,00	0,062
		200	1131,05	0,00	0,000	0,00	0,015			200	1506,66	0,00	0,000	0,00	0,062
		250	1181,05	0,00	0,000	0,00	0,031			250	1556,66	0,00	0,000	0,00	0,063
10	0,8	100	990,94	0,00	0,000	0,00	0,031	15	0,8	100	1390,76	0,00	0,015	0,00	0,078
		150	1081,05	0,00	0,000	0,00	0,016			150	1456,66	0,00	0,000	0,00	0,062
		200	1131,05	0,00	0,000	0,00	0,031			200	1506,66	0,00	0,000	0,00	0,063
		250	1181,05	0,00	0,000	0,00	0,031			250	1556,66	0,00	0,000	0,00	0,078
10	1,0	100	1031,05	0,00	0,000	0,00	0,031	15	1,0	100	1406,66	0,00	0,000	0,00	0,078
		150	1081,05	0,00	0,000	0,00	0,031			150	1456,66	0,00	0,000	0,00	0,062
		200	1131,05	0,00	0,000	0,00	0,031			200	1506,66	0,00	0,015	0,00	0,062
		250	1181,05	0,00	0,000	0,00	0,015			250	1556,66	0,00	0,000	0,00	0,078
Média				0,00	0,001	0,00	0,030	Média				0,00	0,019	0,00	0,069

Tabela 4.6: Resultados para o HubTS – conjunto de dados CAB ($n = 20, \dot{2}5$)

				HubTS		MSTS-3						HubTS		MSTS-3	
n	α	f_k	OptSol	BestSol	CPUt(s)	BestSol	CPUt(s)	n	α	f_k	OptSol	BestSol	CPUt(s)	BestSol	CPUt(s)
20	0,2	100	967,74	0,00	0,266	0,00	0,203	25	0,2	100	1029,63	0,00	0,437	0,00	0,375
		150	1174,53	0,00	0,062	0,00	0,218			150	1217,34	0,00	0,203	0,00	0,406
		200	1324,53	0,00	0,078	0,00	0,218			200	1367,34	0,00	0,203	0,00	0,406
		250	1474,53	0,00	0,078	0,00	0,218			250	1500,90	0,00	0,062	0,00	0,375
20	0,4	100	1127,09	0,00	0,14	0,00	0,187	25	0,4	100	1187,51	0,00	0,546	0,00	0,437
		150	1297,76	0,00	0,078	0,00	0,218			150	1351,69	0,00	0,234	0,00	0,406
		200	1442,56	0,00	0,015	0,00	0,187			200	1501,62	0,00	0,078	0,00	0,390
		250	1542,56	0,00	0,031	0,00	0,187			250	1601,62	0,00	0,093	0,00	0,375
20	0,6	100	1269,15	0,00	0,172	0,00	0,203	25	0,6	100	1333,56	0,00	0,218	0,00	0,406
		150	1406,04	0,00	0,031	0,00	0,203			150	1483,56	0,00	0,25	0,00	0,421
		200	1506,04	0,00	0,046	0,00	0,203			200	1601,20	0,00	0,078	0,00	0,375
		250	1570,91	0,00	0,015	0,00	0,203			250	1701,20	0,00	0,093	0,00	0,390
20	0,8	100	1369,52	0,00	0,062	0,00	0,218	25	0,8	100	1458,83	0,00	0,265	0,00	0,406
		150	1469,52	0,00	0,062	0,00	0,203			150	1594,08	0,00	0,078	0,00	0,390
		200	1520,91	0,00	0,015	0,00	0,234			200	1690,57	0,00	0,031	0,00	0,437
		250	1570,91	0,00	0,015	0,00	0,218			250	1740,57	0,00	0,046	0,00	0,437
20	1,0	100	1410,07	0,00	0,078	0,00	0,203	25	1,0	100	1556,63	0,00	0,343	0,00	0,406
		150	1470,91	0,00	0,015	0,00	0,234			150	1640,57	0,00	0,031	0,00	0,484
		200	1520,91	0,00	0,015	0,00	0,218			200	1690,57	0,00	0,031	0,00	0,406
		250	1570,91	0,00	0,015	0,00	0,218			250	1740,57	0,00	0,031	0,00	0,391
Média				0,00	0,064	0,00	0,210	Média				0,00	0,168	0,00	0,406

Tabela 4.7: Comparação do desempenho das heurísticas HubTS, MSTS-3 e SATLUHLP (CHEN, 2007)

Tempos de processamento – conjunto de dados CAB

		$n = 10$			$n = 15$			$n = 20$			$n = 25$		
α	f_k	SATLUHLP*	HubTS	MSTS-3	SATLUHLP*	HubTS	MSTS-3	SATLUHLP*	HubTS	MSTS-3	SATLUHLP*	HubTS	MSTS-3
0,2	100	0,16	0,000	0,040	0,63	0,140	0,100	1,25	0,296	0,340	1,79	0,484	0,610
	150	0,11	0,000	0,040	0,46	0,109	0,100	0,61	0,078	0,330	1,10	0,218	0,550
	200	0,12	0,000	0,040	0,22	0,015	0,100	0,63	0,078	0,340	1,10	0,234	0,550
	250	0,11	0,000	0,040	0,23	0,016	0,100	0,62	0,093	0,330	0,70	0,078	0,550
0,4	100	0,16	0,015	0,040	0,46	0,093	0,100	0,94	0,140	0,320	1,90	0,609	0,550
	150	0,10	0,015	0,030	0,34	0,046	0,100	0,64	0,093	0,330	1,19	0,250	0,600
	200	0,11	0,000	0,040	0,22	0,015	0,100	0,35	0,031	0,310	0,76	0,093	0,550
	250	0,11	0,000	0,040	0,08	0,015	0,100	0,33	0,031	0,320	0,76	0,078	0,550
0,6	100	0,11	0,000	0,030	0,35	0,046	0,100	0,95	0,203	0,320	1,30	0,250	0,520
	150	0,11	0,000	0,040	0,22	0,015	0,100	0,36	0,046	0,310	1,28	0,281	0,530
	200	0,03	0,000	0,040	0,08	0,000	0,100	0,35	0,031	0,310	0,84	0,093	0,550
	250	0,03	0,000	0,040	0,07	0,000	0,100	0,09	0,015	0,310	0,84	0,109	0,550
0,8	100	0,12	0,000	0,040	0,23	0,015	0,100	0,38	0,047	0,320	1,44	0,296	0,530
	150	0,03	0,000	0,040	0,07	0,000	0,090	0,37	0,062	0,310	0,89	0,093	0,550
	200	0,03	0,015	0,030	0,07	0,000	0,090	0,09	0,015	0,320	0,43	0,031	0,550
	250	0,03	0,000	0,040	0,07	0,015	0,100	0,08	0,015	0,320	0,42	0,031	0,550
1,0	100	0,04	0,000	0,040	0,07	0,015	0,100	0,47	0,078	0,310	1,52	0,390	0,580
	150	0,04	0,000	0,040	0,06	0,015	0,100	0,13	0,015	0,320	0,24	0,016	0,550
	200	0,04	0,000	0,040	0,06	0,015	0,100	0,13	0,015	0,320	0,24	0,031	0,550
	250	0,04	0,000	0,040	0,06	0,000	0,100	0,13	0,015	0,320	0,24	0,031	0,550
	Média	0,08	0,002	0,040	0,20	0,029	0,100	0,44	0,070	0,320	0,95	0,185	0,550

(*) tempos obtidos diretamente em Chen (2007)

Nesta seção foram resolvidos problemas de tamanho n =10, 20, 25, 40, 50, 100 e 200 nós, a partir dos dados obtidos diretamente de Andreas T. Ernst, um dos autores que introduziram pela primeira vez esse conjunto de dados na literatura (ERNST; KRISHNAMOORTHY, 1996), uma vez que não estão disponíveis na OR-Library (BEASLEY, 1990) os custos fixos para as instâncias maiores do AP com 100 e 200 nós.

As soluções ótimas para os problemas do conjunto AP foram obtidas utilizando-se também o CPLEX, sendo os problemas gerados com programa em C++ apresentado na Figura 4.7. As soluções ótimas para todas as instâncias do conjunto AP são apresentadas na Tabela 4.8. Como mencionado anteriormente, tais soluções ótimas nunca haviam sido publicadas anteriormente. As instâncias do AP com 200 nós continuam ainda sendo um desafio, dado que não foi possível encontrar a solução ótima para esses problemas devido ao limite de memória dos computadores de 32 *bits*. Para os problemas de 200 nós foi estimado ser necessário mais de 4 *gigabytes* de memória para a árvore de enumeração. Adicionalmente, após 24 horas consecutivas de processamento (CPU) e mais de 2 *gigabytes* de memória alocada, o gap entre a melhor solução inteira encontrada e a de relaxação linear dos nós da árvore *branch-and-bound*, estava ainda acima de 20%. Para todos os outros problemas do conjunto AP a solução ótima pôde ser obtida em um tempo de processamento considerado razoável, pois todos os tempos foram inferiores a 25 segundos, exceto para a instância de 100 nós com custo fixo *loose*, que necessitou em torno de cinco minutos para que a solução ótima fosse encontrada.

Tabela 4.8: Soluções ótimas para os problemas do conjunto de dados AP

	Loose			Tight		
n	OptSol	Hubs	CPUt(s)	OptSol	Hubs	CPUt(s)
10	224250,05	3,4,7	0,078	263399,94	4,5,10	0,125
20	234690,95	7,14	0,220	271128,18	7,19	0,250
25	236650,62	8,18	0,563	295667,84	13	0,251
40	240986,23	14,28	12,364	293164,83	19	1,002
50	237421,98	15,36	15,512	300420,98	24	22,771
100	238016,28	29,73	275,976	305097,96	52	23,139

Deve-se destacar que as maiores instâncias do AP, com custo fixo *tight*, apresentaram somente um *hub* na solução ótima final, dado que nós com maior tráfego, que seriam naturais candidatos a *hub*, possuem maior custo fixo, sendo assim inviável, do ponto de vista de custo, que estes fossem selecionados como *hubs*.

A Tabela 4.9 apresenta os resultados computacionais obtidos ao se resolver os problemas do conjunto de dados AP com custos fixos *loose* e *tight* utilizando as heurísticas propostas: MSTS (nas suas três variantes) e HubTS. A notação utilizada nestas tabelas seguiu o mesmo padrão que nos resultados apresentados na Seção 4.3.1 para os problemas do conjunto CAB.

Quando se comparam os resultados das três variantes da heurística MSTS, pode se verificar que todas estas heurísticas puderam obter a solução ótima para todos os 12 problemas do AP em que a solução ótima é conhecida (isto é, para $n \leq 100$).

De forma similar aos resultados obtidos para os problemas do conjunto CAB, a heurística MSTS-3 exibiu desempenho ligeiramente superior à MSTS-2, no que diz respeito à qualidade das soluções obtidas e aos menores tempos de processamento, notadamente para as instâncias de maior porte. Um exemplo é o caso do problema com 200 nós e custo fixo *loose*: a heurística MSTS-2 não conseguiu obter a melhor solução para esse problema.

Contudo, tanto a heurística MSTS-2 como a heurística MSTS-3 apresentam um desempenho superior quando comparadas com a heurística MSTS-1. Um ponto importante a se destacar é que as instâncias do conjunto AP com custos fixos *tight* tendem a ser mais difíceis de serem resolvidas, entretanto, isto não afetou o desempenho computacional: o tempo total necessário para resolver cada problema foi considerado pequeno para todas as três variantes da heurística MSTS.

A fim de se avaliar a qualidade das soluções obtidas pelas heurísticas MSTS para os problemas de 200 nós, em que as soluções ótimas não puderam ser determinadas utilizando o CPLEX, utilizou-se a formulação matemática já apresentada para se resolver o subproblema de alocação dos nós aos *hubs* localizados, admitindo que as

melhores soluções em termos da quantidade e localização dos *hubs*, determinadas pelas heurísticas MSTS, estão corretas. Em outras palavras, definidos os *hubs* como dado de entrada, utilizou-se a formulação matemática para determinar a alocação dos nós não-*hub.*

Com base nos resultados obtidos das alocações ótimas para os problemas de 200 nós, resolvidas utilizando o CPLEX, pôde-se comprovar que a busca tabu implementada nas três variantes MSTS obteve sucesso na determinação da alocação ótima dos nós não-*hub* para esses problemas. Para essas duas instâncias, MSTS-3 obteve melhores resultados tanto em termos da qualidade da solução final obtida como também em termos de tempos de processamento. Os melhores resultados obtidos com os problemas de 200 nós foram todos obtidos com a heurística MSTS-3.

A heurística HubTS, quando comparada com as três variantes MSTS, obteve os melhores resultados em termos da qualidade da solução final obtida, como pode ser visto na Tabela 4.9. Entretanto, os tempos de processamento foram ligeiramente maiores quando comparados, por exemplo, com os tempos obtidos com a heurística MSTS-3, notadamente para as instâncias com 200 nós.

O desempenho apresentado pela heurística HubTS é, de certa forma, esperado, pois uma etapa do método que consome grande parte do tempo de processamento é relativa a solução da parte locacional do problema, que define a quantidade e localização dos *hubs*. Assim, os tempos de processamento tendem a se deteriorar mais que nas outras heurísticas, particularmente MSTS-3 para o USAHLP, conforme o tamanho do problema, e a quantidade de *hubs* localizados, aumentam.

Foi possível também notar que as soluções ótimas e aproximadas obtidas para as instâncias do AP não puderam ser diretamente comparadas com os resultados publicados na literatura (TOPCUOGLU et al., 2005 e CHEN, 2007), dado que existem algumas diferenças em termos dos valores das melhores soluções reportadas nos trabalhos desses autores.

Essas diferenças podem ser devido a alguma divergência em termos dos dados de entrada; em particular, diferentes considerações em termos dos custos fixos utilizados para as maiores instâncias do AP, que não estão disponíveis na OR-Library, como mencionado anteriormente (os dados utilizados nesse trabalho foram obtidos diretamente com o Dr. Andreas T. Ernst). Essas diferenças estão detalhadas abaixo:

- Para o caso de $n = 200$ e custo fixo do tipo *loose*, Topcuoglu et al. (2005) reportaram um custo total de 228.944,77 com dois *hubs* localizados nos nós 53 e 184. Entretanto, se esses dois nós (53,184) fossem definidos como *hubs*, a alocação ótima determinada utilizando a formulação matemática já apresentada, resultaria em um custo total de 352.383,37, que é superior à solução aqui apresentada (223.803,00) selecionando os *hubs* nos nós 44 e 149.

- A mesma diferença ocorre para $n = 200$ e custo fixo *tight.* O melhor custo reportado por Topcuoglu et al. (2005) para esse problema foi 233.537,93 com dois *hubs* localizados nos nós 53 e 184. Entretanto, se esses dois nós (53, 184) fossem definidos como *hubs*, a alocação ótima resultaria em um custo de 356.976,52, muito maior que a solução obtida pelas heurísticas MSTS (272.237,78) com *hubs* localizados nos nós 55 e 123.

Finalmente, deve se relatar que os resultados aqui apresentados coincidem com alguns resultados reportados em Ernst e Krishnamoorthy (1999) para o CSAHLP (isto é, a versão capacitada do USAHLP) para problemas com folga de capacidade (*loose*), no qual as restrições de capacidade nos *hubs* são altas e possivelmente não alcançadas, confirmando assim a coerência dos resultados aqui apresentados, que possivelmente podem ser ótimos.

Tabela 4.9: Resultados das Heurísticas MSTS e HubTS – conjunto de dados AP

AP data set			MSTS-1				MSTS-2				MSTS-3				HubTS	
n	f_k-type	OptSol	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)
10	*loose*	224250,05	0,00	0,016	8,09	2,00	0,00	0,015	5,12	2,00	0,00	0,016	7,88	4,00	0,00	0,015
20	*loose*	234690,95	0,00	0,422	7,14	2,00	0,00	0,109	5,52	10,00	0,00	0,062	9,87	6,00	0,00	0,046
25	*loose*	236650,62	0,00	1,234	7,52	2,00	0,00	0,203	6,65	4,00	0,00	0,125	9,31	8,00	0,00	0,125
40	*loose*	240986,23	0,00	0,125	12,21	2,00	0,00	0,812	5,71	6,00	0,00	0,375	8,75	4,00	0,00	0,781
50	*loose*	237421,98	0,00	0,422	13,45	2,00	0,00	1,688	8,73	2,00	0,00	0,703	11,16	2,00	0,00	1,891
100	*loose*	238016,28	0,00	13,219	10,91	2,00	0,00	11,906	9,06	2,00	0,00	4,484	7,95	2,00	0,00	28,515
200	*loose*	233803,02*	1,94	1147,515	13,92	0,00	3,39	743,328	11,78	0,00	0,00	678,469	10,27	2,00	0,00	310,875
		Média	0,28	166,136	10,46	1,71	0,48	108,294	7,51	3,71	0,00	97,747	9,31	4,00	0,00	48,893
10	*tight*	263399,94	0,00	0,031	3,02	4,00	0,00	0,015	4,42	2,00	0,00	0,015	4,31	2,00	0,00	0,015
20	*tight*	271128,18	0,00	0,422	3,57	6,00	0,00	0,109	4,21	2,00	0,00	0,062	4,85	2,00	0,00	0,062
25	*tight*	295667,84	0,00	1,156	3,99	40,00	0,00	0,218	2,97	60,00	0,00	0,094	3,32	58,00	0,00	0,031
40	*tight*	293164,83	0,00	0,140	3,59	54,00	0,00	0,812	4,78	44,00	0,00	0,344	4,95	44,00	0,00	0,188
50	*tight*	300420,98	0,00	0,438	3,63	44,00	0,00	1,578	4,14	36,00	0,00	0,719	4,84	40,00	0,00	0,469
100	*tight*	305097,96	0,00	12,281	20,99	20,00	0,00	11,781	23,71	14,00	0,00	5,187	24,23	12,00	0,00	7,141
200	*tight*	272237,78*	0,69	1112,360	31,67	0,00	0,00	742,203	26,41	2,00	0,00	693,922	20,13	2,00	0,00	536,688
		Média	0,10	160,975	10,07	24,00	0,00	108,102	10,09	22,85	0,00	100,049	9,52	22,86	0,00	77,799

*otimalidade não provada.

4.4 NOVAS INSTÂNCIAS COM CUSTOS FIXOS REDUZIDOS

Baseado na informação de que o número de *hubs* na solução ótima pode afetar a desempenho das heurísticas multi-início e busca tabu desenvolvidas, no que diz respeito ao tempo de processamento e qualidade das soluções obtidas, e também no fato de que o número de *hubs* na solução ótima para as instâncias do AP é pequeno (especialmente instâncias grandes com custos fixos *tight* para os quais a solução ótima compreende somente um *hub*), decidiu-se estender os experimentos computacionais para também considerar versões modificadas do AP, aplicando um fator de redução θ aos custos fixos ao se estabelecer *hubs*. Os valores utilizados para θ foram 0,9 e 0,8.

De forma análoga às instâncias originais do AP, foi possível encontrar a solução ótima para estas instâncias modificadas para problemas com até 100 nós, sendo que as soluções ótimas para esses problemas são apresentadas na Tabela 4.10. Os resultados mostram que o número de *hubs* aumentou para a maioria dos problemas. Pode se evidenciar também que, para algumas instâncias, especialmente quando $n = 100$ e custos fixos *loose*, os tempos de processamento aumentam significativamente, refletindo a grande dificuldade para resolver problemas com mais *hubs* na solução ótima respectiva.

Os resultados computacionais das três variantes da heurística MSTS e do HubTS para os novos dados do AP são apresentados na Tabela 4.11. Foram mantidos os mesmos parâmetros utilizados nas heurísticas MSTS e HubTS para solução dos problemas anteriores.

Analisando-se os resultados obtidos ao se resolver estas instâncias modificadas, utilizando as heurísticas MSTS e HubTS, pode se verificar uma leve superioridade do MSTS-3 frente às outras duas variantes da heurística MSTS, tanto no que diz respeito à qualidade final das soluções obtidas como também no menor tempo de processamento. Entretanto, a solução ótima não pode ser obtida para algumas instâncias. Para o MSTS-3 o desvio ficou abaixo de 2% e 3% para custos fixos *loose* e *tight* respectivamente.

Os resultados também mostraram que o HubTS foi capaz de encontrar a solução ótima, ou a melhor solução (no caso de 200 nós), para todas estas instâncias modificadas. Por outro lado, de forma análoga aos resultados prévios, os tempos de processamento do HubTS foram maiores do que a heurística MSTS-3, mas estas diferenças são maiores que as observadas nas instâncias originais do AP.

Novamente, isso pode ser explicado pelo fato de que quanto maior a quantidade de *hubs* na solução ótima, maior será o esforço computacional necessário para a heurística HubTS determinar o correto número de *hubs*, e em seguida para a busca tabu melhorar a alocação dos nós não-*hub* na presença de uma quantidade maior de *hubs* selecionados.

Tabela 4.10: Soluções ótimas – conjunto de dados AP com $\theta = 0,9$ e $\theta = 0,8$

		Loose			Tight		
n	θ	Opt, Sol	Hubs	CPUt(s)	Opt, Sol	Hubs	CPUt(s)
10	0,9	215425,88	3,4,7	0,062	253798,41	3,4,10	0,125
20	0,9	228785,69	7,14	0,390	263350,03	7,19	0,297
25	0,9	230539,77	8,18	0,922	288066,69	9,24	0,812
40	0,9	234013,75	11,22,28	10,469	289382,06	19	1,422
50	0,9	231658,98	15,36	17,954	295239,13	17,48	13,501
100	0,9	232712,97	29,73	390,111	301719,69	52	249,777
10	0,8	206555,04	1,4,5,7	0,110	242936,29	1,4,5,10	0,078
20	0,8	222085,76	6,11,14	0,313	255571,86	7,19	0,265
25	0,8	224428,91	8,18	1,000	279548,44	9,24	0,765
40	0,8	226032,08	11,22,28	6,703	285599,31	19	7,000
50	0,8	225895,98	15,36	27,141	281803,93	17,48	7,234
100	0,8	227409,66	29,73	890,090	29764626	5,52	88,345

Em contrapartida, o fato do MSTS-3 não conseguir obter, em algumas instâncias, a solução ótima para os problemas, pode também sugerir que o número de rodadas independentes deva ser aumentado para conseguir uma maior diversidade devido ao maior número de *hubs*, o que pode acarretar um maior tempo de processamento para execução.

Tabela 4.11: Heurísticas MSTS e HubTs – conjunto de dados $\theta = 0,9$ e $\theta = 0,8$

AP data set				MSTS-1				MSTS-2				MSTS-3				HubTS	
n	f_k-type	θ	OptSol	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)	AvgSol	SRun	BestSol	CPUt(s)
10	*loose*	0,9	215425,88	0,00	0,015	9,18	2,00	0,00	0,015	6,12	2,00	0,00	0,015	9,31	4,00	0,00	0,015
20	*loose*	0,9	228785,69	0,00	0,422	6,72	2,00	0,00	0,125	5,23	10,00	0,00	0,078	9,89	6,00	0,00	0,046
25	*loose*	0,9	230539,77	0,00	1,172	7,07	2,00	0,00	0,203	6,59	4,00	0,00	0,140	9,55	4,00	0,00	0,109
40	*loose*	0,9	234013,75	0,65	0,125	13,94	0,00	0,66	0,813	6,21	0,00	0,00	0,484	9,72	2,00	0,00	2,047
50	*loose*	0,9	231658,98	0,00	0,422	14,65	2,00	0,00	1,688	8,71	2,00	0,00	1,078	11,51	2,00	0,00	1,860
100	*loose*	0,9	232712,97	0,00	13,266	11,36	2,00	0,00	12,282	8,91	2,00	0,00	43,860	12,09	2,00	0,00	28,094
200*	*loose*	0,9	228753,70	1,85	1165,890	13,45	0,00	3,19	742,921	11,13	0,00	0,00	690,734	14,09	2,00	0,00	467,829
			Média	0,36	168,759	10,91	1,43	0,55	108,292	7,56	2,86	0,00	106,198	10,88	3,14	0,00	71,429
10	*loose*	0,8	206555,04	0,02	0,016	10,31	0,00	0,02	0,016	6,98	0,00	0,00	0,015	10,36	4,00	0,00	0,000
20	*loose*	0,8	222085,76	0,35	0,421	6,55	0,00	0,00	0,108	5,15	10,00	0,00	0,078	10,74	2,00	0,00	0,140
25	*loose*	0,8	224428,91	0,00	1,172	6,58	2,00	0,00	0,202	6,45	4,00	0,00	0,141	9,88	4,00	0,00	0,125
40	*loose*	0,8	226032,08	1,71	0,140	16,29	0,00	1,71	0,793	7,11	0,00	0,00	0,486	11,11	2,00	0,00	2,016
50	*loose*	0,8	225895,98	0,00	0,437	15,78	2,00	0,00	1,679	8,59	2,00	0,00	1,081	11,81	2,00	0,00	1,843
100	*loose*	0,8	227409,66	0,00	13,312	11,68	2,00	0,00	11,759	8,47	2,00	1,39	13,312	11,92	0,00	0,00	27,828
200*	*loose*	0,8	223704,44	1,75	1173,314	12,86	0,00	2,85	742,478	10,28	0,00	0,00	679,706	14,28	2,00	0,00	463,047
			Média	0,55	169,830	11,44	0,86	0,65	108,148	7,58	2,57	0,20	99,260	11,44	2,29	0,00	70,714
10	*tight*	0,9	253798,41	0,00	0,016	4,27	4,00	0,12	0,015	6,06	0,00	0,00	0,031	6,74	2,00	0,00	0,000
20	*tight*	0,9	263350,03	0,00	0,438	5,03	6,00	0,00	0,109	5,85	2,00	0,00	0,094	6,71	2,00	0,00	0,047
25	*tight*	0,9	288066,69	0,00	1,156	4,43	6,00	0,59	0,218	4,01	0,00	0,00	0,187	5,41	2,00	0,00	0,140
40	*tight*	0,9	289382,06	0,00	0,140	3,32	54,00	0,00	0,796	4,33	44,00	0,00	0,562	4,59	44,00	0,00	0,188
50	*tight*	0,9	295239,13	0,00	0,437	3,51	0,00	0,00	1,563	3,96	0,00	0,00	1,141	4,67	40,00	0,00	2,266
100	*tight*	0,9	301719,69	0,00	12,406	19,32	20,00	0,00	11,719	21,56	14,00	0,00	6,937	22,16	12,00	0,00	7,140
200*	*tight*	0,9	266275,22	0,84	1121,704	30,79	0,00	0,00	743,438	26,41	2,00	1,58	50,109	17,56	0,00	0,00	625,097
			Média	0,12	162,328	10,10	12,86	0,10	108,265	10,31	8,86	0,23	8,437	9,69	14,57	0,00	90,697
10	*tight*	0,8	242936,29	0,12	0,015	5,84	0,00	0,77	0,015	7,99	0,00	0,00	0,016	10,16	2,00	0,12	0,015
20	*tight*	0,8	255571,86	0,00	0,421	6,18	6,00	0,00	0,110	7,38	2,00	0,00	0,062	8,49	2,00	0,00	0,062
25	*tight*	0,8	279548,44	0,00	1,170	5,08	6,00	0,12	0,220	5,25	0,00	0,00	0,110	6,86	2,00	0,00	0,156
40	*tight*	0,8	285599,31	0,00	0,155	3,03	54,00	0,00	0,818	3,87	44,00	0,00	0,343	4,21	44,00	0,00	0,187
50	*tight*	0,8	281803,93	2,92	0,452	6,31	0,00	2,93	1,591	6,73	0,00	0,00	0,750	6,83	2,00	0,00	2,281
100	*tight*	0,8	297646,26	0,23	12,662	17,75	0,00	0,23	12,141	19,62	0,00	0,23	5,015	20,26	0,00	0,00	36,032
200*	*tight*	0,8	260312,67	1,01	1129,026	29,42	0,00	0,00	747,145	26,18	2,00	1,62	39,625	18,27	0,00	0,00	623,297
			Média	0,61	163,414	10,52	9,43	0,58	108,863	11	6,86	0,26	6,560	10,73	7,43	0,02	94,576

*otimalidade não provada.

4.5 NOVAS INSTÂNCIAS BASEADAS NO CONJUNTO AP

Como sugerido por alguns autores, (por exemplo, Ernst e Krishnamoorthy (1996, 1999) e Ebery et al. (2000)), o grande desafio da atualidade na solução do problema de configuração de redes *hub-and-spoke* é resolver, de forma eficiente, instâncias cada vez maiores que são comuns em situações práticas. Dessa forma, com base nos dados completos do conjunto AP com 200 nós, foram geradas quatro novas instâncias de grande porte correspondentes a 300 e 400 nós.

De forma análoga às instâncias do conjunto de dados AP, para cada tamanho de problema, dois tipos de custos fixos foram considerados: *loose* e *tight*, ambos compartilhando os mesmos dados geográficos (isto é, as localizações dos nós) e fluxos. As instâncias com custos fixos *loose* e *tight* com 400 nós foram geradas de acordo com o seguinte procedimento:

1. Duas constantes d_{min} e d_{max} foram definidas, com valores 500 e 2000 respectivamente. Esses valores foram escolhidos com base na distância média e máxima entre dois nós no problema AP original com 200 nós.
2. Para cada nó i da instância AP com os 200 nós cujas coordenadas são (X_i, Y_i), foram gerados dois novos nós i' e i''com as coordenadas $(X_{i'}, Y_{i'})$ e $(X_{i''}, Y_{i''})$, sendo as mesmas determinadas de acordo com as seguintes expressões:

$$X_{i'} = X_i + [d_{min} + r_i\ (d_{max} - d_{min})]\ cos(\phi) \quad (4.10)$$

$$Y_{i'} = Y_i + [d_{min} + r_i\ (d_{max} - d_{min})]\ sin(\phi) \quad (4.11)$$

$$X_{i''} = X_i + [d_{min} + r_i\ (d_{max} - d_{min})]\ cos(\phi + \pi) \quad (4.12)$$

$$Y_{i''} = Y_i + [d_{min} + r_i\ (d_{max} - d_{min})]\ sin(\phi + \pi) \quad (4.13)$$

onde r é um número aleatório no intervalo entre $[0, 1]$ e ϕ é um ângulo aleatório no intervalo $[0, \pi]$.

3. Para qualquer par de nós (i, j) dado no AP original, a quantidade correspondente de fluxo W_{ij} foi igualmente distribuída entre os quatro novos nós gerados, isto é:$W_{i'j'} = W_{i'j''} = W_{i''j'} = W_{i''j''} = 0.25W_{ij}$.
4. O custo fixo para os novos nós i' e i'' são diferentes e dados por:

$$f_{i'} = (1 - r_i)\ f_i \quad (4.14)$$

$$f_{i''} = (1 - \overline{r}_i)\ f_i \quad (4.15)$$

onde r_i e $\overline{r}_i)$ são dois números aleatórios dentro do intervalo $[0; 0, 3]$.

Dessa forma, para cada nó *i* da instância AP de 200 nós, os dois nós i' e i'' estão simetricamente localizados e a mesma distância no que diz respeito à i. É interessante também notar que estas novas instâncias são intrinsecamente mais difíceis de serem resolvidas, dado que os custos fixos ao se estabelecer *hubs* são proporcionalmente baixos, levando assim não somente a uma maior quantidade de *hubs* na solução ótima, mas também para combinações mais promissoras em termos da seleção de *hubs,* levando a soluções de alta qualidade a serem exploradas.

Foi utilizado um programa escrito em C, disponível eletronicamente na OR-Library (BEASLEY, 1990), para gerar as duas instâncias de 300 nós a partir das instâncias de 400 nós. Esse programa foi criado e utilizado por Ernst e Krishnamoorthy (1996) para gerar instâncias menores a partir do conjunto de dados AP completo com 200 nós, e consistiu em combinar nós e fluxos, com o objetivo de se assegurar que as novas instâncias geradas sejam uma aproximação razoável do problema original.

Na Tabela 4.12 são apresentados os resultados obtidos com as heurísticas MSTS-3 e HubTS para as novas instâncias de grande porte geradas. Decidiu-se não avaliar as heurísticas MSTS-1 e MSTS-2, dado que MSTS-3 conseguiu alcançar melhores resultados em todos os experimentos conduzidos.

As colunas “Referência” e “# hubs” da Tabela 4.12 denotam, respectivamente, a melhor solução obtida resolvendo-se os problemas com o HubTS e o número de

hubs encontrados na melhor solução obtida por cada heurística. Para os problemas com 300 e 400 nós, a heurística HubTS conseguiu alcançar melhores resultados, ambos em termos da qualidade da solução final obtida, como também no menor tempo de processamento.

Tabela 4.12: Resultados para o conjunto de dados AP com 300 e 400 nós

			HubTS			MSTS-3		
n	**Tipo fk**	**Referência**	***# hubs***	**CPUt(s)**	**Gap (%)**	***# hubs***	**CPUt(s)**	**Gap (%)**
300	*loose*	264.837,88	4	4.605,078	0,00	3	6.221,969	2,79
400	*loose*	268.164,13	4	8.447,437	0,00	2	15.116,031	3,84
300	*tight*	276.047,75	3	3.483,375	0,00	2	6.266,391	5,67
400	*tight*	284.212,47	3	9.530,828	0,00	2	15.225,438	3,08

4.6 CONCLUSÕES DO CAPÍTULO

O modelo matemático proposto se mostrou eficiente na obtenção da solução ótima dos 92 problemas resolvidos, sendo 80 problemas do conjunto de dados CAB e 12 problemas do conjunto de dados AP, para redes com até 100 nós, em um razoável tempo de processamento.

Para os problemas com 100 nós do AP, o tempo de processamento necessário para se obter a solução ótima utilizando o CPLEX variou de aproximadamente 5 minutos até 15 minutos, tempo esse que se mostrou diretamente relacionado à quantidade de *hubs* localizados. De acordo com os trabalhos já apresentados na literatura, que estudaram especificamente o USAHLP, esta pesquisa foi o primeiro esforço de sucesso para a obtenção de soluções exatas para o USAHLP.

As heurísticas desenvolvidas apresentaram um bom desempenho, podendo obter a solução ótima em todos os 80 problemas do conjunto de dados CAB, e nos 12 problemas do conjunto de dados AP, em um tempo de processamento muito pequeno. Um exemplo é o problema de 100 nós e custo fixo *loose* do AP que pôde ser resolvido pela heurística MSTS-3 em menos de 5 segundos.

Tanto a heurística MSTS (nas suas três variantes), como também a heurística HubTS, se mostraram muito eficientes para solução dos problemas, sendo

heurísticas simples e fáceis de serem implementadas, e ainda robustas no que diz respeito à qualidade das soluções obtidas em todos os problemas testados, não necessitando de calibrações específicas de parâmetros para obtenção de soluções de qualidade.

A heurística HubTS, de um modo geral, obteve as melhores soluções para os problemas utilizando um tempo de processamento menor. Esse melhor desempenho ficou significativamente evidenciado nos quatro problemas de grande porte gerados, com 300 e 400 nós, obtendo soluções 3 a 6% melhores que a heurística MSTS-3, necessitando tempos de processamento 35 a 80% menores.

À partir dos resultados apresentados, conclui-se que as heurísticas aqui descritas para solução do USAHLP constituem o estado-da-arte no que diz respeito à obtenção de soluções ótimas ou aproximadas para o problema.

5 CARREGAMENTO DE CARGAS PARCELADAS EM UMA REDE DE TRANSPORTE

Este capítulo trata do problema de carregamento de cargas parceladas em uma rede transporte do tipo *hub-and-spoke*. É proposta uma formulação matemática e, tendo em vista a dificuldade para sua solução, é proposta também uma heurística baseada em busca tabu para se resolver problemas de tamanho como o encontrado na prática.

O capítulo anterior tratou da decisão em nível de planejamento estratégico relacionada à configuração de redes do tipo *hub-and-spoke.* Mais especificamente, busca-se determinar o número e a localização dos terminais concentradores de carga (*hubs*) e a alocação dos demais terminais aos *hubs*.

Conforme visto anteriormente no Capítulo 2, um despacho é composto por um único documento fiscal conhecido por Conhecimento de Transporte Rodoviário de Cargas (CTRC), e corresponde a um embarque único contendo uma origem, um destino e dimensões (peso e volume) definidos, um único cliente remetente e um único cliente destinatário. Da mesma forma, uma carga corresponde ao agrupamento de todos os despachos que possuem o mesmo terminal de origem e destino, enquanto que uma rota representa uma ligação entre dois terminais para o transporte de cargas, sendo operada por um ou mais veículos, não necessariamente do mesmo tamanho no que diz respeito à capacidade em peso e volume transportado.

No projeto de redes *hub-and-spoke*, especificamente para o USAHLP aqui tratado, como solução do problema, tem-se uma rede em que não são permitidas rotas diretas entre terminais não-*hub*, sem paradas intermediárias para consolidação. E ainda, dada a alocação única dos terminais não-*hub* aos *hubs* localizados, o roteamento da carga se torna trivial, pois todo o tráfego recebido e enviado é sempre direcionado ao *hub* alocado.

Entretanto, não é o que ocorre na prática, pois na operação diária de empresas de transporte rodoviário de carga parcelada, carregamentos diretos são não só

realizados, como também incentivados devido ao menor custo para operação. Em algumas situações são realizadas modificações no padrão apresentado acima para o fluxo da carga, visando também reduzir o trabalho de manuseio nos *hubs* intermediários, além de melhorar o nível de serviço.

Conforme visto no Capítulo 2, a forma como as cargas são despachadas para os seus destinos nem sempre seguem o fluxos de uma estrutura / configuração do tipo *hub-and-spoke*. Isso não significa que a modelagem do problema estratégico de configuração da rede do tipo *hub-and-spoke* seja inadequada para representar corretamente o problema encontrado em transportadoras de carga parcelada, mas sim que no dia-a-dia podem ocorrer situações que levam a decisões que não podem ser representadas no nível estratégico. Uma das mais relevantes é a variação na demanda de carga entre terminais; a outra é a consideração da capacidade dos veículos que realizam o transporte entre terminais.

Às vezes, se toda a carga originada em um terminal for enviada ao *hub* ao qual está alocado e daí para os demais *hubs*, isso pode acarretar ociosidade dos veículos, ou ainda pode comprometer, dependendo do caso, o prazo de entrega, pois, cada parada intermediária em um terminal de consolidação adiciona em média um dia no prazo total para a carga chegar ao seu terminal de destino, que será responsável pela entrega final. Assim, em algumas situações é melhor despachar parte das cargas diretamente para os destinos ao invés de passar por dois *hubs*, caso exista carga suficiente para ocupar razoavelmente um veículo direto para algum terminal de destino, ou então com somente uma parada intermediária em um único *hub*.

Na Figura 5.1 são apresentadas quatro alternativas possíveis de rotas para um fluxo originado no terminal *i* com destino o terminal *j*, e que os terminais *i* e *j* estejam alocados respectivamente aos *hubs* K e L. Existem, portanto quatro alternativas básicas de percurso para que estas cargas cheguem ao seu destino:

1. Carregamento direto de *i* para *j*. O carregamento direto de *i* para *j* somente é justificado se existir quantidade de carga suficiente para ocupar razoavelmente a capacidade de um veículo;
2. Carregamento via *hub* de origem K, onde o terminal *i* está alocado. No carregamento de *i* para o *hub* K, além das cargas destinadas ao terminal *j*,

podem ser enviadas também cargas para outros terminais ligados ao *hub* K, ou outro hub da rede, dado que em uma rede hub-and-spoke, todos os hubs estão interconectados;

3. Carregamento via *hub* de destino L, onde o terminal *j* está alocado. No carregamento de *i* para o *hub* L, além das cargas destinadas ao terminal *j*, da mesma forma que na situação anterior, podem ser enviadas cargas para todos os terminais alocados ao *hub* L, ou para outros *hubs* da rede;
4. A última alternativa seria o envio das cargas de *i* para *j* estritamente via *hubs*, sendo então enviadas do terminal *i* para o *hub* K, do *hub* K para o *hub* L, e do *hub* L para o terminal de destino *j*.

Todas as quatro alternativas de rotas apresentadas são diretamente dependentes da quantidade total de cargas entre os terminais *i* e *j*, e também das cargas originadas no terminal *i* para os outros terminais / hubs da rede de transporte.

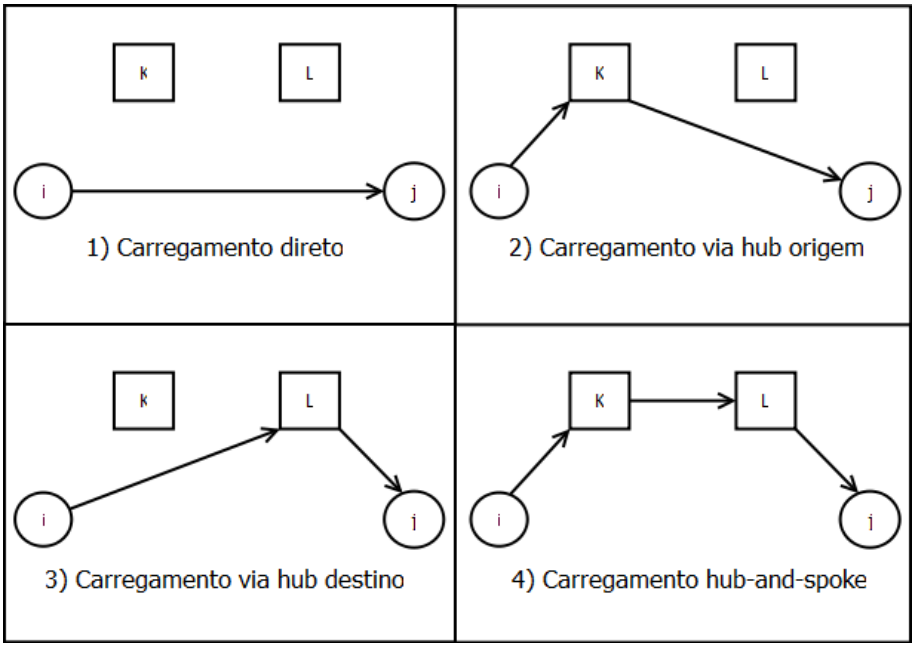

Figura 5.1: Possíveis rotas entre os terminais *i* e *j* alocados aos *hubs* K e L

Uma última alternativa de rota consideraria uma parada intermediária em um *hub* diferente dos *hubs* ao quais os terminais *i* e *j* estão alocados. Esse tipo de rota é mais raro de se ocorrer, mas deve ser prevista na modelagem. Um exemplo desse tipo de rota pode ser visto na Figura 5.2 em que a carga de *i* para *j* é enviada para o

seu *hub* de origem K, e em seguida para o *hub* intermediário ‘M’ para aí sim alcançar o terminal de destino *j*.

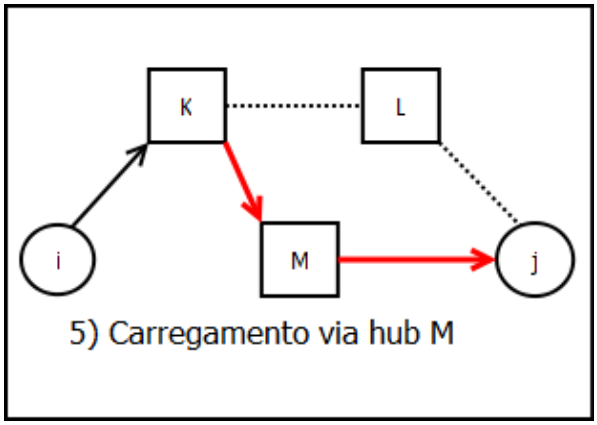

Figura 5.2: Alternativa de rota entre os terminais *i* e *j* utilizando um *hub* ‘M’

Empresas transportadoras de carga parcelada utilizam basicamente dois tipos de veículos para o transporte de cargas entre terminais, que são:

3. *Truck*: veículo com até 12.000 kg de capacidade líquida de carga em peso, e 40 a 50 m³ de capacidade em volume, com uma carroceria do tipo baú.

4. *Carreta*: veículo composto por um cavalo trator e um semi-reboque, também com carroceria do tipo baú, tendo o conjunto a capacidade líquida de carga, em peso, de 25.000 kg, e em volume de 90 a 110 m³.

Nesta pesquisa, para efeito de estudo, foram considerados esses dois tipos de veículo, sendo que muitas vezes, em situações práticas, as ligações entre *hubs* são realizadas utilizando principalmente carretas, e nas ligações entre *hubs* e terminais, em alguns casos, são utilizados também veículos menores, do tipo truck.

Busca-se propor um método para solução do problema da definição de uma programação de carregamento entre terminais no transporte de carga parcelada, que pode ser sintetizado como: dada uma rede *hub-and-spoke* criada, definir as rotas a serem operadas, e o dimensionamento da quantidade e tipo de veículos necessários em cada rota, para que todas as cargas sejam transportadas até o seu

terminal de destino a um menor custo possível, valendo-se da possibilidade de consolidação de cargas nos *hubs*.

Esse problema é comumente tratado em nível de decisão tático, e está centrado especificamente na definição do fluxo (percurso ou caminho) que cada carga deve percorrer até chegar ao seu destino, tendo como dado de entrada uma rede de transporte do tipo *hub-and-spoke* com as possíveis ligações entre terminais já pré-definidas. Dado um conjunto de cargas a serem transportadas, deseja-se, portanto decidir como essas cargas serão movimentadas nessa rede, estabelecendo as rotas a serem operadas, incluindo as possíveis paradas intermediárias, e também na definição da quantidade e tipo de veículo utilizado em cada rota.

Esse problema possui grande importância prática, pois o objetivo consiste em definir diferentes formas de consolidação e as rotas a serem operadas, tentando acompanhar a variação na demanda por transporte de carga parcelada, sendo essa variação comum no Brasil, em certas semanas do mês, e em certos meses do ano.

Conforme visto na revisão bibliográfica, o problema de carregamento de cargas em uma rede de uma transportadora, conhecido na literatura específica como *network loading problem* (NLP), é um problema *NP-hard* (MAGNANTI; MIRCHANDANI; VACHANI, 1995), portanto não existem métodos exatos eficientes para solução de problemas de tamanho prático.

O restante desse capítulo está organizado da seguinte forma: na Seção 5.1 é apresentado o modelo matemático proposto para representar o problema de carregamento de carga parcelada em uma rede de transporte.

Dada a dificuldade, evidenciada nos experimentos computacionais, para se obter a solução ótima do modelo matemático formulado para o problema através do pacote de otimização CPLEX, é proposta uma estratégia de solução baseada em busca tabu, que é descrita na Seção 5.2.

A fim de facilitar o entendimento do problema, e também do modelo matemático e da heurística de solução baseada em busca tabu, é apresentado na Seção 5.3 um

problema exemplo de pequeno porte contendo 16 terminais, 14 cargas, 89 rotas alternativas e dois tipos de veículos (truck e carreta).

Na Seção 5.4 são apresentados os experimentos computacionais utilizando dados reais de uma empresa transportadora de carga parcelada pelo modal rodoviário no Brasil. Finalmente, na Seção 5.5 são apresentadas as conclusões deste capítulo.

5.1 MODELO MATEMÁTICO

O modelo matemático proposto para representar o problema de carregamento de carga parcelada em uma rede de transporte foi inspirado no problema conhecido na literatura como o *network loading problem* (NLP).

O NLP modela situações em que o custo variável unitário ($/t) associado à quantidade de carga a ser transportada é igual a zero, e veículos com capacidade fixa, e custos que independem da sua ocupação, estão disponíveis para o transporte de cargas. Esses veículos podem ser alocados a rotas de um conjunto pré-definido de rotas candidatas que ligam os terminais, sendo que mais de um veículo, não necessariamente do mesmo tipo, pode ser alocado a uma rota. Conforme visto no capítulo 2, esse problema corresponde a uma variante do problema do projeto de redes capacitado com múltiplas *commodities*

Mais especificamente, esse problema consiste na definição dos arcos ou ligações a serem utilizados, dentre um conjunto de ligações possíveis entre nós ou terminais de uma rede, de tal forma a atender toda a demanda de transporte, de cada origem para cada destino, ao menor custo possível. O custo nos arcos corresponde a um valor fixo proporcional ao número de veículos utilizados, caso o mesmo seja selecionado.

O modelo matemático aqui proposto utilizou os conceitos apresentados na revisão bibliográfica para o problema “não bifurcado” do *network loading problem* (NLP), estudado pela primeira vez por Barahona (1996), e consiste em restringir que cada carga deva percorrer um único caminho até chegar ao seu destino, não permitindo o seu particionamento (ou fracionamento) entre dois caminhos que ligam dois nós.

Essa consideração é muito relevante em termos práticos, em razão da forma como operam as transportadoras de carga parcelada, pois é fundamental que o gerente operacional local de cada terminal de origem tenha rotas pré-definidas como uma regra única a ser seguida no que diz respeito à forma com que as cargas devam ser carregadas, especialmente com relação às possíveis paradas em terminais intermediários, até chegarem aos seus destinos. Em outras palavras, é indesejável que uma carga, de um dado terminal de origem para um terminal de destino, seja fracionada em veículos que seguem rotas distintas, com tempos de viagem diferentes, dificultando o rastreamento das encomendas dos clientes que compõem a carga.

Para a modelagem do problema, define-se N como o conjunto de terminais (nós) da rede em estudo, K o conjunto de cargas a serem transportadas, T o conjunto de tipos de veículos disponíveis (em geral dois tipos), e ainda A o conjunto de arcos candidatos para a seleção das rotas da rede aos quais serão alocados veículos. Como explicado anteriormente, esses arcos são pré-definidos, e podem ser de um dos cinco tipos apresentados anteriormente com base no percurso que uma carga pode percorrer para chegar ao seu destino (carregamento direto, carregamento via hub de origem, carregamento via hub de destino, carregamento passando por dois hubs e carregamento via outro hub intermediário), tendo em vista a configuração resultante de uma rede do tipo *hub-and-spoke*.

Deve-se salientar que a definição das alternativas de percurso das cargas, que determinam os arcos candidatos que compõem o conjunto *A*, é decisão da empresa, com base na sua experiência e em restrições, condicionantes e regras de negócio que não podem ser modeladas.

Em geral consideram-se, por exemplo, arcos correspondentes a ligações diretas para os pares de terminais cujas demandas de carga superem a capacidade de um veículo carreta; no entanto, como essas demandas quase nunca permitem ocupar totalmente, na prática, um número de veículos inteiros (por exemplo, a demanda direta entre dois terminais lota 1,3 carretas), é necessário considerar também ligações entre esses dois terminais via *hub* de origem, via *hub* de destino e, entre os

dois *hubs*. Essa é a lógica que os gerentes operacionais utilizam na definição das alternativas de envio das cargas.

Definem-se ainda os seguintes parâmetros de entrada para o problema:

- R^k: peso da carga k, $k \in K$, a ser transportada;
- $O(k)$: origem da carga k, $k \in K$;
- $D(k)$: destino da carga k, $k \in K$;
- a_{ij}^t: custo de cada veículo do tipo t, $t \in T$ alocado ao arco $(i,j) \in A$. Esse custo é fixo para um determinado veículo, independente da quantidade de carga carregada no mesmo.
- $\mathcal{K}^t$: capacidade do veículo do tipo t, $t \in T$;

As variáveis de decisão para o problema são:

- x_{ij}^t: variáveis de fluxo nos arcos, do tipo inteira não negativa, indicando o número de veículos do tipo t, $t \in T$, alocados ao arco $(i,j) \in A$;
- f_{ij}^{kt}: variável binária que recebe o valor 1 se a carga k, $k \in K$, é transportada por um veículo do tipo t, $t \in T$, no arco $(i,j) \in A$ e zero caso contrário

O modelo matemático proposto para representar o problema de carregamento de carga parcelada em uma rede de transporte pode ser formulado como segue:

$$\text{Min} \sum_{t \in T} \sum_{(i,j) \in A} a_{ij}^t x_{ij}^t, \tag{5.1}$$

sujeito às restrições:

$$\sum_{t \in T} \left(\sum_{\{j \in N:(i,j) \in A\}} f_{ij}^{kt} - \sum_{\{l \in N:(l,i) \in A\}} f_{li}^{kt} \right) = \begin{cases} 1, & se\ i = O(k) \\ -1, & se\ i = D(k) \\ 0, & \text{caso contrario} \end{cases} \quad i \in N, k \in K, \tag{5.2}$$

$$\sum_{k \in K} R^k f_{ij}^{kt} \leq \mathcal{K}^t x_{ij}^t \ \ \forall (i,j) \in A,\ t \in T, \tag{5.3}$$

$$x_{ij}^{t} \geq 0,\ inteiro,\quad \forall (i,j) \in A,\ t \in T, \tag{5.4}$$

$$f_{ij}^{kt} \in \{0,1\},\quad \forall (i,j) \in A, k \in K,\ t \in T. \tag{5.5}$$

A função objetivo (5.1) visa minimizar o custo resultante das decisões de criação de rotas e da definição da quantidade de veículos alocados em cada rota. As restrições de conservação de fluxo para cada uma das cargas em cada terminal são representadas pelas restrições (5.2). Já as restrições (5.3) asseguram que a quantidade total de cargas alocadas a um dado tipo de veículo, em um determinado percurso, não deve exceder a capacidade disponível.

Embora não necessárias para a viabilidade do problema, verificou-se que, se forem adicionadas à formulação matemática as restrições (5.6) e (5.7), elas tornam a formulação mais robusta, melhorando a qualidade da relaxação linear, e acarretando um menor número de iterações do *branch-and-bound* para se atingir a solução ótima, conforme evidenciado em experimentos computacionais com o CPLEX em problemas de pequeno porte, que podem ser resolvidos até a sua otimalidade.

$$\sum_{t \in T} f_{ij}^{kt} \leq \sum_{t \in T} x_{ij}^{t},\quad \forall (i,j) \in A, k \in K, \tag{5.6}$$

$$R^{k} f_{ij}^{kt} \leq \mathcal{K}^{t},\quad \forall (i,j) \in A, k \in K,\ t \in T, \tag{5.7}$$

$$\sum_{t \in T} f_{ij}^{kt} \leq 1,\quad \forall (i,j) \in A, k \in K. \tag{5.8}$$

As restrições (5.6) são similares às utilizadas por Barahona (1996) para o problema “não bifurcado”, com somente um tipo de veículo. Já as restrições (5.7) visam garantir que nenhuma carga individual exceda a capacidade de um veículo disponível (de qualquer tipo), ou seja, a maior carga deve possuir um peso menor que a capacidade do menor veículo disponível para transporte (menor capacidade em peso); dessa forma, restringe-se que veículos de menor capacidade sejam

alocados a arcos cuja somatória de fluxo supere a sua capacidade. As restrições (5.8) ajudam a limitar a variável, já declarada como binária em (5.5), a desiginar uma carga a somente um tipo de veículo em um dado percurso.

Como apresentado na Seção 3.2 da revisão bibliográfica, o NLP é um problema fortemente *NP-hard* e há pouca esperança de desenvolvimento de algoritmos teoricamente eficientes para sua solução (MAGNANTI; MIRCHANDANI; VACHANI, 1995). A versão não bifurcada adiciona ainda mais complexidade ao problema, pois impõe que as variáveis de fluxos no modelo matemático sejam declaradas como binárias. O NLP que permite bifurcação nos fluxos possui somente $|A|$ variáveis inteiras, enquanto que o problema não bifurcado possui, além da mesma quantidade de variáveis inteiras, $|A| \times |K| \times |T|$ variáveis binárias.

Experimentos computacionais realizados, utilizando o CPLEX versão 9.1, indicaram que esta formulação é capaz de resolver apenas problemas de pequeno porte, com número muito reduzido de terminais, arcos e cargas, significativamente inferiores às instâncias de problemas encontrados na prática das transportadoras de carga fracionada, o que exige a proposição de uma estratégia de solução heurística, conforme apresentado a seguir.

5.2 ESTRATÉGIA DE SOLUÇÃO BASEADA EM BUSCA TABU

Dada a complexidade para resolução do problema de carregamento de carga parcelada em uma rede de transporte, é proposto neste trabalho um método de solução baseado na metaheurística busca tabu. Conforme visto no capítulo anterior, a busca tabu é um procedimento de busca local que utiliza estruturas de memória para guiar os movimentos de uma solução viável para outra, com o objetivo de se explorar regiões do espaço de busca que poderiam não ser explorados, tentando escapar do ótimo local. Os princípios fundamentais da busca tabu são apresentados em detalhes em Glover (1986, 1989).

A descrição básica da heurística desenvolvida para se resolver o problema de carregamento de carga parcelada em uma rede de transporte pode ser visualizada

na Figura 5.6. Os principais componentes envolvidos na estratégia baseada em busca tabu aqui proposta são:

1. procedimento para geração de solução inicial;
2. função para cálculo da quantidade e tipo de veículos utilizados em cada rota;
3. função para cálculo do custo da solução;
4. a busca tabu propriamente dita.

Cada um deles é detalhado a seguir.

5.2.1 Geração da solução inicial

Para a geração da solução inicial do problema, é necessário propor um método para selecionar os arcos a serem utilizados e, concomitantente, determinar o número de veículos de cada tipo que será alocado ao arco.

Para tanto, seja $A = \{1, \ldots, m\}$ cada um dos arcos do conjunto de arcos possíveis de serem utilizados para a definição das rotas a serem operadas para o carregamento de carga parcelada na rede de transporte. Isso permite utilizar um vetor binário de tamanho igual à cardinalidade do conjunto A (ou simplesmente $|A|$), para representar uma solução para o problema.

A Figura 5.3 ilustra, de forma gráfica, o vetor criado para armazenar os arcos selecionados na solução. A posição com valor igual a 1 nesse vetor significa que o arco correspondente a esta posição foi inserido na solução inicial, ou zero caso contrário. De acordo com a figura, por exemplo, os arcos de número 2, 4, m-2, m-1 e m foram escolhidos para fazerem parte da solução.

1	2	3	4	...	m-2	m-1	m
0	1	0	1		1	1	0

Figura 5.3: Representação da solução

Para a obtenção da solução inicial, inicialmente são ordenadas todas as cargas a serem transportadas, por par origem-destino, em ordem decrescente de peso total. O algoritmo de ordenação utilizado foi uma versão modificada do *quicksort* descrita em Singleton (1969).

Em seguida, iniciando com a carga de maior peso, determina-se o caminho mínimo partindo do terminal de origem da carga, até o seu terminal de destino, tendo como entrada uma rede fictícia contendo todos os arcos disponíveis no conjunto A, tendo como atributo dos arcos a distância entre terminais. Se fôssemos representar esta rede fictícia da mesma forma que na Figura 5.3, todas as posições do vetor solução teriam o valor 1.

Uma vez encontrado o caminho mínimo para a carga de maior peso, atribui-se o valor zero para a distância em cada um dos arcos que compõem o caminho mínimo. Em seguida, encontra-se o novo caminho mínimo para a carga de maior peso ainda não examinada, considerando-se esse grafo com distâncias nos arcos modificados. O procedimento é repetido até que todas as cargas tenham sido examinadas, em ordem decrescente de peso total.

Nesta etapa o peso de cada uma das cargas não é levado em consideração, pois primeiramente são definidas as ligações a serem abertas entre os terminais da rede e, em seguida, conforme descrito na etapa seguinte, é realizado o cálculo da quantidade e tipo de veículo utilizado em cada uma dessas ligações.

O objetivo da atribuição de custo igual a zero é incentivar novas cargas a utilizarem os mesmos arcos já abertos por cargas previamente avaliadas, resultando assim em um menor custo total da solução.

Esse procedimento é executado até que todas as cargas tiverem as rotas definidas para chegarem ao seu destino, sendo em seguida calculado o custo da solução inicial. O algoritmo do caminho mínimo utilizado foi o mesmo apresentado em Gallo e Pallottino (1988) baseado no método de Dijkstra com a fila de prioridades, sendo implementada computacionalmente como um *heap* binário.

Assim, como resultado parcial da geração de uma solução inicial para o problema, obtem-se um vetor solução conforme a Figura 5.3, que indica quais arcos foram selecionados, e uma estrutura de dados contendo os arcos utilizados por cada carga para chegar até o seu terminal de destino.

Note-se que no vetor solução são representados somente quais arcos serão utilizados na solução, não apresentando nenhuma indicação da quantidade e tipo de veículo utilizado em cada arco.

Tendo como dados: (i) a lista de cargas que utilizam cada um dos arcos abertos, resultante da geração da solução inicial;.(ii) o peso de cada uma dessas cargas; e (iii) a capacidade dos veículos disponíveis para operação; e com base na somatória total de peso que cada arco necessita transportar, é realizado então o cálculo da quantidade de veículos a serem utilizados.

Dado que cada arco transporta uma ou mais cargas, que podem ser alocadas a um ou mais veículos, a ordem com que as cargas são selecionadas para serem alocadas nos veículos é de extrema importância, e esse subproblema é por si só muito difícil de ser resolvido. Em suma, esse problema consiste em determinar o número de veículos que represente a solução de mínimo custo, e quais cargas alocar a cada veículo. Esse problema pode ser modelado como um problema conhecido na literatura de *bin packing problem* (BPP).

O problema de *bin-packing* (BPP) pode ser descrito da seguinte forma: dados j = 1,..., n objetos ou cargas com seus respectivos pesos w_j , e i = 1,..., n *bins* (ou veículos) idênticos de capacidade finita c, determinar a alocação (ou designação) das n cargas aos veículos (sendo $w_j \leq c$, $\forall j$), de tal modo que o número de veículos utilizados seja mínimo, e respeitando-se as restrições de capacidade em cada um dos veículos. O BPP é *NP-hard* (GAREY;JOHNSON, 1979).

No caso especial da determinação de uma solução inicial para o problema do carregamento de cargas parceladas em uma rede de transporte, com T tipos

diferentes de veículos (em geral, *T*=2, como visto anteriormente), a quantidade de veículos utilizados de cada tipo é armazenada em um vetor específico, sendo que cada posição desse vetor corresponde à quantidade de veículos utilizados em cada arco da solução.

Um exemplo pode ser visto na Figura 5.4. Nesta figura tem-se o vetor solução e, conforme já apresentado, as posições desse vetor com valor igual a 1 sinalizam que o respectivo arco foi inserido na solução. Supondo dois tipos de veículo, T1 e T2, têm-se, de acordo com a referida figura, que o arco de número 2 será operado por dois veículos, sendo um de cada tipo (T1 e T2). Já no arco *m-1* serão utilizados três veículos, sendo dois veículos do tipo T1 e um veículo do tipo T2.

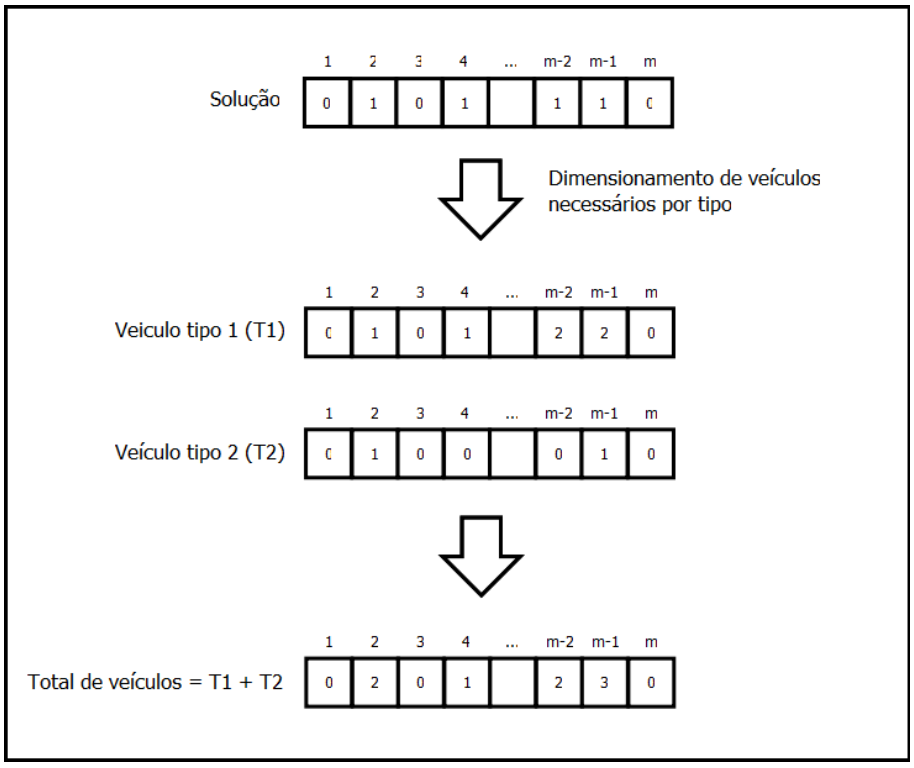

Figura 5.4: Definição da quantidade de veículos em cada arco

Desta forma, dado que o BPP é NP-hard, o mesmo é utilizado para determinar o menor número de veículos para cada arco aberto na solução, utilizando também uma heurística baseada em busca tabu. Para representar a solução do BPP é criado um vetor de tamanho igual à quantidade de cargas que utilizam um determinado arco e, em cada posição desse vetor, são colocados os índices das cargas. Deve-se

notar que essa heurística baseada em busca tabu para o BPP é executada somente se a somatória do peso das cargas que utilizam o arco for maior que a capacidade do menor veículo do problema. Observa-se, na prática, que para muitos arcos selecionados na solução inicial, não é necessária a solução do BPP.

Na Figura 5.5 é apresentado um exemplo para a definição da quantidade de veículos utilizados em um dado arco. Conforme o exemplo, o arco é utilizado por quatro cargas, sendo três delas com 8.000 kg de peso a ser transportado, e a última com 12.000 kg. Existem dois tipos de veículos disponíveis: carretas e trucks, possuindo 24.000 kg e 12.000 kg de capacidade em peso, respectivamente.

Na geração da solução inicial, a primeira etapa consiste na ordenação das cargas por ordem decrescente de peso. Em seguida é alocado primeiramente o veículo de maior capacidade (no caso a carreta), e as cargas são designadas seqüencialmente aos veículos de acordo com o vetor solução, até atingir a sua capacidade limite. As decisões de escolha da ordenação das cargas em ordem decrescente de peso, e também a decisão de se iniciar a alocação com o veículo de maior capacidade, foram tomadas com base nos experimentos computacionais realizados que apresentaram melhores soluções para problemas de grande porte.

Dado que esse procedimento para dimensionamento da quantidade de veículos foi realizado em cada arco da rede, e a cada iteração da busca tabu, o objetivo foi criar uma heurística simples, e que pudesse obter resultados satisfatórios em um curto tempo de processamento.

De acordo com o problema exemplo apresentado na Figura 5.5, após a ordenação das cargas em ordem decrescente, e utilizando primeiramente carretas, uma solução inicial é gerada utilizando-se duas carretas. Após o movimento de troca de posições do vetor solução, a nova solução gerada necessita de uma carreta e um truck para transportar as mesmas cargas no arco, resultando em um custo menor.

Ainda de acordo com o exemplo, as posições 1 e 4 do vetor solução são feitas “tabu”, não permitindo que a mesma alteração seja realizada por um determinado número fixo de iterações.

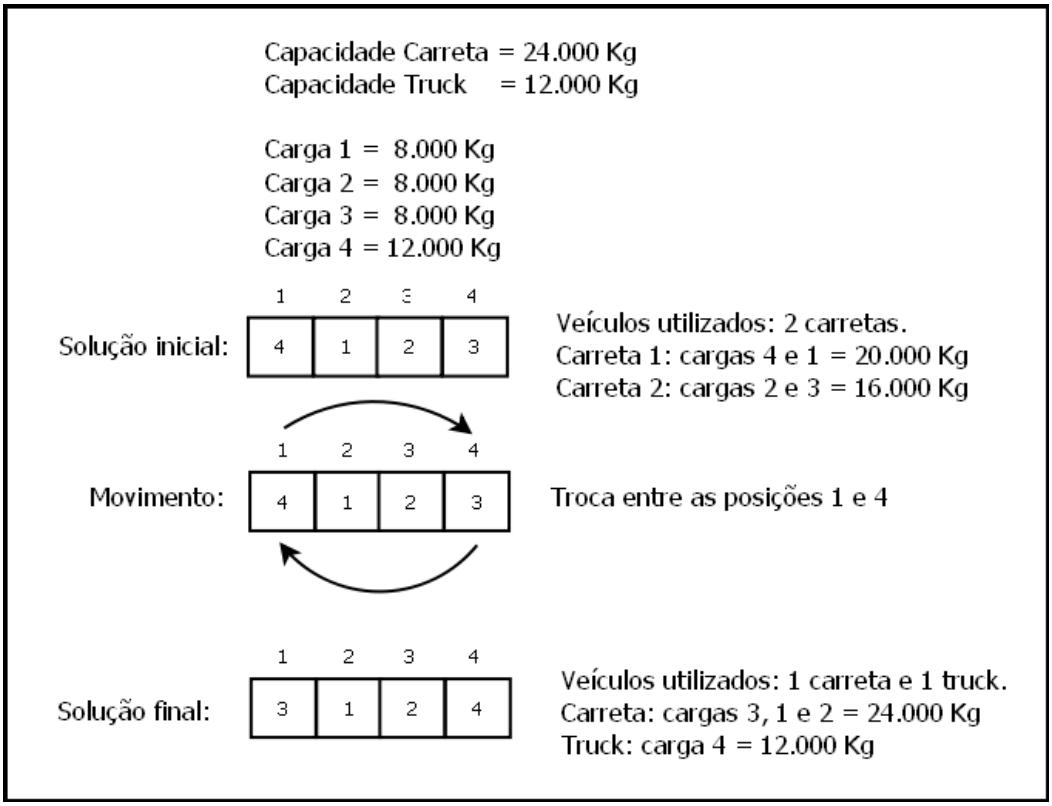

Figura 5.5: Exemplo de definição da quantidade de veículos em cada arco

Uma vez definida a quantidade e tipo de veículos alocados em cada arco aberto na rede, o cálculo do custo total da solução se torna trivial, conforme a função objetivo representada pela equação (5.1) apresentada no modelo matemático para o problema.

5.2.2 Busca tabu para a criação da rede

A descrição formal da heurística de melhoria baseada em busca tabu é apresentada na Figura 5.6. Uma vez determinada uma solução inicial contendo a configuração inicial da rede e seu custo, conforme descrito anteriormente, o próximo passo consiste na execução da rotina principal da heurística busca tabu, que primeiramente visa encontrar movimentos que modificam a solução corrente com o objetivo de se obter uma solução final de melhor qualidade. O movimento básico utilizado é o de inclusão e exclusão de arcos da solução, sendo permitidos somente movimentos que explorem o espaço de soluções viáveis do problema.

Em linhas gerais, a cada iteração da busca tabu, a heurística consiste em primeiramente selecionar a modificação na solução corrente a ser realizada, ou melhor, um movimento a ser executado, entendendo como movimento a inclusão ou exclusão de um determinado arco da rede. Todos os arcos existentes, inclusos ou não na solução corrente, são testados sequencialmente, sendo escolhido o arco que resulta na obtenção de uma solução de melhor qualidade ao se comparar com a solução corrente. O arco selecionado para movimento é feito tabu por uma quantidade fixa de iterações.

A cada avaliação de movimento a ser realizado é calculado o custo da nova solução, que consiste em: (i) definir o caminho que cada carga deve percorrer, utilizando o algoritmo do caminho mínino, armazenando os arcos utilizados por cada carga para chegar até o seu destino e, em seguida, (ii) resolver o BPP para dimensionar a quantidade de veículos (por tipo) utilizado em cada arco da rede, tendo como entrada as cargas que utilizarão cada um dos arcos dessa nova solução.

É explorado somente o espaço de soluções viáveis do problema, entendendo como uma solução viável aquela que possui uma quantidade definida de arcos abertos, de tal forma a possibilitar que, com esse conjunto de arcos, todas as cargas possam ser transportadas desde o seu terminal de origem até o seu terminal de destino (existe um caminho possível a ser percorrido). Sendo assim, um movimento não é escolhido para ser realizado se resultar em uma solução inviável para o problema.

O “critério de aspiração”, que faz parte de toda implementação da heurística busca tabu, consistiu para esse problema em, se um dado arco não pode ser modificado na solução corrente por ser “tabu”, mas sua alteração melhora a melhor solução obtida até o momento, esse movimento, apesar de tabu, é executado.

Dois mecanismos simples são utilizados para tentar evitar que a heurística fique presa em um ótimo local. O primeiro mecanismo implementado tem como referência o trabalho de Battiti e Tecchiolli (1994), que desenvolveram uma busca tabu reativa (*reactive tabu search*).

A busca tabu reativa possui um mecanismo de memória que tem como objetivo armazenar soluções previamente visitadas, e dinamicamente alterar os parâmetros da busca tabu para tentar explorar diferentes regiões do espaço de soluções do problema.

Neste trabalho é utilizado um mecanismo simples que armazena o valor da função objetivo, e em qual iteração da busca tabu ela foi encontrada, permitindo com isso que o parâmetro que define a quantidade de iterações que um movimento permanece tabu (conhecido como duração tabu ou *tabu tenure*), seja alterado dinamicamente a fim de tentar inibir a visita de uma mesma solução, em um dado intervalo mínimo de iterações.

O segundo mecanismo utilizado para se explorar diferentes regiões do espaço de soluções é baseado no conceito que alguns autores denominam por diversificação forte (*strong diversification*) (DELL'AMICO; LODI; MAFFIOLI, 1999; STECCO; CORDEAU; MORETTI, 2009), e consiste na geração de uma nova solução inicial, escolhendo-se aleatoriamente arcos a serem inseridos na solução, até que esta se torne viável. Esse procedimento é aplicado após um número fixo de iterações sem melhoria na busca tabu.

Heurística *Busca Tabu*
Início
Ler dados de entrada
Criar uma lista $\mathcal{L}$ de tamanho igual a cardinalidade do conjunto A de rotas candidatas
Para cada posição i de $\mathcal{L}$ **Faça**
  $\mathcal{L}[i] = 0$
Fim Para cada
Ordenar as cargas em ordem decrescente de peso
Armazenar as cargas ordenadas em uma estrutuda de dados do tipo pilha $\mathcal{P}$
mantendo a carga de maior peso no topo da pilha.
Criar uma rede artificial $\mathcal{R}$ contendo todas as rotas candidatas contidas no conjunto A
Enquanto houver elementos na pilha $\mathcal{P}$ **Faça**
  Remova a carga c do topo da pilha $\mathcal{P}$
  Calcular o caminho mínimo desde a origem até o destino da carga, na rede $\mathcal{R}$
  Inserir as rotas utilizadas pela carga c na solução inicial do problema
  (equivalente a $\mathcal{L}[i] = 1$ para todos as rotas i utilizadas pela carga c)
  Definir custo igual a zero para as rotas utilizadas pela carga c na rede $\mathcal{R}$
Fim Enquanto
Calcular o custo da solução inicial
Enquanto a quantidade máxima de iterações da tabu não for alcançada **Faça**
  Para cada posição i da lista $\mathcal{L}$ **Faça**
    $\mathcal{L}[i] = 1 - \mathcal{L}[i]$
    Verificar viabilidade da solução
    Se solução inviável **Então**
      desfazer o movimento ($\mathcal{L}[i] = 1 - \mathcal{L}[i]$)
      testar a próxima rota ($i = i + 1$)
    Senão
      Calcular o custo da solução após o movimento
      Armazenar a rota alterada no movimento corrente
      Se a rota alterada não é tabu **ou** solução corrente $<$ melhor solução **Então**
        Se solução corrente $<$ melhor movimento **Então**
        Gravar como melhor movimento a ser executado
        Grava rota a ser alterada
        Fim Se
      Fim Se
    Fim Se
  Fim Para cada
  Se foi possível encontrar movimento viável **Então**
    Executar melhor movimento encontrado
    Fazer rota alterada como tabu
    Se o custo do melhor movimento encontrado for igual a melhor solução corrente **Então**
      aumentar a quantidade que um dado movimento permanece tabu em λ iterações
    Fim Se
    Se não foi possível encontrar um movimento que melhore a solução corrente
      após uma quantidade fixa de iterações **Então**
      Gerar uma nova solução inicial aleatória
    Fim Se
  Fim Se
Fim Enquanto
Fim Heurística *Busca Tabu*

Figura 5.6: Esqueleto básico da busca tabu para solução do problema

5.3 PROBLEMA EXEMPLO

A fim de facilitar o entendimento da modelagem e da solução do problema de carregamento de carga parcelada em uma rede de transporte, é apresentado nessa seção um exemplo de pequeno porte composto por 16 terminais, 14 cargas a serem transportadas, 89 arcos disponíveis para utilização, e dois tipos distintos de veículos: carreta e truck.

Os municípios envolvidos nesse exemplo estão relacionados na Tabela 5.1, juntamente com seu respectivo número e sigla, que serão utilizados para facilitar a representação do problema. Na Tabela 5.2 são apresentadas as cargas a serem transportadas, contendo para cada uma delas a origem, destino e peso. Os arcos disponíveis na rede para serem utilizados como rotas estão apresentados na Tabela 5.3, que contêm para cada arco, a origem, o destino, a distância entre terminais e ainda o custo tanto para o truck como para a carreta.

Tabela 5.1: Terminais envolvidos no problema exemplo

Número	Sigla	Município
1	CON	Contagem - MG
2	MCL	Montes Claros - MG
3	SSA	Salvador - BA
4	RIB	Rio Bonito - RJ
5	QBN	Barra Mansa - RJ
6	RIO	Rio de Janeiro - RJ
7	RAO	Ribeirão Preto - SP
8	VIX	Vitória - ES
9	PPY	Pouso Alegre - MG
10	SJK	São José dos Campos - SP
11	BAU	Bauru - SP
12	NVF	Nova Friburgo - RJ
13	CTN	Colatina - ES
14	SAO	São Paulo - SP
15	QCF	Birigüi - SP
16	CPQ	Campinas - SP

Para esse exemplo os *hubs* estão localizados nos terminais de São Paulo, Contagem, Vitória, Campinas, Salvador e Rio de Janeiro. Os arcos candidatos foram criados conforme as cinco possibilidades de rotas a serem operadas em uma rede

hub-and-spoke, conforme descrito no início desse capítulo. Os terminais envolvidos na criação dos arcos candidatos são todos aqueles (*hubs* ou não) que são origem de uma ou mais cargas. As alternativas para a criação de rotas são:

(i) carregamento direto entre os terminais de origem e destino;

(ii) carregamento via *hub* onde o terminal de origem está alocado(*hub* origem);

(iii) carregamento via *hub* onde o terminal de destino está alocado (*hub* destino);

(iv) carregamento típico *hub-and-spoke*, sendo a carga transferida do terminal de origem para o *hub* origem, do *hub* origem para o *hub* destino, e finalmente do *hub* destino para o terminal de destino da carga;

(v) via um *hub* qualquer da rede.

A rede *hub-and-spoke* para o problema exemplo está representada na Figura 5.7. Os nós de cor vermelha representam os *hubs* da rede e os arcos também na cor vermelha representam a ligação entre *hubs*. Os nós e arcos de cor azul representam os terminais não-*hub*, e a ligação *hub* – terminal, respectivamente.

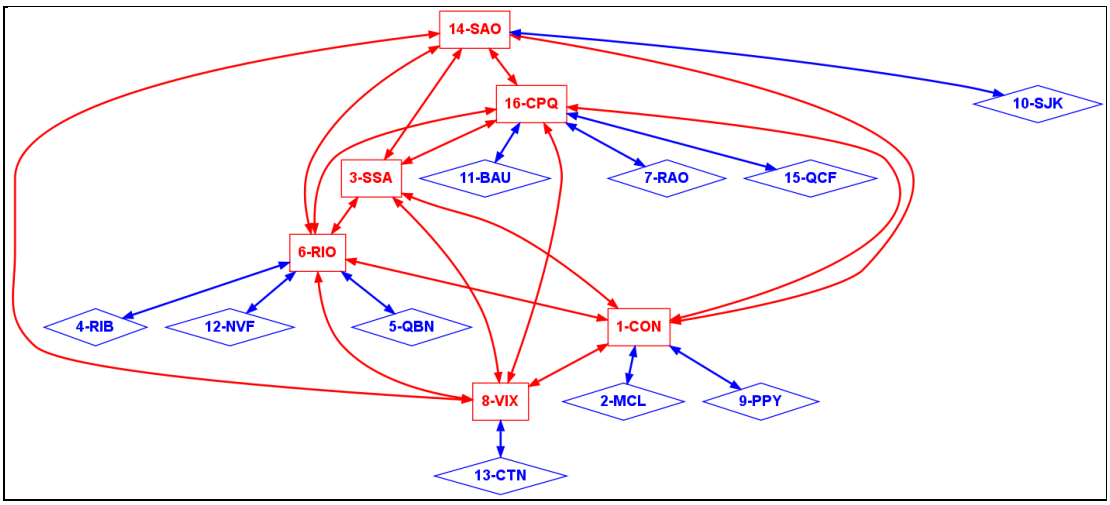

Figura 5.7: Rede *hub-and-spoke* para o problema exemplo

Dois tipos de veículo são considerados nesse exemplo: truck e carreta. A capacidade disponível para a carreta foi fixada em 24.000 kg por veículo com um custo de $ 1,40 / km. Já para o truck, a capacidade de cada veículo foi fixada em 12.000 kg, e custo de $ 1,00 / km. É importante relembrar que os custos dos veículos não dependem da quantidade de carga transportada nos mesmos. Com

base nesses custos unitários, é possível determinar o custo de cada tipo de veículo em cada um dos arcos, como mostrado na Tabela 5.3.

Uma representação gráfica das demandas, e das rotas disponíveis na rede, é apresentada nas Figura 5.8 e 5.9, respectivamente. Os arcos de cor azul da Figura 5.8 representam os arcos disponíveis para serem utilizados na criação de rotas, tendo como atributo a distância em quilômetros entre os terminais. Já os arcos de cor vermelha da Figura 5.9 representam a demanda de cargas a serem transportadas, com suas respectivas origem e destino, tendo como atributo desses arcos o peso (em kg) a ser transportado e, entre parênteses, o número identificador da carga.

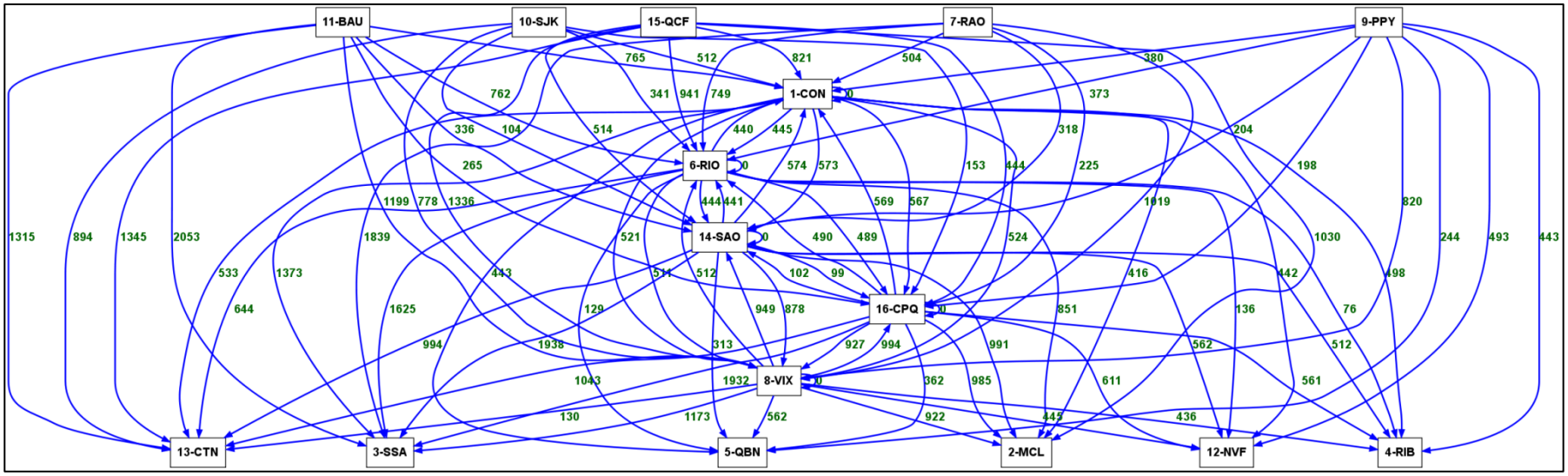

Figura 5.8: Representação gráfica dos arcos disponíveis do problema exemplo

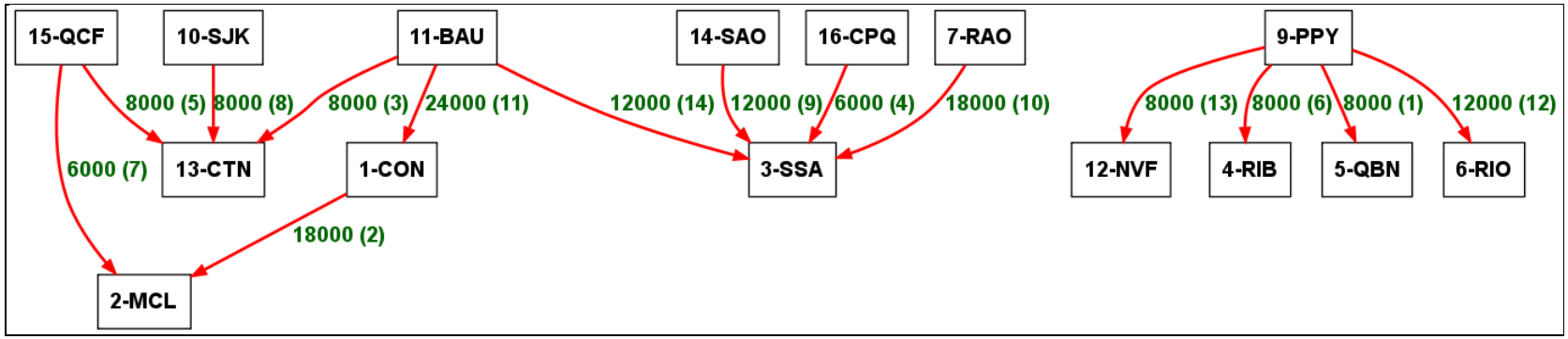

Figura 5.9: Demanda de cargas a serem transportadas no problema exemplo

Tabela 5.2: Cargas a serem transportadas

Carga nº	Origem	Destino	Peso (Kg)
1	9-PPY	5-QBN	8.000
2	1-CON	2-MCL	18.000
3	11-BAU	13-CTN	8.000
4	16-CPQ	3-SSA	6.000
5	15-QCF	13-CTN	8.000
6	9-PPY	4-RIB	8.000
7	15-QCF	2-MCL	6.000
8	10-SJK	13-CTN	8.000
9	14-SAO	3-SSA	12.000
10	7-RAO	3-SSA	18.000
11	11-BAU	1-CON	24.000
12	9-PPY	6-RIO	12.000
13	9-PPY	12-NVF	8.000
14	11-BAU	3-SSA	12.000

Com os dados aqui apresentados, e utilizando o modelo matemático apresentado na Seção 5.1, foi construído o modelo matemático correspondente utilizando-se a linguagem algébrica AMPL com a sintaxe descrita em Fourer, Gay e Kernighan (1990). O modelo matemático em AMPL, juntamente com os dados de entrada utilizados para representar o problema exemplo, podem ser vistos nas Figura 5.10 e Figura 5.11, respectivamente.

Existem diversos pacotes comerciais que possuem interface para solução de problemas de otimização, representados utilizando a linguagem AMPL, como por exemplo, o CPLEX, o Gurobi, o Mosek, o XpressMP, entre outros. O problema exemplo aqui apresentado pôde ser facilmente resolvido, devido ao seu pequeno porte, utilizando o CPLEX.

Tabela 5.3: Arcos disponíveis para transporte

n°	Origem	Destino	Distância (Km)	Custo ($) Truck	Carreta
1	1-CON	12-NVF	442	442,00	618,8
2	1-CON	6-RIO	445	445,00	623,00
3	1-CON	14-SAO	573	573,00	802,20
4	1-CON	5-QBN	443	443,00	620,20
5	1-CON	2-MCL	416	416,00	582,40
6	1-CON	13-CTN	533	533,00	746,20
7	1-CON	3-SSA	1373	1.373,00	1.922,20
8	1-CON	8-VIX	524	524,00	733,60
9	1-CON	16-CPQ	567	567,00	793,80
10	1-CON	4-RIB	498	498,00	697,20
11	1-CON	1-CON	0	0,00	0,00
12	6-RIO	2-MCL	851	851,00	1.191,40
13	6-RIO	4-RIB	76	76,00	106,40
14	6-RIO	13-CTN	644	644,00	901,60
15	6-RIO	3-SSA	1625	1.625,00	2.275,00
16	6-RIO	1-CON	440	440,00	616,00
17	6-RIO	16-CPQ	489	489,00	684,60
18	6-RIO	14-SAO	444	444,00	621,60
19	6-RIO	6-RIO	0	0,00	0,00
20	6-RIO	5-QBN	129	129,00	180,60
21	6-RIO	12-NVF	136	136,00	190,40
22	6-RIO	8-VIX	511	511,00	715,40
23	7-RAO	14-SAO	318	318,00	445,20
24	7-RAO	3-SSA	1839	1.839,00	2.574,60
25	7-RAO	8-VIX	1019	1.019,00	1.426,60
26	7-RAO	1-CON	504	504,00	705,60
27	7-RAO	16-CPQ	225	225,00	315,00
28	7-RAO	6-RIO	749	749,00	1.048,60
29	8-VIX	1-CON	521	521,00	729,40
30	8-VIX	5-QBN	562	562,00	786,80
31	8-VIX	16-CPQ	994	994,00	1.391,60
32	8-VIX	13-CTN	130	130,00	182,00
33	8-VIX	2-MCL	922	922,00	1.290,80
34	8-VIX	3-SSA	1173	1.173,00	1.642,20
35	8-VIX	12-NVF	445	445,00	623,00
36	8-VIX	8-VIX	0	0,00	0,00
37	8-VIX	4-RIB	436	436,00	610,40
38	8-VIX	14-SAO	949	949,00	1.328,60
39	8-VIX	6-RIO	512	512,00	716,80
40	9-PPY	14-SAO	204	204,00	285,60
41	9-PPY	5-QBN	244	244,00	341,60
42	9-PPY	1-CON	380	380,00	532,00
43	9-PPY	12-NVF	493	493,00	690,20
44	9-PPY	8-VIX	820	820,00	1.148,00
45	9-PPY	6-RIO	373	373,00	522,20
46	9-PPY	4-RIB	443	443,00	620,20
47	9-PPY	16-CPQ	198	198,00	277,20
48	10-SJK	16-CPQ	153	153,00	214,20
49	10-SJK	13-CTN	894	894,00	1.251,60
50	10-SJK	6-RIO	341	341,00	477,40
51	10-SJK	14-SAO	104	104,00	145,60
52	10-SJK	8-VIX	778	778,00	1.089,20
53	10-SJK	1-CON	512	512,00	716,80
54	11-BAU	6-RIO	762	762,00	1.066,80
55	11-BAU	8-VIX	1199	1.199,00	1.678,60
56	11-BAU	3-SSA	2053	2.053,00	2.874,20
57	11-BAU	13-CTN	1315	1.315,00	1.841,00
58	11-BAU	14-SAO	336	336,00	470,40
59	11-BAU	1-CON	765	765,00	1.071,00
60	11-BAU	16-CPQ	265	265,00	371,00
61	14-SAO	6-RIO	441	441,00	617,40
62	14-SAO	1-CON	574	574,00	803,60
63	14-SAO	16-CPQ	99	99,00	138,60
64	14-SAO	5-QBN	313	313,00	438,20
65	14-SAO	4-RIB	512	512,00	716,80
66	14-SAO	3-SSA	1938	1.938,00	2.713,20
67	14-SAO	8-VIX	878	878,00	1.229,20
68	14-SAO	14-SAO	0	0,00	0,00
69	14-SAO	13-CTN	994	994,00	1.391,60
70	14-SAO	12-NVF	562	562,00	786,80
71	14-SAO	2-MCL	991	991,00	1.387,40
72	15-QCF	8-VIX	1336	1.336,00	1.870,40
73	15-QCF	2-MCL	1030	1.030,00	1.442,00
74	15-QCF	14-SAO	514	514,00	719,60
75	15-QCF	13-CTN	1345	1.345,00	1.883,00
76	15-QCF	16-CPQ	444	444,00	621,60
77	15-QCF	1-CON	821	821,00	1.149,40
78	15-QCF	6-RIO	941	941,00	1.317,40
79	16-CPQ	1-CON	569	569,00	796,60
80	16-CPQ	2-MCL	985	985,00	1.379,00
81	16-CPQ	13-CTN	1043	1.043,00	1.460,20
82	16-CPQ	5-QBN	362	362,00	506,80
83	16-CPQ	6-RIO	490	490,00	686,00
84	16-CPQ	14-SAO	102	102,00	142,80
85	16-CPQ	3-SSA	1932	1.932,00	2.704,80
86	16-CPQ	12-NVF	611	611,00	855,40
87	16-CPQ	8-VIX	927	927,00	1.297,80
88	16-CPQ	4-RIB	561	561,00	785,40
89	16-CPQ	16-CPQ	0	0,00	0,00

```
param n, integer, >= 2;  # quantidade de nós da rede
set V, default {1..n};   # conjunto de nós da rede
set A, within V cross V; # conjunto de arcos da rede
set K;                   # conjunto de cargas a serem transportadas
set T;                   # conjunto de tipos de veiculos disponiveis

param O {k in K} ;       # nó de origem da carga k
param D {k in K} ;       # nó de destino da carga k
param R {k in K} >= 0;   # peso da carga k
param U {t in T}, >= 0;  # capacidade do veículo do tipo t
param c {t in T}, >= 0;  # custo por km de um veículo do tipo t
param distancia {(i,j) in A}; # distância de i para j.

var f {(i,j) in A, k in K, t in T}, binary;
# =1 se um veíc. do tipo t é utilizado para transportar a carga k no arco ij

var x {(i,j) in A, t in T}, integer, >= 0; # qtd. veíc. do tipo t no arco ij

minimize z : sum{t in T} sum{(i,j) in A} distancia[i,j] * c[t] * x[i,j,t];

s.t. r1 {i in V, k in K} : sum{t in T} ( sum{j in V:(i,j) in A} f[i,j,k,t]
                    - sum{l in V: (l,i) in A} f[l,i,k,t] )
                    = (if i=O[k] then 1 else (if i=D[k] then -1 else 0));


s.t. r2 {(i,j) in A, t in T} :
                             sum{k in K} f[i,j,k,t] * R[k] <= U[t] * x[i,j,t];

s.t. r3 {(i,j) in A, k in K} : sum{t in T} f[i,j,k,t] <= 1;

s.t. r4 {(i,j) in A, k in K, t in T} : f[i,j,k,t] * R[k] <= U[t];
end;
```

Figura 5.10: Modelo matemático em AMPL para o problema exemplo

A solução do problema exemplo é apresentada na Tabela 5.4, e também de forma gráfica na Figura 5.12. Na solução final são alocados 16 veículos para operação, sendo 10 carretas e 6 trucks. A rota partindo de Contagem para Salvador utiliza duas carretas, e a rota de Pouso Alegre para o Rio de Janeiro utiliza uma carreta e um truck. As demais rotas são operadas por um único veículo.

As rotas representadas de cor azul na Figura 5.12 possuem um atributo (em verde e vermelho) que diz respeito ao peso total transportado (em kg), o tipo de veículo utilizado e, entre parênteses, o número indicativo de quais cargas estão sendo transportadas em cada rota.

```
data;
param n := 16;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16;
set A := 15 8, 10 16, 16 1, 8 1, 1 12, 15 2, 10 13, 7 14, 6 2, 1 6, 14 6, 14 1, 11
6, 14 16, 10 6, 7 3, 9 14, 16 2, 9 5, 8 5, 16 13, 14 5, 16 5, 10 14, 15 14, 14 4,
15 13, 15 16, 6 4, 10 8, 14 3, 1 14, 15 1, 11 8, 1 5, 11 3, 7 8, 1 2, 11 13, 16 6,
1 13, 7 1, 8 16, 1 3, 6 13, 10 1, 8 13, 6 3, 16 14, 14 8, 6 1, 16 3, 8 2, 16 12, 6
16, 8 3, 14 14, 7 16, 16 8, 9 1, 14 13, 7 6, 11 14, 8 12, 1 8, 15 6, 11 1, 8 8, 14
12, 9 12, 8 4, 6 14, 9 8, 6 6, 1 16, 6 5, 11 16, 8 14, 9 6, 16 4, 6 12, 16 16, 1 4,
9 4, 1 1, 8 6, 9 16, 6 8, 14 2;
param: K:    R     O     D:=
1      8000   9      5
2      18000  1      2
3      8000   11     13
4      6000   16     3
5      8000   15     13
6      8000   9      4
7      6000   15     2
8      8000   10     13
9      12000  14     3
10     18000  7      3
11     24000  11     1
12     12000  9      6
13     8000   9      12
14     12000  11     3;
param distancia default 0 := [15, 8] 1336 [10, 16] 153  [16, 1] 569 [8, 1] 521 [1,
12] 442 [15, 2] 1030 [10, 13] 894 [7, 14] 318 [6, 2] 851 [1, 6] 445 [14, 6] 441
[14, 1] 574 [11, 6] 762 [14, 16] 99 [10, 6] 341 [7, 3] 1839 [9, 14] 204 [16, 2] 985
[9, 5] 244 [8, 5] 562 [16, 13] 1043 [14, 5] 313 [16, 5] 362 [10, 14] 104 [15, 14]
514 [14, 4] 512 [15, 13] 1345 [15, 16] 444 [6, 4] 76 [10, 8] 778 [14, 3] 1938 [1,
14] 573 [15, 1] 821 [11, 8] 1199 [1, 5] 443 [11, 3] 2053 [7, 8] 1019 [1, 2] 416
[11, 13] 1315 [16, 6] 490 [1, 13] 533 [7, 1] 504 [8, 16] 994 [1, 3] 1373 [6, 13]
644 [10, 1] 512 [8, 13] 130 [6, 3] 1625 [16, 14] 102 [14, 8] 878 [6, 1] 440 [16, 3]
1932 [8, 2] 922 [16, 12] 611 [6, 16] 489 [8, 3] 1173 [14, 14] 0 [7, 16] 225 [16, 8]
927 [9, 1] 380 [14, 13] 994 [7, 6] 749 [11, 14] 336 [8, 12] 445 [1, 8] 524 [15, 6]
941 [11, 1] 765 [8, 8] 0 [14, 12] 562 [9, 12] 493 [8, 4] 436 [6, 14] 444 [9, 8] 820
[6, 6] 0 [1, 16] 567 [6, 5] 129 [11, 16] 265 [8, 14] 949 [9, 6] 373 [16, 4] 561 [6,
12] 136 [16, 16] 0 [1, 4] 498 [9, 4] 443 [1, 1] 0 [8, 6] 512 [9, 16] 198 [6, 8] 511
[14, 2] 991 ;
end;
```

Figura 5.11: Dados de entrada em AMPL do problema exemplo

Por exemplo, na Figura 5.12, que apresenta a solução do problema exemplo, entre Pouso Alegre-MG (sigla PPY, número 9), e Rio de Janeiro-RJ (sigla RIO, número 6), são utilizados dois veículos: um veículo do tipo truck transportando somente a carga número 12 contendo 12.000 kg, e o segundo veículo do tipo carreta, transportando 24.000 kg, que corresponde à soma em peso das cargas número 1, 6 e 13, possuindo cada uma delas 8.000 kg, destinadas respectivamente a Barra Mansa (5-QBN), Rio Bonito-RJ (4-RIB) e Nova Friburgo-RJ (12-NVF).

Como a carga 12 é transportada sozinha em um truck e possui destino final o próprio terminal do Rio de Janeiro, o veículo é descarregado e a carga preparada para a entrega local de todos os despachos. Já as cargas 1, 6 e 13, transportadas na carreta, possuem como destino municípios que são atendidos por rotas individuais partindo do Rio de Janeiro, e, portanto, essa carreta que fez o transporte partindo de Pouso Alegre deve ser completamente descarregada, e as cargas reclassificadas e preparadas para seguirem, em carregamentos individuais utilizando veículos trucks, até os seus terminais de destino finais.

Tabela 5.4: Solução do problema exemplo

Veículo nº	Origem	Destino	Veículo Tipo	Peso Total (Kg)	Cargas
1	11-BAU	16-CPQ	Carreta	20000	(3, 14)
2	1-CON	2-MCL	Carreta	24000	(2, 7)
3	15-QCF	16-CPQ	Carreta	14000	(5, 7)
4	11-BAU	1-CON	Carreta	24000	(11)
5	7-RAO	1-CON	Carreta	18000	(10)
6	16-CPQ	1-CON	Carreta	24000	(4, 7, 14)
7	1-CON	3-SSA	Carreta	24000	(4, 10)
8	1-CON	3-SSA	Carreta	24000	(9, 14)
9	16-CPQ	13-CTN	Carreta	24000	(3, 5, 8)
10	9-PPY	6-RIO	Carreta	24000	(1, 6, 13)
11	9-PPY	6-RIO	Truck	12000	(12)
12	6-RIO	5-QBN	Truck	8000	(1)
13	6-RIO	4-RIB	Truck	8000	(6)
14	10-SJK	16-CPQ	Truck	8000	(8)
15	14-SAO	1-CON	Truck	12000	(9)
16	6-RIO	12-NVF	Truck	8000	(13)

Em linhas gerais, a solução obtida apresentou: (i) rotas diretas entre terminais; (ii) rotas com uma única parada em um hub para consolidação; e (iii) rotas tipicamente *hub-and-spoke* contendo duas paradas intermediárias em dois *hubs* distintos para consolidação.

Um exemplo de rota direta é o caso da carga número 11, que corresponde a 24.000 kg de Bauru para Contagem. A quantidade de carga transportada justificou o envio direto de uma carreta para o terminal de destino, sem paradas intermediárias.

Já as cargas 3, 5 e 8, com origens Bauru, Birigüi e São José dos Campos, respectivamente, possuem cada uma delas 8.000Kg para serem transportados para Colatina. De acordo com a solução do problema, a melhor estratégia, do ponto de vista de custo, é consolidar estas três cargas no *hub* localizado em Campinas para

em seguida enviar um único carregamento direto, utilizando uma carreta, para Colatina.

Esse é um exemplo de uma rota com uma única parada em um *hub* na origem, e ainda apresenta a possibilidade de consolidação de uma carga em um *hub* diferente do qual o terminal de origem está alocado, que é o caso da carga número 8 com origem São José dos Campos, que, na configuração estrita de uma rede *hub-and-spoke*, deveria ser enviado para São Paulo, em vez de Campinas.

Um exemplo típico de rota do tipo *hub-and-spoke* é o transporte da carga de número 7 contendo 6.000 kg que deve ser transportada de Birigüi para Montes Claros. A carga 7 primeiramente é enviada, juntamente com a carga número 5, para seu *hub* de origem localizado em Campinas. Em Campinas esse veículo é descarregado, e a carga 7 consolidada com as cargas 4 e 14, é então enviada para o *hub* de destino localizado em Contagem. Em Contagem a carga 7, juntamente com a carga 2, atinge o seu destino, que corresponde ao terminal em Montes Claros.

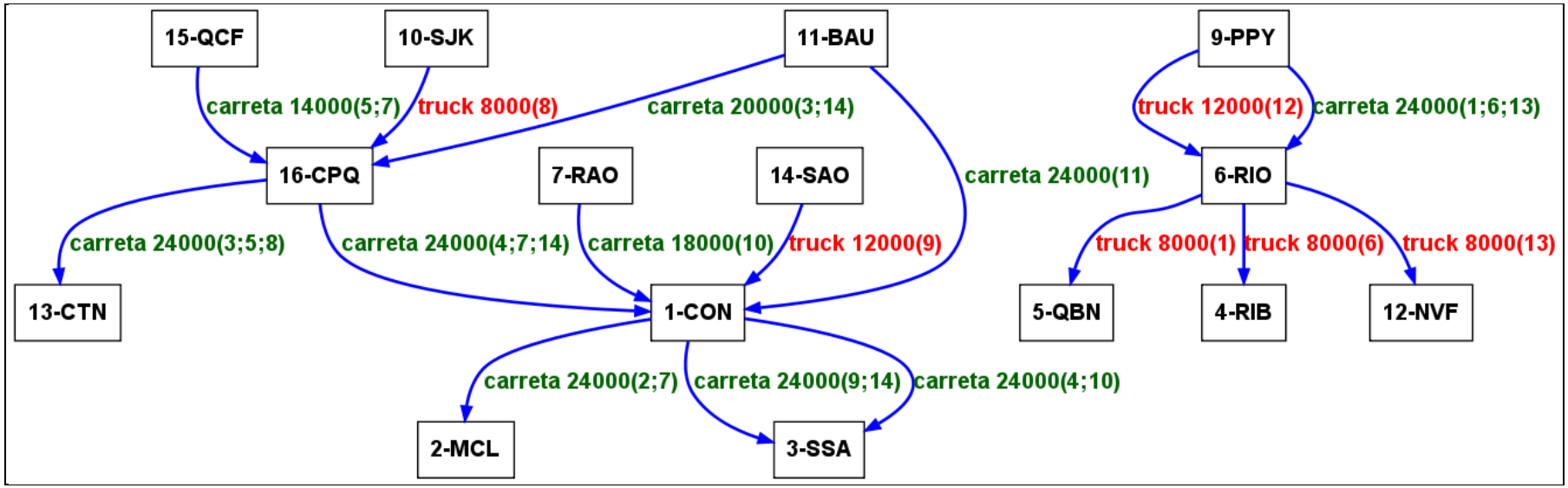

Figura 5.12: Representação gráfica da solução do problema exemplo

5.4 EXPERIMENTOS COMPUTACIONAIS

A fim de avaliar o desempenho do método de solução proposto para o problema de carregamento de carga parcelada em uma rede de transporte, foram realizados experimentos computacionais com dados reais fornecidos pela empresa Patrus Transportes Urgentes Ltda.

Ao contrário do USAHLP, não existem na literatura instâncias de teste para avaliação comparativa do desempenho da heurística proposta, e ainda, apesar das similaridades já descritas, o problema aqui tratado é de natureza diferente do NLP (*network loading problem*), pois o NLP trata da solução de problemas de rede de telecomunicações, tendo proporções de custo, volume de tráfego, e quantidade de arcos bem diferentes dos dados de uma empresa transportadora de carga parcelada, podendo levar a interpretações incorretas, caso esses dados sejam utilizados para avaliação. Sendo assim, optou-se por utilizar diretamente dados reais de uma empresa transportadora de carga parcelada.

A Patrus Transportes[2] é uma empresa especializada no transporte de carga parcelada, e atende as regiões sul e sudeste do Brasil, incluindo os estados da Bahia e Sergipe. Fundada em 1973 em Minas Gerais, a empresa possui atualmente 53 terminais, e dentre esses terminais 10 deles operam, em maior ou menor escala, como um terminal para consolidação de cargas (*hub*). A lista de terminais da Patrus Transportes pode ser vista na Tabela 5.5.

O principal *hub* fica localizado em Contagem-MG, onde está também localizada a matriz da empresa, possuindo 7.500 m² e 60 docas em *cross-docking* para carga e descarga. Na Figura 5.13 é apresentada uma foto aérea desse *hub*.

[2] Maiores de talhes sobre a Patrus Transportes podem ser obtidos no endereço: http://www.patrus.com.br/

Figura 5.13: *Hub* Patrus Transportes em Contagem-MG

Foram obtidos junto à empresa os dados do fluxo de carga parcelada de todo o ano de 2009, a fim de serem utilizados os experimentos computacionais. Esses dados foram manipulados apropriadamente e resultaram em 12 instâncias de teste, uma para cada mês do ano.

Um resumo das instâncias geradas pode ser visto na Tabela 5.6. O significado das colunas dessa tabela são:

- *Terminais*: indica a quantidade de terminais envolvidos em cada problema, correspondendo a um ponto de origem e/ou destino de carga;
- *Cargas*: a quantidade total de cargas a serem transportadas;
- *Peso (Kg)*: peso total de todas as cargas a serem transportadas;
- *Rotas*: número de rotas ou ligações disponíveis para serem utilizadas;
- *Var.* Inteiras: número de variáveis inteiras do problema, para um único tipo de veículo;
- *Var. binárias*: número de variáveis binárias do problema de programação inteira, para um único tipo de veículo;
- *Restrições*: número de restrições de cada problema, para um único tipo de veículo;
- *Não-zeros*: número de elementos diferentes de zero da matriz de restrições, no caso de um único tipo de veículo.

É possível também verificar, na Tabela 5.6, as diferenças no peso médio por carga transportada ao longo do ano, confirmando a usual sazonalidade da demanda por transporte de carga parcelada no Brasil. Analisando os dados fornecidos, observa-se no início do ano uma forte queda na demanda, tendo um peso médio por carga transportada aproximadamente 30% menor, ao se comparar com o último trimestre do ano.

Tabela 5.5: Lista de terminais da Patrus Transportes

Terminal Nº	Localidade	Terminal Nº	Localidade
1	Aracaju, SE	28	Montes Claros, MG
2	Barra Mansa, RJ	29	Nova Serrana, MG
3	Barreiras, BA	30	Novo Hamburgo, RS
4	Bauru, SP	31	Passos, MG
5	Birigüi, SP	32	Patos de Minas, MG
6	Blumenau, SC	33	Petrolina, PE
7	Cachoeiro de Itapemirim, ES	34	Pouso Alegre, MG
8	Campinas, SP	35	Presidente Prudente, SP
9	Campos, RJ	36	Ribeirão Preto, SP
10	Colatina, ES	37	Rio de Janeiro, RJ
11	Contagem, MG	38	Salvador, BA
12	Curitiba, PR	39	Santo Antônio de Jesus, BA
13	Divinópolis, MG	40	Santos, SP
14	Eunápolis, BA	41	São José do Rio Preto, SP
15	Extrema, MG	42	São José dos Campos, SP
16	Feira de Santana, BA	43	São Paulo, SP
17	Franca, SP	44	Senhor do Bonfim, BA
18	Governador Valadares, MG	45	Sete Lagoas, MG
19	Ipatinga, MG	46	Sorocaba, SP
20	Irecê, BA	47	Teixeira de Freitas, BA
21	Itabuna, BA	48	Teófilo Otoni, MG
22	Jequié, BA	49	Uberaba, MG
23	Joinville, SC	50	Uberlândia, MG
24	Juiz de Fora, MG	51	Varginha, MG
25	Londrina, PR	52	Vitória da Conquista, BA
26	Manhuaçu, MG	53	Vitória, ES
27	Marília, SP		

O experimento computacional consistiu em resolver cada um dos 12 problemas gerados, utilizando tanto o pacote de otimização CPLEX versão 9.1, como também com a heurística proposta baseada em busca tabu, a qual foi implementada computacionalmente em linguagem de programação C++. Foram considerados dois tipos de veículos para os experimentos: truck e carreta. As rotas alternativas foram

obtidas conforme descrito na seção anterior, tendo cada carga cinco alternativas básicas para envio/utilização de rotas:

(i) carregamento direto entre os terminais;
(ii) carregamento via *hub* origem;
(iii) carregamento via *hub* destino;
(iv) carregamento típico *hub-and-spoke*;
(v) via um *hub* qualquer da rede.

Na Figura 5.14 é apresentado um exemplo de uma rede contendo três terminais de origem, representados pelos nós de número 1,2 e 3; dois *hubs* representados por h1 e h2; e dois terminais de destino 4 e 5. Nesta rede estão representadas todas as rotas possíveis para os terminais de origem chegarem aos terminais de destino, admitindo que todos os terminais de origem possuam cargas a serem transportadas para ambos os terminais de destino. Para exemplificar as alternativas de percurso, seja uma carga do nó 2 para o nó 5, as alternativas seriam: (i) diretamente de 2 para 5; (ii) com uma parada em um *hub*, h1 ou h2, (iii) com uma parada em ambos os *hubs* h1 e h2, ou na ordem inversa, h2 e em seguida h1.

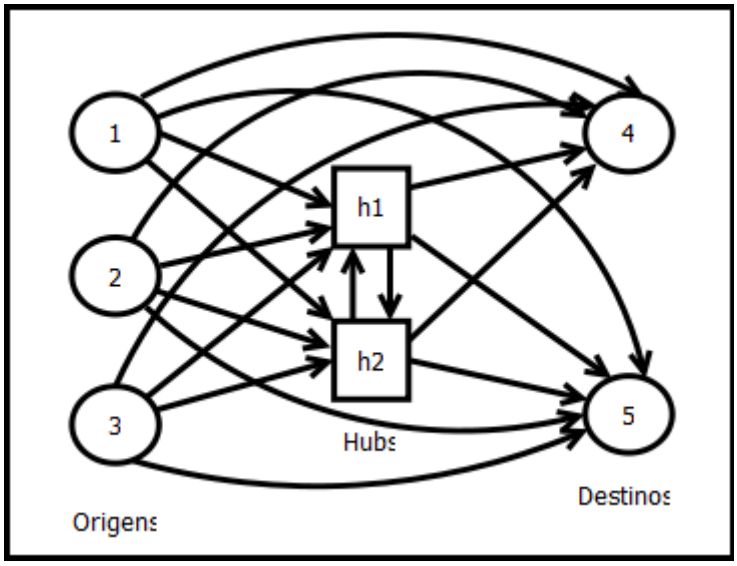

Figura 5.14: Alternativas de rotas entre os terminais da rede

O computador utilizado para os experimentos computacionais foi um laptop com sistema operacional Windows Vista, com processador Intel Core 2 Duo 2,20 GHz e 4

GB de memória RAM, sendo que somente um processador foi utilizado nos experimentos computacionais.

Tabela 5.6: Instâncias Patrus Transportes

Mês	Terminais	Cargas	Peso (Kg)	Rotas	Var. Inteiras	Var. binárias	Restrições	Não-zeros
JAN	51	195	854.929	725	725	141.375	152.045	562.325
FEV	51	198	845.244	737	737	145.926	156.761	580.481
MAR	51	191	948.822	736	736	140.576	151.053	559.220
ABR	51	203	982.572	741	741	150.423	161.517	598.373
MAI	51	192	1.009.790	731	731	140.352	150.875	558.299
JUN	51	200	1.047.962	730	730	146.000	156.930	580.730
JUL	52	200	1.011.689	756	756	151.200	162.356	601.556
AGO	52	208	1.065.274	763	763	158.704	170.283	631.419
SET	51	207	1.091.068	767	767	158.769	170.093	631.703
OUT	52	207	1.144.837	784	784	162.288	173.836	645.796
NOV	52	221	1.207.605	795	795	175.695	187.982	699.155
DEZ	53	217	1.128.746	805	805	174.685	186.991	695.205

Uma etapa usualmente trabalhosa de ser realizada na preparação de dados para solução de problemas de transporte é na preparação de uma tabela de distância entre localidades, e no caso desse problema, a distância entre terminais. No presente trabalho, o cálculo da distância entre terminais, das mais de 800 rotas disponíveis no estudo de caso, foi automatizado utilizando um programa implementado computacionalmente utilizando a linguagem de programação Perl, responsável pela interface com a ferramenta Google Maps.

O Google Maps é um serviço gratuito de pesquisa e visualização de mapas e imagens de satélite da Terra via web, fornecido e desenvolvido pela empresa americana Google. Atualmente, o serviço disponibiliza mapas e rotas para qualquer ponto nos Estados Unidos, Canadá, na União Européia, Austrália, Brasil, entre outros. E ainda disponibiliza também imagens via satélite do mundo todo, com possibilidade de um *zoom* nas grandes cidades, como por exemplo Nova Iorque, Paris e São Paulo. A função em Perl que foi utilizada para cálculo de distância entre localidades é apresentada na Figura 5.15.

```
sub calculaDistancia{
       my @argumento = @_;
       my $distancia = -7;
       $distancia++; $distancia--;

       my $origem  = $argumento[0];
       my $destino = $argumento[1];
       my $url =
"http://maps.google.com.br/maps?saddr=$origem&daddr=$destino&hl=en&z=1";
       my $page = get($url);
       my @dados = split(/,distance:"/, $page);
       my @dist  = split(/ km",/, $dados[1]);
       $dist[0] =~ s/,//g;
       $distancia = $dist[0];
       return $distancia;
}
```

Figura 5.15: Código fonte em Perl para cálculo de distância entre terminais

Os experimentos iniciais realizados ao se tentar resolver quaisquer dos 12 problemas com o CPLEX puderam confirmar a dificuldade em se obter a solução ótima para esses problemas. Por exemplo, ao se tentar resolver a instância correspondente ao mês de janeiro (a menor dentre as 12 instâncias), na sua versão simplificada, considerando apenas um único tipo de veículo, não se conseguiu atingir a solução ótima após 18 horas de processamento, e o *gap* no momento da interrupção do processamento ainda era de 17,61%. Já para o mesmo mês (janeiro), porém considerando os dois tipos de veículos (truck e carreta), não foi possível ao CPLEX encontrar nenhuma solução viável após 18 horas de processamento,

Assim, dada a dificuldade para a obtenção da solução ótima para cada uma das 12 instâncias, e também devido à restrição imposta do tempo aceitável para se obter uma solução para o problema prático, optou-se por realizar os experimentos considerando-se um tempo limite máximo para processamento, tanto para o CPLEX como também para a heurística baseada em busca tabu, de 30 minutos.

Esse tempo limite de 30 minutos condiz com a expectativa da Patrus Transportes, que pretende não só utilizar o modelo como uma ferramenta de planejamento tático, mas também diariamente avaliar possíveis modificações na sua estrutura de rotas, adicionando e/ou excluindo novas rotas, ou ainda alterando o tipo de veículo alocado em cada rota, de acordo com o volume de cargas no dia.

Adicionalmente, foram adotados os seguintes parâmetros para a heurística de melhoria baseada em busca tabu: o número de iterações que cada movimento permanece tabu (*tabu tenure*) foi fixado em 5 iterações; o mecanismo de diversificação é ativado após 100 iterações sem melhoria na solução incumbente. Já para a heurística de resolução do problema de *bin packing*, também baseada em busca tabu, e que permite determinar o número de veículos de cada tipo utilizados em cada rota, a duração tabu também foi fixada em 5 iterações, de um total de 10 iterações realizadas, sendo esse o critério de parada.

Na Tabela 5.7 são apresentados os resultados obtidos com os experimentos computacionais realizados. A coluna com o nome 'CPLEX¹' corresponde à melhor solução inteira obtida pelo software após 30 minutos de processamento, para o problema, considerando somente um tipo de veículo, o veículo maior (carreta). Na coluna 'CPLEX GAP (%)¹' é apresentado o gap do CPLEX no momento da interrupção do processamento. Os resultados da busca tabu são apresentados na coluna 'Tabu¹'. A coluna 'Desvio Tabu (%)¹', refere-se à diferença percentual obtida ao se comparar as soluções obtidas com o CPLEX e com a busca tabu, calculada como:

$$Desvio\ Tabu\,(\%) = \frac{(Tabu - CPLEX)}{CPLEX} \times 100.$$

Essa comparação só pôde ser realizada com os problemas resolvidos com somente um único tipo de veículo (carreta), dado que o CPLEX não conseguiu obter nenhuma solução inteira para os problemas com dois tipos de veículos (truck e carreta), mesmo após um período de tempo 36 vezes maior que o máximo definido, ou seja, 18 horas de processamento. As soluções obtidas com a busca tabu para o problema com dois tipos de veículo (truck e carreta) são apresentadas na última coluna da tabela, com o título 'Tabu²'.

Tabela 5.7: Resultados obtidos – Patrus Transportes

Mês	CPLEX[1]	CPLEX GAP (%)[1]	Tabu[1]	Desvio Tabu (%)[1]	Tabu[2]
JAN	51.241,4	22,2%	47.678,4	-7,0%	44.908,2
FEV	48.214,6	22,3%	46.999,4	-2,5%	42.790,0
MAR	50.591,8	17,8%	49.193,2	-2,8%	47.768,8
ABR	56.277,2	24,8%	49.648,2	-11,8%	49.329,0
MAI	54.451,6	18,5%	53.422,6	-1,9%	51.408,6
JUN	54.093,2	18,9%	52.348,8	-3,2%	50.483,2
JUL	54.385,8	22,0%	51.371,6	-5,5%	50.573,0
AGO	55.281,8	19,3%	54.733,0	-1,0%	53.030,2
SET	59.563,0	23,0%	57.120,4	-4,1%	54.262,2
OUT	61.901,0	19,5%	59.788,4	-3,4%	58.642,4
NOV	65.699,2	23,6%	61.663,0	-6,1%	63.092,4
DEZ	60.303,6	22,9%	56.998,2	-5,5%	56.020,6

Analisando os resultados obtidos apresentados na Tabela 5.7, é possível verificar que a busca tabu foi capaz de obter as melhores soluções em todos os problemas, obtendo um desvio percentual médio, em relação às melhores soluções obtidas através do CPLEX, de -4,6%, tendo como desvio mínimo de -1,0% com o problema do mês de agosto, e máximo de -11,8% no problema de mês de abril. A comparação de forma gráfica das soluções obtidas com o CPLEX e com a busca tabu, para um único tipo de veículo (carreta), é apresentada na Figura 5.16.

As soluções obtidas com o CPLEX, mesmo para somente um único tipo de veículo, apresentaram um *gap* médio de 21,2%, confirmando a dificuldade de se resolver esta classe de problemas. Verificou-se também que, mesmo após 18 horas de processamento, o *gap* estava ainda acima de 15%. Como relatado anteriormente, para a instância correspondente ao mês de janeiro, após 18 horas de processamento, o *gap* estava em 17,61%.

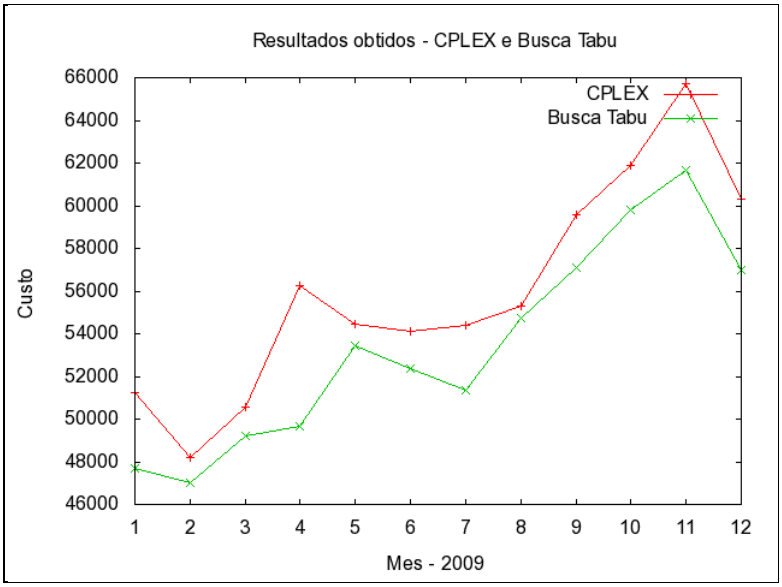

Figura 5.16: Resultados obtidos para um único tipo de veículo

5.5 CONCLUSÕES DO CAPÍTULO

Esse capítulo tratou do problema de carregamento de carga parcelada em uma rede de transporte do tipo *hub-and-spoke.* Foi proposto um modelo matemático para representar o problema, que foi inspirado em uma formulação apresentada para um problema similar conhecido na literatura como *network loading problem*, que trata especificamente da modelagem de problemas de configuração de redes de telecomunicações.

O modelo matemático proposto para representar o problema possui somente variáveis binárias e inteiras, dificultando a utilização de métodos exatos tradicionais, como pacotes comerciais de otimização, para sua solução. Com isso foi desenvolvida uma heurística baseada em busca tabu visando à obtenção de soluções de forma rápida e eficiente para o problema.

Foram realizados experimentos computacionais utilizando dados de um ano de operação de uma empresa transportadora de carga parcelada no Brasil chamada Patrus Transportes Urgentes Ltda. Os dados fornecidos resultaram em 12

problemas, que foram resolvidos tanto utilizando o pacote comercial CPLEX (versão 9.1), como também com a heurística busca tabu proposta.

Os resultados obtidos com os experimentos computacionais evidenciaram a dificuldade de se obter soluções de qualidade, em um tempo de processamento aceitável, para esta classe de problemas. Com isso, o tempo de processamento foi limitado em 30 minutos, tanto para o CPLEX como para a busca tabu, sendo condizente com a expectativa da Patrus Transportes, que pretende não só utilizar o modelo como uma ferramenta de planejamento tático, mas também diariamente avaliar possíveis modificações nas rotas a serem operadas em cada terminal, adicionando e/ou excluindo novas rotas, ou ajustando o tipo de veículo adequado para cada rota, de acordo com o volume de cargas no dia.

A busca tabu foi capaz de obter as melhores soluções em todos os problemas resolvidos, obtendo em média uma solução final 4,6% melhor ao se comparar com os resultados obtidos utilizando o CPLEX, mostrando ser uma alternativa interessante para solução desta classe de problemas, sendo simples de ser implementada computacionalmente, não necessitando de calibrações específicas de parâmetros para a obtenção de soluções de boa qualidade.

É importante destacar ainda que, para o caso especial em que são tratados mais de um tipo de veículo para a operação, no estudo de caso veículos do tipo truck e carreta, o CPLEX, mesmo após 18 horas de processamento, não pôde obter nenhuma solução inteira para nenhuma das 12 instâncias aqui estudadas.

Um ponto importante a ressaltar é que o problema do dimensionamento da quantidade e tipo de veículos utilizados em cada rota, modelados como o problema conhecido na literatura como BPP, por si só já é um problema complexo e o esforço para sua solução poderia ser um objeto de pesquisa.

6 CONCLUSÕES E RECOMENDAÇÕES

Esta pesquisa foi centrada no projeto de redes de distribuição de carga parcelada. Mais especificamente foram tratados dois tipos de problemas que são comuns no planejamento desse tipo de sistema. O primeiro deles corresponde ao problema estratégico de configuração de redes do tipo *hub-and-spoke*, consistindo na definição simultânea da quantidade e localização de terminais para consolidação de carga (ou *hubs*), e na definição da alocação dos terminais aos *hubs* localizados. Uma vez determinada a configuração da rede, o segundo problema, no nível de decisão tático, corresponde na definição do caminho que cada carga parcelada deve percorrer desde sua origem até alcançar seu terminal de destino, a um mínimo custo, tendo a rede *hub-and-spoke* como um dado de entrada do problema.

A configuração da rede tratada na presente pesquisa considerou *hubs* não capacitados com alocação única, tendo sido propostas quatro heurísticas para sua solução, cujos experimentos computacionais proporcionaram resultados que superaram todos os métodos já desenvolvidos e apresentados na literatura. Foi ainda proposto um novo modelo matemático para representar o problema, permitindo a obtenção da solução ótima para instâncias com redes de até 100 terminais.

Já em nível de decisão tático, o estudo foi centrado na definição do fluxo (percurso ou caminho) que cada carga deve percorrer até chegar ao seu destino, tendo como dado de entrada uma rede de transporte do tipo *hub-and-spoke* com as possíveis ligações entre terminais já pré-definidas. Dado um conjunto de cargas a serem transportadas, deseja-se, portanto decidir como estas cargas serão movimentadas nesta rede, definindo as possíveis paradas intermediárias, e também o dimensionamento da frota de veículos necessários para cada rota a ser operada.

Esse problema é de grande importância prática, pois o objetivo consiste em definir diferentes formas de consolidação e as rotas a serem operadas, tentando acompanhar a variação na demanda por transporte de carga parcelada, sendo comum no Brasil, em certas semanas do mês, e em certos meses do ano.

As contribuições desse trabalho podem ser sumarizadas no que diz respeito à (i) modelagem matemática proposta para os problemas tratados, (ii) no desenvolvimento de métodos de solução eficientes, e (iii) aplicação prática dos métodos de solução propostos para tentar verificar a aderência dos modelos matemáticos e métodos de solução desenvolvidos com problemas reais de empresas transportadoras de carga parcelada no Brasil.

6.1 CONCLUSÕES GERAIS

Para o problema estratégico de configuração de redes do tipo *hub-and-spoke*, foi proposto um novo modelo matemático para representar o problema reduzindo tanto a quantidade de variáveis como de restrições, permitindo reportar pela primeira vez a solução ótima para problemas gerados utilizando conjuntos de dados comumente utilizados na literatura como *benchmark*, conhecidos como CAB (*Civil Aeronautics Board)* e AP (*Australian Post).*

Todos os problemas do CAB (80 problemas), e do AP (12 problemas) com até 100 nós, puderam ser resolvidos até a otimalidade, em um razoável tempo de processamento, sendo esta pesquisa a primeira abordagem de sucesso na obtenção de soluções ótimas para o USAHLP utilizando métodos exatos. Foram também geradas instâncias de grande porte e modificadas partindo-se do conjunto de dados AP, com custos fixos menores, sendo intrinsecamente mais difíceis de serem resolvidos, dada a maior quantidade de *hubs* presente na solução final.

Quatro heurísticas foram propostas para solução do problema estratégico de configuração de uma rede do tipo *hub-and-spoke*, Três delas baseadas em uma estratégia multi-início para a definição da quantidade e localização de *hubs*, nomeadas MSTS, e a busca tabu para a definição da alocação de cada terminal da rede aos *hubs* localizados. A última heurística desenvolvida foi baseada exclusivamente na utilização da busca tabu, denominada neste trabalho de HubTS, que foi utilizada tanto para a definição da quantidade e localização de *hubs*, como também na solução do problema de alocação dos terminais aos *hubs.*

Tanto a heurística MSTS (nas suas três variantes), como também a heurística HubTS, se mostraram eficientes para solução do problema, obtendo a solução ótima em todos os problemas testados, possuindo mecanismos simples e fáceis de serem implementadas, e ainda robustas no que diz respeito à qualidade das soluções obtidas, evidenciado pelos experimentos computacionais realizados, não necessitando de calibrações específicas de parâmetros para obtenção de soluções de qualidade.

Em nível de decisão tático, a pesquisa foi centrada no problema de carregamento de cargas parceladas em uma rede de transporte do tipo *hub-and-spoke*. O modelo matemático proposto foi derivado da representação matemática de um problema conhecido na literatura como *network loading problem* (NLP), que trata especificamente do problema de configuração de redes de telecomunicações.

A maior contribuição relativa ao modelo matemático foi justamente a abordagem inédita utilizada para representar o problema. Contudo, esse modelo possui somente variáveis binárias e inteiras, dificultando a utilização de métodos exatos tradicionais, como pacotes comerciais de otimização, para a solução de problemas de tamanho comparável aos encontrados na prática. Com isso, foi desenvolvido um método de solução utilizando a metaheurística busca tabu.

A busca tabu foi utilizada tanto na definição das rotas a serem operadas, como também na quantidade e tipo de veículos utilizados em cada percurso. O problema do dimensionamento da quantidade de veículos utilizados em cada rota na heurística proposta para solução foi modelado como um problema conhecido na literatura com o nome de *bin packing problem* (BPP), sendo por si só muito difícil de ser resolvido.

Os experimentos computacionais realizados consistiram em se resolver o problema tático de carregamento de cargas parceladas tanto utilizando a busca tabu, como também o pacote de otimização CPLEX. Os problemas estudados foram gerados utilizando dados reais fornecidos por uma empresa transportadora de carga parcelada no Brasil pelo modal rodoviário. Dada a dificuldade para a obtenção de

soluções ótimas para o problema de tamanho igual ao encontrado na prática, o tempo de processamento tanto para a busca tabu, como também para o CPLEX, foi limitado em 30 minutos. Assim, a busca tabu foi capaz de obter as melhores soluções em todos os problemas resolvidos, mostrando ser uma alternativa interessante para solução desta classe de problemas, sendo também simples de ser implementada computacionalmente, não necessitando de calibrações específicas de parâmetros para a obtenção de soluções.

6.2 SUGESTÕES PARA ESTUDOS FUTUROS

Dada a natureza das heurísticas multi-início propostas para solução do problema estratégico de configuração de redes do tipo *hub-and-spoke*, uma implementação paralela, seja utilizando arquiteturas de memória distribuída ou compartilhada, seria muito interessante para reduzir o tempo de processamento necessário para solução de problemas de grande porte.

Uma possibilidade também promissora seria definir um limitante inferior para determinar a quantidade de *hubs* a serem localizados, sendo esta alternativa especialmente útil para a heurística HubTS, evitando iniciar a busca com somente um único *hub*.

Finalmente, seria interessante alternativamente investigar também a aplicação das heurísticas desenvolvidas para outras variantes do problema de configuração de redes *hub-and-spoke*, como por exemplo, o problema que possui restrições de capacidade no que diz respeito à quantidade de carga classificada nos *hubs*, e ainda o problema que permite alocação múltipla, ou seja, um terminal não-*hub* alocado a mais de um *hub*.

Já para o problema tático de carregamento de cargas parceladas em uma rede de transporte, uma tarefa imediata seria definir uma forma mais precisa de se avaliar e validar a heurística proposta para solução do problema, talvez utilizando diferentes conjuntos de dados para testes, incluindo talvez instâncias de menor porte.

Outra possibilidade seria também o desenvolvimento de métodos alternativos para solução do problema, como por exemplo, heurísticas baseadas em algoritmos evolucionários, busca em vizinhança variável e de grande porte, heurísticas híbridas, dentre outras.

REFERÊNCIAS

ABDINNOUR-HELM, S. A hybrid heuristic for the uncapacitated hub location problem. **European Journal of Operations Research,** v.106, p.489-499, 1998.

_____. Using simulated annealing to solve the p-hub median problem. **International Journal of Physical Distribution & Logistics Management**, v. 31 (3), p. 203–220, 2001.

ABDINNOUR-HELM, S., VENKATARAMANAN, M.A., Solution approaches to hub location problems. **Annals of Operations Research**, v. 78, p. 31–50, 1998.

ALAMO, J.A.T. **Modelagem para a localização de hubs no transporte de encomendas expressas.** 2005. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2005.

ALTIN, A.; YAMAN, H.; PINAR, M.C. **A Hybrid Polyhedral Uncertainty Model for The Robust Network Loading Problem,** April 2009, forthcoming in *Performance Models and Risk Management in Communication Systems,* Gulpinar, Harrison and Rustem (Eds.), Springer-Verlag, 2010.

ALUMUR, S.; KARA, B.Y. Network hub location problems: The state of the art. **European Journal of Operational Research,** v. 190, p. 1-21, 2008.

ARMACOST, A.P.; BARNHART, C.; WARE, K.A. Composite variable formulations for express shipment service network design. **Transportation Science**, v. 36(1), p. 1-20, 2002.

AVELLA, P.; MATTIA, S.; SASSANO, A. Metric inequalities and the network loading problem. **Discrete Optimization**, v. 4, p. 103-114, 2007.

AYKIN, T., Lagrangean relaxation based approaches to capacitated hub-and-spoke network design problem. **European Journal of Operational Research**, v. 79 (3), 501–523, 1994.

_____. Network policies for hub-and-spoke systems with application to the air transportation system. **Transportation Science**, v. 29(3), 1995.

_____. On the location of hub facilities. **Transportation Science**, v. 22 (2), p. 155–157, 1988.

AYKIN, T., BROWN, G.F., Interacting new facilities and location-allocation problem. **Transportation Science**, v. 26 (3), 212–222, 1992.

BABONNEAU, F.; VIAL, J.-P. A partitioning algorithm for the network loading problem. **European Journal of Operational Research**, v. 204, p. 173-179, 2010.

BALLOU, R.H. **Business logistics management: planning, organizing and controlling the supply chain.** N.J.: Prentice-Hall, 1999, 681p.

BARAHONA, F. Network design using cut inequalities, **SIAM Journal of Optimization**, v. 6(3), p. 823-837, 1996.

BARNHART, C.; JIN, H.; VANCE, P.H. Railroad blocking: a network design application. **Operations Research**, v. 48(4), p. 603-614, 2000.

BARNHART, C.; SCHNEUR, R.R. Air network design for express shipment service. **Operations Research**, v. 44, p. 852-863, 1996.

BARTOLINI, E.; MINGOZZI, A. Algorithms for the non-bifurcated network design problem. **Journal of Heuristics**, v. 15, p. 259-281, 2009.

BATTITI, R.; TECCHIOLLI, G. The reactive tabu search. **ORSA Journal on Computing**, v. 6(2), p. 126-140, 1994.

BAUMGARTNER, S., **Polyhedral analysis of hub center problems**. Diploma thesis, Universita¨ t Kaiserslautern, 2003.

BEASLEY, J. E. OR-library: distributing test problems by electronic mail. **Journal of the Operational Research**, v. 41, p. 1069-1072, 1990.

BENATTI, F. Dificuldades operacionais e necessidades de investimentos exigem reajuste imediato de 18% no frete, **NTC&Logística**, 5 fev. 2010. Disponível em: < http://www.ntcelogistica.org.br/noticias/materia_completa.asp?CodNoti=38847>, Acesso em 10 fev. 2010.

BERGER, D.; GENDRON, B.; POTVIN, J-Y, RAGHAVAN, S.; SORIANO, P. Tabu search for a network loading problem with multiple facilities. **Journal of Heuristics**, v. 6, p. 253-267, 2000.

BIENSTOCK, D.; GÜNLÜK, O. Computational experience with a difficult mixed-integer multicommodity flow problem. Mathematical Programming, v. 68(2), p. 213-237, 1995.

_____. Capacitated Network Design – polyhedral structure and computation. **INFORMS Journal on Computing**, v.8(3), p. 243-259, 1996.

BOLAND, N.; KRISHNAMOORTHY, M., ERNST, A.T.; EBERY, J. Preprocessing and cutting for multiple allocation hub location problems. **European Journal of Operational Research**, v. 155 (3), p. 638–653, 2004.

BRAKLOW, J.W.; GRAHAM, W. W.; HASSLER, S. M.; PECK, K. E.; POWELL, W. B. Interactive optimization improves service and performance for Yellow Freight Systems. **Interfaces**, v. 22(1), p. 147-172, 1992.

BRANDEAU, M.L.; CHIU, S.S. An overview of representative problems in location research. **Management Science**, 1989, p.645-667.

CALIK, H.; ALUMUR, S.A.; KARA, B.Y.; KARASAN, O.E. A tabu search based heuristic for the hub covering problem over incomplete hub networks. **Computers & Operations Research**, v. 36, p. 3088-3096, 2009.

CAMPBELL, J. F., Locating transportation terminals to serve an expanding demand. **Transportation Research B**, v. 24 (3), p. 173–192, 1990.

_____. Location and allocation for distribution systems with transshipments and transportation economies of scale. **Annals of Operations Research**, v. 40, p. 77–99, 1992.

_____. A survey of network hub location. **Studies in Locational Analysis**, v. 6, p. 31–49, 1994.

_____. Hub location problem and the p-hub median problem. **Operations Research**, v.44, p.923-936, 1996.

CAMPBELL, A.M., LOWE, T.J., ZHANG, L., 2007. The p-hub center allocation problem. **European Journal of Operational Research**, v. 176 (2), p. 819–835, 2007.

CÁNOVAS, L.; GARCÍA, S.; MARÍN, A. Solving the uncapacitated multiple allocation hub location problem by means of a dual-ascent technique. **European Journal of Operational Research**, v. 179, p. 990–1007, 2007.

CHAN, Y.; PONDER, R.J. The small package air freight industry in the United States: A review of the federal express experience, **Transportation Research Part A: General**, v. 13, n. 4, p. 221-229, Agosto, 1979.

CHEN, J.F., A hybrid heuristic for the uncapacitated single allocation hub location problem. **Omega**, 35, p. 211–220, 2007.

CORDEAU, J., TOTH, P., and VIGO, D. A Survey of Optimization Models for Train Routing and Scheduling. **Transportation Science, V.** 32(4), p. 380-404, 1998.

COSTA, M.G.; CAPTIVO, M.E.; CLÍMACO, J. Capacitated single allocation hub location problem – A bi-criteria approach. **Computers & Operations Research**, v. 35, p. 3671-3695, 2008.

CRAINIC, T.G. **A survey of optimization models for long-haul freight transportation.** Quebec: Départament des sciences administratives, Université du Québec à Montreal, 1998.

_____. Service network design in freight transportation, **European Journal of Operational Research**, v. 122, p. 272-288, 2000.

CRAINIC, T.G.; FERLAND, J.A.; ROUSSEAU, J.M. A tactical planning model for rail freight transportation. **Transportation Science**, v. 18(2), p. 165-184, 1984.

CRAINIC, T.G.; ROUSSEAU, J.M. Multicommodity, multimode freight transportation: a general modeling and algorithmic framework. **Transportation Research Part B**: **Methodological**, v. 20, p. 225-242, 1986.

CRAINIC, T.G.; ROY, J. O.R. tools for tactical freight transportation planning. **European Journal of Operational Research**, v. 33(3), p. 209-297, 1988.

CUNHA, C.B. **Contribuição à modelagem de problemas de logística e transportes**. 2006. 315p. Tese (Livre Docência) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2006.

CUNHA, C.B., SILVA, M.R., A genetic algorithm for the problem of configuring a hub-and-spoke network for a LTL trucking company in Brazil. **European Journal of Operational Research**, v. 179, p. 747–758, 2007.

DEJAX, P.J.; CRAINIC, T.G. A review of empty flows and fleet management models in freight transportation, **Transportation Science**, v. 21(4), p. 227-247, 1987.

DELL'AMICO, M.; LODI, A; MAFFIOLI, F. Solution of the cumulative assignment problem with a well-structured tabu search method. **Journal of Heuristics**, v. 5, p. 123-143, 1999.

EBERY, J. et al. The capacitated multiple allocation hub location problem: formulations and algorithms. **European Journal of Operations Research**, n. 120, 2000, p.614-631.

EBERY, J., KRISHNAMOORTHY, M., ERNST, A., BOLAND, N., The capacitated multiple allocation hub location problem: Formulations and algorithms. **European Journal of Operational Research**, v. 120, p. 614–631, 2000.

ERNST, A., HAMACHER, H., JIANG, H., KRISHNAMOORTHY, M., WOEGINGER, G., Heuristic algorithms for the uncapacitated hub Center single allocation problem. **Unpublished Report, CSIRO Mathematical and Information Sciences**, Australia, 2002.

ERNST, A.T., JIANG, H., KRISHNAMOORTHY, M., Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems. **Unpublished Report, CSIRO Mathematical and Information Sciences**, Australia, 2005.

ERNST, A. T.; KRISHNAMOORTHY, M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. **Location Science,** v.24, n. 104, p.139-154, 1996.

_____. Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. **European Journal of Operational Research**, v. 104, p. 100–112, 1998a.

_____. Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. **European Journal of Operations Research,** n.104, p.100-112, 1998b.

_____. Solution algorithms for the capacitated single allocation hub location problem. **Annals of Operations Research**, v. 86, p. 141–159, 1999.

FOURER, R.; GAY, D.; KERNIGHAN, B. W. A Modeling Language for Mathematical Programming. **Management Science,** v. 36, p. 519-554, 1990.

GALLO, G.; PALLOTTINO, S. Shortest path algorithms **Annals of Operations Research 13**, p.3-79,1988.

GAREY, M.R. e JOHNSON, D.S. **Computers and Intractability: A Guide to the Theory of NP-Completeness.** New York, NY: W. H. Freeman e Co, 1979.

GAVRILIOUK, E.O., HAMACHER, H.W., **Aggregation for hub location problems**. Working paper. Department of Mathematics, University of Kaiserslautern, Gottlieb-Daimler-Strasse, 67663 Kaiserslautern, Germany, 2006.

GENDRON, B.; POTVIN, J-Y; SORIANO, P. Diversification strategies in local search for a nonbifurcated network loading problem. **European Journal of Operational Research**, v. 142, p. 231-241, 2002.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers & Operations Research**, v. 13, p. 533–49, 1986.

_____. Tabu search. Part i. **ORSA Journal on Computing**, v. 1, p. 1-190, 1989.

GOLDMAN, A.J., Optimal location for centers in a network. **Transportation Science**, v. 3, 352–360, 1969.

GRÜNERT, T.; SEBASTIAN, H.J. Planning models for long-haul operations of postal and express shipment companies, **European Journal of Operational Research**, v. 122, p. 289-309, 2000.

GUALDA, N.D.F **Terminais de transportes: contribuição ao planejamento e ao dimensionamento operacional**. 1995. 277p. Tese (Livre Docência) – Escola Politécnica, Universidade de São Paulo. São Paulo.

HAKIMI, S. Optimal location of switching centers and the absolute centers and medians of a graph. **Operations Research,** v. 12, p.450-559, 1964.

HAMACHER, H.W., LABBÉ, M., NICKEL, S., SONNEBORN, T. Adapting polyhedral properties from facility to hub location problems. **Discrete Applied Mathematics**, v. 145 (1), p. 104–116, 2004.

HAMACHER, H.W., MEYER, T., **Hub cover and hub center problems**. Working paper. Department of Mathematics, University of Kaiserslautern, Gottlieb-Daimler-Strasse, 67663 Kaiserslautern, Germany, 2006.

HAMAD, R. **Modelo para localização de instalações em escala global envolvendo vários elos da cadeia logística.** 2006. 62p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

HELLMUTH, B.R. **Uma contribuição ao estudo do planejamento de transporte rodoviário de cargas fracionadas.** 2004. 199p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

KARA, B.Y., TANSEL, B.C., On the single-assignment p-hub center problem. **European Journal of Operational Research**, v. 125, p. 648–655, 2000.

_____. The single-assignment hub covering problem: Models and linearizations. **Journal of the Operational Research Society**, v. 54, p. 59–64, 2003.

KAWAMURA, K. Commercial vehicle value of time and perceived benefit of congestion pricing. 1999. 107p. Tese (Doutorado) – University of California – Berkley.

KIM, D.; BARNHART, C.; WARE, K.; REINHARDT, G. Multimodal express package delivery: a service network design application. **Transportation Science**, v. 33(4), p. 391-407, 1999.

KLINCEWICZ, J.G. Heuristics for the p-hub location problem. **European Journal of Operations Research**, n.53, p.25-37, 1991.

_____. Avoiding local optima in the p-hub location problem using tabu search and GRASP. **Annals of Operations Research,** n.40, p.283-302, 1992.

_____. A dual algorithm for the uncapacitated hub location problem. **Location Science**, v.4, p.173-184, 1996.

KUBY, M.J.;, GRAY, R.G., The hub network design problem with stopovers and feeders: The case of Federal Express. **Transportation Research A,** v. 27 (1), p. 1–12, 1993.

LABBÉ, M., YAMAN, H., Projecting flow variables for hub location problems. **Networks**, v. 44 (2), p. 84–93, 2004.

LABBÉ, M., YAMAN, H., GOURDIN, E. A branch-and-cut algorithm for the hub location problems with single assignment. **Mathematical Programming,** v. 102, p. 371–405, 2005.

LARSON, R.; ODONI, A. **Urban operations research.** Englewood Cliffs, N.J.: Prentice-Hall, 1981.

MACHADO, S. B. **Estratégias de atendimento em redes logísticas e dimensionamento de terminais.** 1989. 139p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Transportes, São Paulo, 1989.

MAGNANTI, T.L.; MIRCHANDANI, P.; VACHANI, R. The convex hull of two core capacitated network design problems. **Mathematical Programming**, v. 26(2), p. 233-250, 1993.

_____. Modeling and solving the two-facility capacitated network loading problem. **Operations Research**, v. 43(1), p. 142-157, 1995.

MARÍN, A. Formulating and solving splittable capacitated multiple allocation hub location problems. **Computers & Operations Research**, v. 32 (12), p. 3093–3109, 2005.

MARÍN, A; CÁNOVAS, L; LANDETE, M. New formulations for the uncapacitated multiple allocation hub location problem. **European Journal of Operational Research**, v. 172 (1), p. 274–292, 2006.

MARTÍ, R. Multi-start methods. In: Glover F, Kochenberger G, editors. **Handbook of metaheuristics. International series in operations research and management science**, vol. 57. Dordrecht: Kluwer Academic Publishers, p. 321–53, 2002.

MARTÍ, R; VEJA, M. M. Multistart methods. **Inteligencia Artificial**, v. 19, p. 49–60, 2003.

MATSUMOTO, M; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. **ACM Transactions on Modeling and Computer Simulation**, v. 8(1), p. 3–30, 1998.

MAYER, G.; WAGNER, B. Hublocator: an exact solution method for the multiple allocation hub location problem. **Computers & Operations Research,** n.29, p.715-739, 2002.

METROPOLIS, N.; ROSENBLUTH, A.W.; ROSENBLUTH, M.N.; TELLER, A.H.; TELLER, E. Equation of state calculation by fast computer machines. **Journal of Chemical Physics,** n.20, 1953.

MEYER, T.; ERNST, A.T.; KRISHNAMOORTHY, M. A 2-phase algorithm for solving the single allocation p-hub center problem. **Computers & Operations Research**, v. 36, p. 3143-3151, 2009.

MIRCHANDANI, P. Projections of the capacitated network loading problem. **European Journal of Operational Research**, v. 122(3), p. 534-560, 2000.

MUTARELLI, F. **Modelagem de redes de distribuição aplicada ao caso de uma editora de revistas.** 2004. 108p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Transportes, São Paulo, 2004.

NOVAES, A.G. **Sistemas logísticos: transporte, armazenagem, e distribuição física de produtos.** São Paulo: Edgard Blücher Ltda., 1989, 372p.

O'KELLY, M.E., The location of interacting hub facilities. **Transportation Science**, 20 (2), 92–105, 1986.

_____. A quadratic integer program for the location of interacting hub facilities. **European Journal of Operations Research**, n. 32, p.393-404, 1987.

_____. Hub facility location with fixed costs. **The Journal of the Regional Science Association International**, n.71, v.3, p.293-306, 1992.

O'KELLY, M. E.; MILLER, H. J. Solution strategies for the single facility minimax hub location problem. **Papers in Regional Science**, v. 70 (4), p. 367–380, 1991.

PAMUK, F.S., SEPIL, C., A solution to the hub center problem via a single-relocation algorithm with tabu search. **IIE Transactions**, v. 33 (5), p. 399–411, 2001.

PIRES, T. **Configuração de uma rede de distribuição capacitada com restrição de cobertura.** 2006. 111p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

PIRKUL, H.; SCHILLING, D. A. An efficient procedure for designing single allocation hub and spoke systems. **Management Science***,* v. 44 (12), p. 235-242, 1998.

POWELL, W. B. A local improvement heuristic for the design of less-than-truckload motor carrier networks, **Transportation Science**, v. 20 (4), 246-357, 1986.

POWELL, W.B.; JAILLET, P.; ODONI, A. **Stochastic and dynamic networks and routing**, Handbook in Operations Research and Management Science, Networks, v. 4, (M.O. Ball, T.L. Magnanti, C.L. Monma e G.L. Nemhauser, eds.), p. 141-295, 1995.

POWELL, W.B.; SHEFFI, Y. The load-planning problem of motor carriers: problem description and a proposed solution approach, **Transportation Research A: Policy and Practice**, v. 17(6), p. 471-480, 1983.

_____. Design and implementation of an interactive optimization system for the network design in the motor carrier industry, **Operations Research**, v. 37(1), p. 12-29, 1989.

RARDIN, R.L.; WOLSEY, L.A. Valid Inequalities and Projecting the Multicommodity Extended Formulation for Uncapacitated Fixed Charge Network Flow Problems, **European Journal of Operational Research**, v. 71, p. 95-109, 1993.

ROMERO, B.C. **Análise da localização de plataformas logísticas: aplicação ao caso do ETSP – Entreposto do Terminal São Paulo – da CEAGESP.** 2006. 144p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

SASAKI, M., FUKUSHIMA, M., On the hub-and-spoke model with arc capacity constraints. **Journal of the Operations Research Society of Japan**, v. 46 (4), p. 409–428, 2003.

SASAKI, M.; SUZUKI, A.; DREZNER, Z. On the selection of hub airports for an airline hub-and-spoke system. **Computers & Operations Research**, n.26, p.1411-1422, 1999.

SILVA, M.R. **Uma contribuição ao problema de localização de terminais de consolidação no transporte de carga parcelada.** 2004. 85p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Transportes, São Paulo, 2004.

SINGLETON, R. C. (1969) An efficient algorithm for sorting with minimal storage: Algorithm 347. **Communications of the ACM 12**, p. 185–187, 1969.

SKORIN-KAPOV, D.; SKORIN-KAPOV, J. On tabu search for the location of interacting hub facilities. **European Journal of Operations Research,** n.73, p.502-509, 1994.

SKORIN-KAPOV, D.; SKORIN-KAPOV, J.; O'KELLY, M. Tight linear programming relaxations of uncapacitated p-hub median problem. **European Journal of Operations Research**, n.94, p.582-593, 1996.

SOHN, J.; PARK, S., A linear program for the two-hub location problem. **European Journal of Operational Research**, v. 100 (3), p. 617–622, 1997.

_____. Efficient solution procedure and reduced size formulations for p-hub location problems. **European Journal of Operations Research**, n.108, p.118-126, 1998.

_____. The single allocation problem in the interacting three-hub network. **Networks**, v. 35 (1), p. 17–25, 2000.

STARR, R. M.; STINCHCOMBE, M. B. Efficient transportation planning and natural monopoly in the airline industry: An economic analysis of hub-spoke and related systems. Discussion Paper 92-25, University of California, San Diego, Department of Economics, 1992.

STECCO, G.; CORDEAU, J-F.; MORETTI, E. A tabu search heuristic for a sequence-dependent and time-dependent scheduling problem on a single machine. **Journal of Scheduling**, v. 12 (1), p. 3-16, 2009.

TEYPAZ, N.; SCHRENK, S.; CUNG, V.-D. A decomposition scheme for large-scale service network design with asset management, **Transportation Research Part E**, v. 46, p. 156-170, 2010.

TONDO, C.M. **Um modelo matemático para a localização estratégica de terminais de contêineres no interior: aplicação ao estado de São Paulo.** 1992. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo. São Paulo, 1992.

TOPCUOGLU, H., CORUT, F., ERMIS, M., YILMAZ, G., Solving the uncapacitated hub location problem using genetic algorithms. **Computers & Operations Research**, v. 32 (4), p. 967–984, 2005.

VALLIN FILHO, A.R.A. **ULocalização de centros de distribuição de carga: contribuições a modelagem matemática.** 2004. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2004.

WAGNER, B. **Model formulations for hub covering problems**. Working paper, Institute of Operations Research, Darmstadt University of Technology, Hochschulstrasse 1, 64289 Darmstadt, Germany, 2004.

WIEBERNEIT, N. Service network design for freight transportation: a review, **OR Spectrum**, v. 30, p. 77-112, 2008.

APÊNDICE A – DADOS UTILIZADOS PARA O ESTUDO DE CASO PATRUS TRANSPORTES

Neste apêndice são apresentados os dados fornecidos pela empresa Patrus Transportes Urgentes Ltda., para os experimentos computacionais relativos ao problema tático de carregamento de cargas parceladas em uma rede *hub-and-spoke*.

Os dados fornecidos foram manipulados apropriadamente e resultou em 12 instâncias, uma para cada mês do ano, contendo o valor médio diário do peso das cargas transportadas.

Por solicitação da Patrus Transportes, a localidade de cada terminal foi preservada, sendo substituída por um número seqüencial de 1 até a quantidade de terminais de cada problema.

As 12 instâncias aqui apresentadas estão representadas na linguagem algébrica AMPL. A notação utilizada para representar os conjuntos, parâmetros e variáveis de decisão do modelo matemático, bem como os resultados obtidos com os experimentos computacionais realizados, estão detalhados no Capítulo 5.

```
janeiro.dat
data;
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 ;
set A := 43 24, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 15 37,
16 43, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23
20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6,
23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16,
15 29, 45 6, 16 9, 24 31, 5 16, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49
27, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 25, 19 43, 21 20, 24
23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 7 8, 24 21, 15 14, 15 7,
6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15
22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33 29, 19
32, 49 24, 29 1, 6 50, 43 46, 7 2, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18
23, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 43 23, 27 37, 27 33, 23
49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 15 20, 19 18, 21 1, 24 20, 7 29, 15 9,
47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46,
33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13,
23 45, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47,
19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 5 21, 15 11, 15 50, 21 27,
45 15, 29 28, 16 38, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19,
5 23, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27
50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 29 17, 33 27, 16 28, 15 27, 23
43, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 5 19, 35 19, 48 19, 16 46, 30 27, 16
7, 24 1, 19 51, 6 32, 21 50, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 7 9, 16 1,
6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2
29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40,
7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5,
29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31
16, 29 16, 24 47, 15 31, 19 20, 6 7, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17
19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17 21, 15 1, 2 24,
12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29,
23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 19 9, 43
19, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30,
27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42,
15 34, 16 20, 15 21, 29 3, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24
33, 23 12, 21 14, 16 26, 29 8, 45 24, 6 21, 5 7, 33 24, 49 6, 19 39, 46 7, 24 22, 5
6, 7 44, 23 37, 21 39, 25 27, 23 46, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30,
23 35, 30 7, 21 36, 29 43, 27 44, 7 28, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3,
19 25, 16 32, 48 27, 23 17, 30 36, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 21
43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36,
23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 31 29, 47 19, 23 29, 20 6, 6
```

```
37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 7 33, 27 20, 16 39,
23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16
41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24,
19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15
16, 23 1, 18 16, 29 14, 15 18, 35 2, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1,
23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24
15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 19 37, 15 28, 43 27, 20 21, 5
29, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8,
36 29, 24 51, 25 19, 24 46, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 24 38, 47
23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 24 35, 6 17, 16 21, 20 19, 19
3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30,
27 42, 24 40, 16 14, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29
12, 18 19, 7 14, 46 43, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21
45, 23 23, 24 27, 5 24, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45,
27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 27 41, 6 46, 19 31, 16 47, 6 22, 27 32, 16
3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 5 27, 20 29, 21 41, 35 15, 16 23,
29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7
31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 21 21, 45 23, 6 38, 5 15, 27 13, 6
34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24
43, 15 17, 31 6, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39,
35 23, 23 6, 21 22;
param: K:    R      O      D:= 1  3456   43     29 2   12002  23     27 3   664    21
       49 4   2407   23     34 5   1076   7      46 6   4123   21     33 7   1192   37
       7 8    7026   21     30 9   15648  27     7 10   2396   27     26 11  2020   31
       21 12  3931   23     24 13  13991  36     7 14   492    7      43 15  650    45
       27 16  406    6      36 17  4457   43     27 18  3232   7      15 19  13042  15
       21 20  2765   17     27 21  16624  48     21 22  725    24     8 23   2659   21
       8 24   1719   29     26 25  16151  21     2 26   891    21     29 27  3018   32
       27 28  2829   29     39 29  18313  29     23 30  318    30     19 31  2185   25
       7 32   1522   27     49 33  231    35     27 34  942    2      21 35  5808   6
       21 36  10814  6      2 37   915    49     6 38   10063  27     40 39  3131   27
       3 40   4237   49     27 41  1182   30     21 42  9784   7      6 43   379    37
       44 44  3440   24     29 45  5053   24     14 46  2245   16     21 47  3468   16
       27 48  3654   47     21 49  1814   25     21 50  6335   27     30 51  177    44
       7 52   625    36     3 53   873    19     6 54   5217   33     27 55  514    23
       37 56  17477  6      24 57  1968   21     20 58  17542  15     27 59  5465   23
       43 60  1784   21     37 61  12777  24     23 62  5890   25     27 63  8431   27
       25 64  2492   24     37 65  302    12     21 66  4147   27     15 67  3158   2
       27 68  123    15     30 69  438    36     30 70  684    21     36 71  10291  29
       43 72  6885   30     27 73  599    23     21 74  6180   27     44 75  1756   21
       50 76  261    19     49 77  1611   6      19 78  6617   27     6 79   329    49
       2 80   1797   6      44 81  3573   7      8 82   20043  24     21 83  3914   46
       27 84  3729   27     19 85  5203   15     7 86   5010   29     45 87  2460   21
       19 88  589    30     6 89   4808   48     27 90  242    19     17 91  586    46
       43 92  545    19     11 93  5670   27     18 94  193    30     36 95  170    37
       27 96  8463   27     2 97   10133  21     25 98  4183   46     24 99  2187   21
```

```
      47 100 8378   7      24 101 403    43     45 102 1985   48     7 103  650    19
      30 104 314    21     43 105 23538  24     27 106 980    25     6 107  2018   29
      9 108  3438   44     37 109 7169   7      23 110 4388   44     24 111 1146   20
      24 112 6116   23     7 113  6508   6      25 114 500    15     36 115 3960   27
      51 116 2522   29     42 117 2148   24     44 118 1751   6      33 119 7456   43
      46 120 6777   21     6 121  506    7      2 122  2360   27     36 123 13477  23
      29 124 506    27     46 125 1012   27     32 126 2927   6      37 127 8675   27
      5 128  783    30     12 129 2332   6      8 130  760    47     27 131 128    5
      27 132 1809   6      1 133  473    36     13 134 3719   23     14 135 3128   27
      11 136 22408  27     21 137 13935  27     43 138 1695   6      3 139  1866   43
      23 140 16430  27     22 141 3214   27     33 142 10497  27     20 143 4988   6
      30 144 160    29     6 145  8109   19     27 146 655    24     6 147  152    17
      19 148 360    2      6 149  3332   7      37 150 8901   24     20 151 4975   7
      29 152 22031  21     24 153 2330   21     28 154 4333   6      49 155 6895   27
      35 156 310    4      21 157 22518  7      27 158 1331   21     31 159 368    33
      15 160 675    46     23 161 6634   6      15 162 371    45     23 163 3283   36
      15 164 761    37     24 165 4786   24     34 166 3422   36     21 167 836    12
      27 168 5202   27     23 169 11954  36     27 170 432    47     7 171  2128   29
      46 172 2493   21     44 173 7051   7      21 174 7604   27     45 175 4068   15
      33 176 515    2      49 177 2928   29     38 178 5866   27     10 179 433    23
      45 180 10017  29     27 181 409    24     43 182 425    30     11 183 3455   44
      27 184 429    6      18 185 1889   21     16 186 1051   27     17 187 362    19
      2 188  5483   21     15 189 252    35     2 190  132    18     27 191 2349   45
      41 192 11889  24     7 193  319    46     29 194 321    15     6 195  4636   7
      25 ;
param distancia default 0 := [43, 24] 1092 [6, 27] 569 [43, 29] 118 [23, 27] 1061
[21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909 [37, 7] 274 [15,
37] 643 [16, 43] 2225 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24] 460 [36, 7]
762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267
[23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15, 40] 601 [17,
16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444
[23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29,
39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29]
1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [5, 16] 1305 [33, 19] 326 [29, 29] 0
[27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459 [49, 27] 468 [27,
12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46, 21]
2128 [27, 39] 1168 [23, 25] 1187 [19, 43] 1717 [21, 20] 994 [24, 23] 735 [23, 2]
1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16]
1240 [20, 16] 1397 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562
[19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31,
24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [15, 22] 797 [21, 25]
458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532
[45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982
[29, 1] 2029 [6, 50] 176 [43, 46] 307 [7, 2] 942 [16, 11] 650 [23, 31] 2267 [48,
24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [24, 2] 1040
```

[12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [15, 20] 903 [19, 18] 959 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [5, 21] 896 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [5, 23] 737 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [15, 27] 380 [23, 43] 362 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [5, 19] 827 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [5, 7] 583 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [5, 6]

890 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [25, 27] 269 [23, 46] 631 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [27, 44] 496 [7, 28] 339 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [31, 29] 2554 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [7, 33] 394 [27, 20] 533 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [29, 14] 651 [15, 18] 835 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [5, 29] 1045 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [24, 40] 396 [16, 14] 1854 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [18, 19] 961 [7, 14] 1035 [46, 43] 307 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [5, 24] 346 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [5, 27] 327 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [5, 15] 701

[27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;

---

fevereiro.dat
data;
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 18 17 16 19 20 21 22 23 24 26 25 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 ;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 15 37, 16 43, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 25, 19 43, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15 22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 43 46, 7 2, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 25 30, 43 23, 27 37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 15 27, 23 43, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 51 7, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 19 20, 6 7, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27

17, 4 29, 16 50, 47 6, 15 10, 19 9, 43 19, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 19 39, 46 7, 24 22, 24 48, 7 44, 23 37, 21 39, 25 27, 23 46, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 7 33, 27 20, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 29 14, 15 18, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 19 48, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 24 40, 16 14, 51 23, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 46 43, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 27 41, 6 46, 19 31, 16 47, 51 16, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 20 29, 21 41, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 31 6, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;

param: K: R O D:= 1 5081 43 29 2 11119 23 27 3 988 21
49 4 806 29 7 5 2034 23 34 6 5003 21 33 7 628 37
7 8 6538 21 30 9 13575 27 7 10 2412 27 26 11 2734 31
21 12 5771 23 24 13 9500 36 7 14 191 7 43 15 1354 45
27 16 661 6 36 17 4388 43 27 18 2854 7 15 19 14334 15
21 20 3976 17 27 21 17385 48 21 22 565 24 8 23 3333 21
8 24 149 32 21 25 1713 29 26 26 17421 21 2 27 856 21
29 28 4210 32 27 29 2922 29 39 30 17766 29 23 31 478 30
19 32 2405 25 7 33 278 36 19 34 2406 27 49 35 292 45
6 36 723 27 48 37 196 35 27 38 1343 2 21 39 5000 6
21 40 11138 6 2 41 671 49 6 42 1236 46 7 43 10430 27
40 44 4157 27 3 45 5500 49 27 46 453 30 21 47 7112 7
6 48 398 37 44 49 6148 24 29 50 4220 24 14 51 739 49
21 52 3498 16 21 53 3550 16 27 54 5161 47 21 55 1582 25
21 56 9524 27 30 57 285 36 3 58 1016 19 6 59 4767 33

```
      27 60  881    23     37 61  19184  6      24 62  1060   21     20 63  16176  15
      27 64  4489   23     43 65  1677   21     37 66  6895   24     23 67  6329   25
      27 68  7822   27     25 69  2352   24     37 70  2169   12     21 71  5175   27
      15 72  4423   2      27 73  202    36     30 74  779    21     36 75  893    12
      30 76  8031   29     43 77  4974   30     27 78  5579   27     44 79  1886   21
      50 80  361    19     49 81  2043   6      19 82  6692   27     6 83   747    49
      2 84   227    25     44 85  1819   6      44 86  3679   7      8 87   16683  24
      21 88  7341   46     27 89  2279   27     19 90  4063   15     7 91   5827   29
      45 92  2818   21     19 93  292    6      26 94  688    30     6 95   282    27
      16 96  1575   48     27 97  3500   46     43 98  766    19     11 99  8331   27
      18 100 216    30     36 101 13016  27     2 102  10427  21     25 103 1210   21
      47 104 6919   7      24 105 722    48     7 106  209    30     32 107 728    19
      30 108 304    21     43 109 23086  24     27 110 906    25     6 111  2560   29
      9 112  3954   44     37 113 7484   7      23 114 3669   44     24 115 267    20
      24 116 8446   23     7 117  6937   6      25 118 159    15     36 119 5823   27
      51 120 2828   29     42 121 2783   24     44 122 397    49     7 123  305    6
      33 124 7011   43     46 125 6793   21     6 126  532    7      2 127  904    29
      21 128 2599   27     36 129 107    51     27 130 10643  23     29 131 487    27
      46 132 985    27     32 133 2737   6      37 134 10333  27     5 135  172    30
      12 136 1832   6      8 137  764    47     27 138 1697   6      1 139  438    36
      13 140 3277   23     14 141 3565   27     11 142 21527  27     21 143 14628  27
      43 144 455    25     30 145 1293   6      3 146  2628   43     23 147 15861  27
      22 148 4539   27     33 149 10279  27     20 150 4317   6      30 151 7390   19
      27 152 526    24     6 153  150    17     19 154 374    2      6 155  2733   7
      37 156 7892   24     20 157 1678   7      29 158 2694   21     28 159 4857   6
      49 160 22219  7      27 161 6939   27     35 162 702    4      21 163 983    21
      31 164 684    46     23 165 6897   6      15 166 956    45     23 167 956    36
      15 168 312    37     24 169 4574   24     34 170 2701   36     21 171 4844   12
      27 172 6227   27     23 173 9005   36     27 174 3595   47     7 175  262    29
      46 176 3793   21     44 177 5121   7      21 178 9051   27     45 179 3864   15
      33 180 781    2      49 181 3343   29     38 182 6100   27     10 183 426    23
      45 184 12520  29     27 185 3028   24     43 186 490    30     11 187 3698   44
      27 188 1782   21     16 189 638    27     17 190 538    19     2 191  6802   21
      15 192 213    18     27 193 2484   45     41 194 11506  24     7 195  980    46
      29 196 1784   15     6 197  5473   7      25 198 959    43     7 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [15, 37] 643 [16, 43] 2225 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7,
42] 1875 [25, 24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21]
1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502
[15, 42] 1994 [15, 40] 601 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21,
12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13]
519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178
[17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585
[33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24,
25] 459 [49, 27] 468 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47]
```

3001 [44, 7] 396 [46, 21] 2128 [27, 39] 1168 [23, 25] 1187 [19, 43] 1717 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [43, 46] 307 [7, 2] 942 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [25, 30] 324 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [15, 27] 380 [23, 43] 362 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4,

29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [25, 27] 269 [23, 46] 631 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [7, 33] 394 [27, 20] 533 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [29, 14] 651 [15, 18] 835 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 0 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [24, 40] 396 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [46, 43] 307 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [21, 45] 1415 [23, 23] 0 [24, 27]

```
521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318
[15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29,
31] 2575 [29, 42] 516 [6, 31] 711 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47]
663 [51, 16] 1158 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331
[30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [20, 29] 1142 [21,
41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21]
745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504
[23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49]
391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739
[27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27]
765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6,
45] 1410 [24, 43] 1093 [15, 17] 366 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023
[21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24,
39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;
```

```
março.dat
data;
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 18 17 16 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 ;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7,
15 37, 16 43, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21,
16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21
12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25
7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51,
24 25, 49 27, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 25, 19 43,
21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 7 8, 24 21, 15
14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34,
19 8, 15 22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33
29, 19 32, 49 24, 29 1, 6 50, 43 46, 7 2, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3,
35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 43 23, 27
37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24
20, 7 29, 15 9, 47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23,
24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 19, 44 15, 29 38, 29 10, 7
7, 24 13, 23 45, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15
6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21
27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20
15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24
29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16
28, 15 27, 23 43, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 30
```

27, 16 7, 24 1, 19 51, 6 32, 21 50, 51 7, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 19 20, 6 7, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 19 9, 43 19, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 19 39, 46 7, 24 22, 24 48, 7 44, 23 37, 21 39, 25 27, 23 46, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 7 33, 27 20, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 29 14, 15 18, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 35 49, 4 19, 7 6, 37 44, 19 48, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 24 40, 16 14, 51 23, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 46 43, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 27 41, 6 46, 19 31, 16 47, 51 16, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 20 29, 21 41, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 31 6, 32 30, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;

param: K: R O D:= 1 187 43 24 2 5273 43 29 3 12948 23
27 4 2013 21 49 5 2504 23 34 6 6063 21 33 7 1673 37
7 8 9243 21 30 9 22914 27 7 10 3153 27 26 11 3735 31
21 12 6949 23 24 13 6142 36 7 14 1965 7 43 15 1398 45
27 16 653 6 36 17 4863 43 27 18 6109 7 15 19 13235 15
21 20 7058 17 27 21 539 24 8 22 3444 21 8 23 1826 29

```
      26 24  19267  21    2 25   860    21    29 26  6760   32    27 27  3479   29
      39 28  22298  29    23 29  341    30    19 30  3511   25    7 31   237    36
      19 32  2558   27    49 33  3396   27    48 34  182    35    27 35  429    2
      21 36  6669   6     21 37  12001  6     2 38   1051   49    6 39   13460  27
      40 40  5105   27    3 41   8050   49    27 42  196    35    49 43  758    30
      21 44  7702   7     6 45   561    37    44 46  6830   24    29 47  467    7
      44 48  3933   24    14 49  1079   49    21 50  6049   16    21 51  4503   16
      27 52  4014   47    21 53  2456   25    21 54  9694   27    30 55  312    36
      3 56   2111   19    6 57   5579   33    27 58  1123   23    37 59  14101  6
      24 60  21371  15    27 61  2871   23    43 62  2217   21    37 63  6752   24
      23 64  7433   25    27 65  9803   27    25 66  3180   24    37 67  2782   12
      21 68  8276   27    15 69  280    32    7 70   5509   2     27 71  563    36
      30 72  1611   21    36 73  1509   12    30 74  11764  29    43 75  5059   30
      27 76  400    23    21 77  8634   27    44 78  2187   21    50 79  424    19
      49 80  2065   6     19 81  8590   27    6 82   879    49    2 83   2098   6
      44 84  5488   7     8 85   9813   46    27 86  4006   27    19 87  6459   15
      7 88   6408   29    45 89  3153   21    19 90  536    27    16 91  669    48
      27 92  621    19    17 93  387    46    43 94  1135   19    11 95  9777   27
      18 96  397    30    36 97  12787  27    2 98   14164  21    25 99  1686   21
      47 100 9592   7     24 101 1940   48    7 102  815    19    30 103 1037   25
      6 104  2938   29    9 105  396    19    21 106 4835   44    37 107 10080  7
      23 108 4973   44    24 109 986    20    24 110 9911   23    7 111  9670   6
      25 112 418    15    36 113 7427   27    51 114 4139   29    42 115 4619   24
      44 116 552    49    7 117  4018   6     33 118 6571   43    46 119 8593   21
      6 120  1363   7     2 121  160    17    30 122 1707   29    21 123 3670   27
      36 124 138    51    27 125 13586  23    29 126 879    27    46 127 1122   27
      32 128 1768   6     37 129 12868  27    5 130  110    30    12 131 1735   6
      8 132  5187   47    27 133 2664   6     1 134  770    36    13 135 3112   23
      14 136 3375   27    11 137 21189  27    21 138 16039  27    43 139 1281   6
      3 140  388    27    37 141 1242   43    23 142 19426  27    22 143 5470   27
      33 144 284    19    35 145 17659  27    20 146 5217   6     30 147 9377   19
      27 148 3084   24    6 149  235    2     6 150  3581   7     37 151 8167   24
      20 152 1420   7     29 153 3028   21    28 154 4827   6     49 155 9224   27
      35 156 2659   4     21 157 1534   21    31 158 443    47    24 159 5721   6
      15 160 232    45    23 161 2674   36    15 162 242    37    24 163 5670   24
      34 164 3618   36    21 165 7901   12    27 166 6839   27    23 167 13947  36
      27 168 5248   47    7 169  2682   29    46 170 4848   21    44 171 7849   7
      21 172 9766   27    45 173 3189   15    33 174 4216   29    38 175 7184   27
      10 176 427    23    45 177 16553  29    27 178 3329   24    43 179 517    30
      11 180 5209   44    27 181 184    32    30 182 3008   21    16 183 828    27
      17 184 987    19    2 185  8062   21    15 186 155    18    27 187 4335   45
      41 188 16009  24    7 189  5874   46    29 190 623    15    6 191  8646   7
      25 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [15, 37] 643 [16, 43] 2225 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7,
```

42] 1875 [25, 24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15, 40] 601 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459 [49, 27] 468 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46, 21] 2128 [27, 39] 1168 [23, 25] 1187 [19, 43] 1717 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [43, 46] 307 [7, 2] 942 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [15, 27] 380 [23, 43] 362 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31,

7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [25, 27] 269 [23, 46] 631 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [7, 33] 394 [27, 20] 533 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [29, 14] 651 [15, 18] 835 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [35, 49] 250 [4, 19] 813

[7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [24, 40] 396 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [46, 43] 307 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [51, 16] 1158 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [31, 6] 687 [32, 30] 690 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;

---

abril.dat
data;
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 39 38 40 41 42 43 44 45 46 47 48 49 50 51 ;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 16 43, 15 37, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6, 21 2, 23 47, 30 15, 15 13, 20 27, 32 27, 29 39, 50 16, 24 12, 30 19, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 27 12, 36 23, 49 21, 15 2, 29 47, 50 6, 44 7, 46 21, 27 39, 19 43, 23 25, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15 22, 21 25, 46 24, 27 34, 16 31, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 7 2, 16 11, 48 24, 23 31, 16 10, 27 36, 7 3, 35 7, 18 23, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 47 44, 16 37, 48 6, 21 40, 43 23, 27 37, 27 33, 46 14, 50 29, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18,

15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 21 10, 44 47, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 50 21, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 32 36, 19 22, 16 27, 24 14, 47 21, 4 27, 25 21, 6 28, 29 17, 33 27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 30 27, 16 7, 43 38, 24 1, 19 51, 6 32, 21 50, 51 7, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 7 9, 16 1, 6 41, 19 11, 21 17, 6 43, 6 14, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 15 31, 19 20, 6 7, 24 47, 37 15, 16 25, 45 43, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 19 9, 43 19, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 50 7, 50 19, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 50 24, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 19 39, 46 7, 24 22, 7 44, 23 37, 21 39, 25 27, 23 46, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 51 29, 27 44, 7 28, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24, 21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 47 19, 23 29, 50 23, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 7 33, 27 20, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 29 14, 15 18, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 43 6, 27 7, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 35 27, 7 47, 18 24, 7 16, 7 20, 23 28, 2 21, 47 23, 24 38, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 24 40, 16 14, 51 23, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 18 19, 7 14, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 44 24, 27 31, 33 21, 29 42, 29 31, 6 31, 17 30, 27 41, 6 46, 19 31, 51 16, 16 47, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 50 27, 20 29, 21 41, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15

```
24, 15 33, 27 10, 6 45, 24 43, 15 17, 50 15, 31 6, 32 30, 6 6, 37 6, 6 18, 21 16,
18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;
param: K:    R      O     D:= 1  4557   43    29 2   16144  23    27 3   3218   21
       49 4   501    29    7 5    2349   23    34 6   4733   21    33 7   1699   37
       7 8    7461   21    30 9   118    50    21 10  2882   27    26 11  3970   31
       21 12  6735   23    24 13  5509   36    7 14   482    45    29 15  896    7
       43 16  1013   45    27 17  968    6     36 18  5267   43    27 19  4618   7
       15 20  9376   15    21 21  7459   17    27 22  22259  48    21 23  1351   24
       8 24   3528   21    8 25   2135   29    26 26  116    19    36 27  16446  21
       2 28   889    21    29 29  8926   32    27 30  2374   29    39 31  20657  29
       23 32  575    30    19 33  3402   25    7 34   273    36    19 35  3987   27
       49 36  878    24    46 37  296    35    27 38  435    29    24 39  464    2
       21 40  5245   6     21 41  12609  6     2 42   308    49    6 43   1949   46
       7 44   14248  27    40 45  5232   27    3 46   8325   49    27 47  269    30
       21 48  6098   7     6 49   600    37    44 50  7114   24    29 51  130    7
       44 52  193    32    36 53  4633   24    14 54  5079   16    21 55  7586   16
       27 56  2823   47    21 57  482    4     27 58  2419   25    21 59  11632  27
       30 60  202    36    3 61   2469   19    6 62   6021   33    27 63  1361   23
       37 64  22931  6     24 65  20548  15    27 66  1778   23    43 67  2258   21
       37 68  10558  24    23 69  9255   25    27 70  11066  27    25 71  2635   24
       37 72  2258   12    21 73  542    21    35 74  9131   27    15 75  934    32
       7 76   2061   2     27 77  249    36    30 78  759    21    36 79  865    12
       30 80  9388   29    43 81  5385   30    27 82  400    16    7 83   322    43
       38 84  11481  27    44 85  2243   21    50 86  364    19    49 87  2125   6
       19 88  9716   27    6 89   544    49    2 90   1706   6     44 91  6298   7
       8 92   21152  24    21 93  6977   46    27 94  3355   27    19 95  8858   15
       7 96   6054   29    45 97  2779   21    19 98  715    19    11 99  8758   27
       18 100 249    30    36 101 145    37    27 102 19697  27    2 103  12518  21
       25 104 1049   46    24 105 1386   21    47 106 392    16    31 107 8367   7
       24 108 842    43    45 109 701    30    32 110 1054   19    30 111 426    4
       16 112 1448   25    6 113  2450   29    9 114  6091   44    37 115 10822  7
       23 116 5929   44    24 117 568    20    24 118 10402  23    7 119  7709   6
       25 120 224    15    36 121 8480   27    51 122 2874   29    42 123 4662   24
       44 124 397    49    7 125  3353   6     33 126 3498   21    6 127  636    7
       2 128  212    17    30 129 1232   29    21 130 4118   27    36 131 161    51
       27 132 18692  23    29 133 174    18    23 134 1095   27    46 135 1168   27
       32 136 2283   6     37 137 11545  27    5 138  228    30    12 139 3128   6
       8 140  10027  47    27 141 2127   6     1 142  1144   47    44 143 1127   36
       13 144 2811   23    14 145 3125   27    11 146 12603  27    43 147 2062   6
       3 148  2833   43    23 149 18237  27    22 150 22918  6     7 151  1184   46
       14 152 5132   27    33 153 153    7     33 154 12677  27    20 155 3891   6
       30 156 10666  19    27 157 165    45    43 158 3416   24    6 159  116    17
       19 160 1970   2     6 161  3888   7     37 162 9192   24    20 163 3814   7
       29 164 2458   21    28 165 4408   6     49 166 9324   27    35 167 5594   4
       21 168 1453   21    31 169 450    47    24 170 6161   6     15 171 1851   36
       15 172 268    37    24 173 4238   24    34 174 176    44    47 175 2773   36
```

```
      21 176 13069  12     27 177 8081   27     23 178 10684  36     27 179 4609   47
      7 180  14463  29     46 181 107    29     37 182 2792   21     44 183 4427   7
      21 184 9035   27     45 185 3363   15     33 186 3819   29     38 187 8104   27
      10 188 153    23     45 189 14688  29     27 190 1182   24     43 191 1291   30
      11 192 7755   44     27 193 215    32     30 194 2310   21     16 195 1021   27
      17 196 1237   19     2 197  7253   21     15 198 3064   45     41 199 13484  24
      7 200  8976   46     29 201 3094   15     6 202  6836   7      25 203 202    21
      22 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [16, 43] 2225 [15, 37] 643 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25,
24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18]
1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994
[15, 40] 601 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19,
36] 218 [32, 6] 444 [21, 2] 587 [23, 47] 2693 [30, 15] 317 [15, 13] 519 [20, 27]
529 [32, 27] 821 [29, 39] 480 [50, 16] 414 [24, 12] 631 [30, 19] 398 [25, 7] 178
[17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585
[33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24,
25] 459 [49, 27] 468 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47]
3001 [50, 6] 174 [44, 7] 396 [46, 21] 2128 [27, 39] 1168 [19, 43] 1717 [23, 25]
1187 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549
[23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [7, 8] 129 [24, 21] 949
[15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6,
47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21,
34] 1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [27, 34] 723 [16, 31]
253 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122
[33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [7, 2] 942 [16,
11] 650 [48, 24] 905 [23, 31] 2267 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7]
812 [18, 23] 602 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6,
1] 88 [47, 44] 1826 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37]
536 [27, 33] 300 [46, 14] 841 [50, 29] 2004 [23, 49] 1365 [31, 15] 819 [48, 23]
1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903
[21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [44,
29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [44, 47] 1895 [36,
21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803
[7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 19] 280 [44, 15] 715 [29,
38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [29, 27] 1369 [16, 35]
1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498
[37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [50,
21] 80 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11]
483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19]
789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043
[16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809
[36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30,
21] 513 [24, 29] 1173 [27, 50] 637 [32, 36] 180 [19, 22] 840 [16, 27] 983 [24, 14]
522 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [29, 17] 1724 [33, 27] 303
```

[16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [43, 38] 113 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 43] 1811 [6, 14] 1440 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [24, 47] 2011 [37, 15] 639 [16, 25] 866 [45, 43] 404 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [50, 7] 507 [50, 19] 392 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [50, 24] 944 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [25, 27] 269 [23, 46] 631 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [47, 19] 1353 [23, 29] 442 [50, 23] 1695 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [7, 33] 394 [27, 20] 533 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392

```
[27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16,
45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [29, 14] 651 [15, 18]
835 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104
[19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29,
40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [43, 6] 1811 [27, 7]
445 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910
[23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896 [15, 28] 146 [43, 27] 1247
[20, 21] 990 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47]
1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443
[36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [35, 27] 396 [7, 47] 1506
[18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [47, 23] 2694 [24,
38] 979 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813
[7, 6] 489 [37, 44] 122 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3]
168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21,
37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42]
1624 [12, 30] 44 [24, 40] 396 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174
[6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29,
45] 525 [29, 12] 1465 [18, 19] 961 [7, 14] 1035 [19, 17] 89 [24, 11] 1201 [27, 18]
464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415
[23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15]
727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [44, 24] 137 [27, 31] 1210
[33, 21] 317 [29, 42] 516 [29, 31] 2575 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6,
46] 2121 [19, 31] 928 [51, 16] 1158 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3]
778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434
[7, 40] 554 [50, 27] 640 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038
[29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45,
16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31]
1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024
[21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46,
15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495
[4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15,
17] 366 [50, 15] 269 [31, 6] 687 [32, 30] 690 [6, 6] 0 [37, 6] 756 [6, 18] 1023
[21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24,
39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;
```

```
maio.dat
data;
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 5 4 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 36 35 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 ;
```

set A := 43 24, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 16 43, 15 37, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 4, 19 43, 23 25, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 43 39, 20 16, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15 22, 21 25, 46 24, 27 34, 16 31, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 7 2, 16 11, 23 31, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 47 44, 16 37, 21 40, 43 23, 27 37, 27 33, 23 49, 31 15, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 24 4, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 19 28, 21 42, 21 13, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 16 46, 30 27, 16 7, 43 38, 24 1, 19 51, 6 32, 21 50, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 15 4, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 6 7, 19 20, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 43 26, 17 7, 17 21, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 19 9, 43 19, 7 4, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 32 16, 21 4, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 27 4, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 46 7, 24 22, 19 39, 24 48, 7 44, 23 37, 23 46, 25 27, 21 39, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 23 17, 30 36, 27 14, 7 24, 7 17, 19 30, 16 6, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24, 21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 23 32, 16 39, 16 30, 23 16, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 31 4, 23 1, 18 16, 6 4, 29 14, 15 18, 19 4, 35 2, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 45 27, 18 29, 23 13, 46 19, 19 37, 15 28, 43 27, 20 21, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26,

```
20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18
24, 7 20, 7 16, 23 28, 2 21, 47 23, 24 38, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7
6, 37 44, 19 48, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24,
24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 29 4, 24 40, 16 14,
12 23, 19 49, 47 29, 27 6, 6 39, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14,
19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16,
21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 44 24, 27 31, 33 21, 29 42, 29
31, 6 31, 17 30, 27 41, 6 46, 19 31, 16 47, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12,
24 41, 23 41, 17 23, 7 40, 20 29, 21 41, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43
15, 27 22, 45 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50,
18 6, 21 31, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9,
7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 16 4, 31 6, 32 30, 6 6, 37 6, 6
18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 35 23, 23 6, 21 22, 24 39;
param: K:    R     O     D:= 1  5337  43    29 2   13778 23    27 3   3085  21
       49 4   1596  29    7 5    2406  23    34 6   5735  21    33 7   1806  37
       7 8    8484  21    30 9   4129  27    26 10  4807  31    21 11  7279  23
       24 12  7453  36    7 13   1155  45    27 14  838   6     36 15  6734  43
       27 16  6434  7     15 17  7880  15    21 18  9436  17    27 19  993   24
       8 20   5190  6     20 21  4057  21    8 22   446   32    21 23  2396  29
       26 24  215   21    4 25   384   19    36 26  20456 21    2 27   2531  21
       29 28  6997  32    27 29  3599  29    39 30  365   30    19 31  3528  25
       7 32   342   21    5 33   3313  27    49 34  1091  27    48 35  132   35
       27 36  709   29    24 37  668   2     21 38  5781  6     21 39  15097 6
       2 40   16397 27    40 41  6675  27    3 42   7172  49    27 43  545   30
       21 44  6993  7     6 45   682   37    44 46  5320  24    29 47  441   7
       44 48  4199  24    14 49  6444  16    21 50  15079 16    27 51  1061  4
       27 52  3082  25    21 53  12073 27    30 54  645   36    3 55   2275  19
       6 56   8037  33    27 57  1315  23    37 58  22154 6     24 59  1052  21
       20 60  7887  23    43 61  2676  21    37 62  19899 24    23 63  8061  25
       27 64  11207 27    25 65  3890  24    37 66  1703  12    21 67  9558  27
       15 68  2686  2     27 69  137   36    30 70  649   21    36 71  290   12
       30 72  8856  29    43 73  6501  30    27 74  1960  23    21 75  450   43
       38 76  9354  27    44 77  1839  21    50 78  875   43    39 79  251   19
       49 80  2745  6     19 81  7900  27    6 82   778   49    2 83   1821  6
       44 84  7361  7     8 85   3425  46    27 86  4994  27    19 87  9374  15
       7 88   359   6     35 89  8096  29    45 90  2602  21    19 91  229   19
       17 92  798   19    11 93  10904 27    18 94  336   30    36 95  12665 21
       25 96  2226  46    24 97  1876  21    47 98  110   16    31 99  10655 7
       24 100 2628  43    45 101 500   30    32 102 873   19    30 103 211   21
       43 104 488   4     16 105 1901  25    6 106  2637  29    9 107  6952  44
       37 108 6826  44    24 109 4244  20    24 110 14530 23    7 111  11660 7
       23 112 146   33    21 113 8776  6     25 114 248   15    36 115 9148  27
       51 116 3859  29    42 117 3628  24    44 118 750   49    7 119  3918  6
       33 120 1983  21    6 121  237   17    30 122 1114  29    21 123 4291  27
       36 124 20883 23    29 125 1553  27    32 126 2863  6     37 127 13430 27
       5 128  868   30    12 129 3283  6     8 130  8540  47    27 131 2533  6
```

```
      1 132  418    47     44 133 710    36     13 134 3464   23     14 135 3515   27
      11 136 21488  27     21 137 750    21     40 138 15045  27     43 139 1957   6
      3 140  3177   43     23 141 22671  27     22 142 22318  6      7 143  6380   27
      33 144 14742  27     20 145 5777   6      30 146 10036  19     27 147 2640   24
      6 148  770    2      6 149  5898   7      37 150 7063   24     20 151 1180   7
      29 152 2600   21     28 153 5780   6      49 154 9352   27     35 155 3860   4
      21 156 1843   21     31 157 7144   6      15 158 251    45     23 159 165    43
      26 160 2682   36     15 161 662    37     24 162 4924   24     34 163 3494   36
      21 164 10337  27     23 165 15538  36     27 166 6346   47     7 167  12533  29
      46 168 4545   21     44 169 6192   7      21 170 12810  27     45 171 3825   15
      33 172 200    2      19 173 6536   29     38 174 8759   27     10 175 20256  29
      27 176 145    24     43 177 124    31     4 178  513    30     11 179 8431   44
      27 180 301    32     30 181 3141   21     16 182 1046   27     17 183 834    19
      2 184  7744   21     15 185 446    35     2 186  205    18     27 187 4498   45
      41 188 20333  24     7 189  15422  46     29 190 4272   15     6 191  9402   7
      25 192 513    21     22 ;
param distancia default 0 := [43, 24] 1092 [6, 27] 569 [43, 29] 118 [23, 27] 1061
[21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909 [37, 7] 274 [16,
43] 2225 [15, 37] 643 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24]
460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437
[6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15,
40] 601 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36]
218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529
[32, 27] 821 [29, 39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37,
16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33, 19] 326 [29,
29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459 [49, 27] 468
[27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46,
21] 2128 [27, 39] 1168 [23, 4] 2156 [19, 43] 1717 [23, 25] 1187 [21, 20] 994 [24,
23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21]
1630 [44, 16] 1240 [43, 39] 359 [20, 16] 1397 [7, 8] 129 [24, 21] 949 [15, 14] 1251
[15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49,
29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8]
621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [27, 34] 723 [16, 31] 253 [15, 3] 253
[6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667
[19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [7, 2] 942 [16, 11] 650 [23, 31]
2267 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457
[24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [47, 44]
1826 [16, 37] 1120 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49]
1365 [31, 15] 819 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959
[15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7,
27] 440 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36,
21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803
[7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 19] 280 [44, 15] 715 [29,
38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [24, 4] 1474 [29, 27]
1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234
[36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16,
```

24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [43, 38] 113 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [6, 7] 490 [19, 20] 1027 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [43, 26] 248 [17, 7] 710 [17, 21] 401 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [7, 4] 969 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [27, 4] 1099 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [46, 7] 1814 [24, 22] 922 [19, 39] 1598 [24, 48] 917 [7, 44] 394 [23, 37] 975 [23, 46] 631 [25, 27] 269 [21, 39] 1744 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [23, 17] 1434 [30, 36] 611 [27, 14] 881 [7, 24] 511 [7, 17] 708 [19, 30] 396 [16, 6] 480 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23]

1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [23, 32] 1764 [16, 39] 2152 [16, 30] 921 [23, 16] 2038 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [31, 4] 97 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [15, 18] 835 [19, 4] 817 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 20] 644 [7, 16] 851 [23, 28] 1543 [2, 21] 592 [47, 23] 2694 [24, 38] 979 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [29, 4] 2464 [24, 40] 396 [16, 14] 1854 [12, 23] 1157 [19, 49] 174 [47, 29] 3002 [27, 6] 567 [6, 39] 1738 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [44, 24] 137 [27, 31] 1210 [33, 21] 317 [29, 42] 516 [29, 31] 2575 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [16, 4] 142 [31, 6] 687 [32, 30] 690 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 [24, 39] 1323 ;
end;

---

junho.dat
data;

```
param n := 51;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 ;
set A := 43 24, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 16 43,
15 37, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23
20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6,
23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16,
15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 27
12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 4, 19 43, 23 25, 21 20, 24
23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 25 44, 7 8, 24 21, 15
14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34,
19 8, 15 22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33
29, 19 32, 49 24, 29 1, 6 50, 7 2, 36 32, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3,
35 7, 18 23, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 43 23, 27 37,
27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7
29, 15 9, 47 16, 7 27, 44 29, 47 24, 15 23, 24 17, 45 26, 21 10, 36 21, 2 15, 27
23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38,
29 10, 7 7, 24 13, 24 4, 23 45, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36
16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15
11, 15 50, 21 27, 45 15, 29 28, 16 38, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16
42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30
21, 24 29, 27 50, 32 36, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 29 17, 33
27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16
46, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21
19, 15 4, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27
2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6,
30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 19 19, 24 32, 6 42, 49 7, 12 24, 6 33,
44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8, 47 27, 16 22, 23 14, 16 5, 27 21, 31
7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 6 7, 19 20, 37 15, 16 25, 46 16, 29 32,
33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17
21, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7
49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 15 51, 27 17, 4 29, 16 50, 47
6, 15 10, 19 9, 43 19, 7 4, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27
38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7
43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 25 20, 32 16, 21 4, 21 38, 36
24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 27 4, 21 14, 16 26, 29 8, 45 24,
6 21, 33 24, 49 6, 19 39, 46 7, 24 22, 7 44, 23 37, 21 39, 23 46, 25 27, 19 7, 21
35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 27 44, 7 28, 16 2,
19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 48 27, 23 17, 30 36, 27 14, 7
24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16
12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21
6, 17 29, 19 29, 31 29, 47 19, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18,
21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19
```

15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 6 4, 29 14, 15 18, 19 4, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 19 37, 15 28, 43 27, 20 21, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 7 47, 35 27, 18 24, 7 20, 7 16, 23 28, 2 21, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 29 4, 24 40, 16 14, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 18 19, 7 14, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 27 41, 6 46, 19 31, 16 47, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 20 29, 21 41, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 16 4, 31 6, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;

param: K: R O D:= 1 5727 43 29 2 11303 23 27 3 4308 21
49 4 3118 23 34 5 5128 21 33 6 850 37 7 7 8196 21
30 8 22142 27 7 9 3039 27 26 10 8561 31 21 11 5228 25
24 12 7550 23 24 13 7972 36 7 14 149 7 43 15 802 45
27 16 762 6 36 17 217 49 19 18 4211 43 27 19 6002 7
15 20 6205 15 21 21 388 25 20 22 7470 17 27 23 21271 48
21 24 633 24 8 25 2135 6 20 26 4184 21 8 27 2767 29
26 28 108 21 4 29 20517 21 2 30 3019 21 29 31 3958 32
27 32 3065 29 39 33 20510 29 23 34 157 30 19 35 3036 25
7 36 2657 27 49 37 116 35 27 38 4664 29 24 39 306 2
21 40 5464 6 21 41 14400 6 2 42 1499 24 25 43 14484 27
40 44 6228 27 3 45 8069 49 27 46 936 30 21 47 10479 7
6 48 612 37 44 49 5924 24 29 50 334 32 36 51 4032 24
14 52 591 49 21 53 3621 16 21 54 9894 16 27 55 1256 47
21 56 1471 4 27 57 2650 25 21 58 10443 27 30 59 364 36
3 60 2400 19 6 61 6869 33 27 62 618 23 37 63 21159 6
24 64 1585 21 20 65 21933 15 27 66 4099 23 43 67 2726 21
37 68 15592 24 23 69 8241 25 27 70 10000 27 25 71 3370 24
37 72 2541 12 21 73 7812 27 15 74 1015 32 7 75 1818 2
27 76 206 36 30 77 827 21 36 78 281 12 30 79 7332 29
43 80 5636 30 27 81 792 23 21 82 7576 27 44 83 2220 21
50 84 178 19 49 85 2126 6 19 86 7978 27 6 87 737 49
2 88 532 25 44 89 4026 6 44 90 7842 7 8 91 3146 46
27 92 4888 27 19 93 8598 15 7 94 7311 29 45 95 2143 21
19 96 333 19 17 97 531 19 11 98 10229 27 18 99 336 30
36 100 108 37 27 101 17964 27 2 102 14048 21 25 103 2440 21
47 104 9939 7 24 105 1512 43 45 106 3144 48 7 107 524 30

```
      32 108 1042   19     30 109 677    4      16 110 935    25     6 111  2329   29
      9 112  5338   44     37 113 11063  7      23 114 4749   44     24 115 493    20
      24 116 6525   23     7 117  9613   6      25 118 384    15     36 119 7209   27
      51 120 3888   29     42 121 3416   24     44 122 718    49     7 123  3342   6
      33 124 2958   21     6 125  888    7      2 126  128    36     32 127 168    17
      30 128 354    29     21 129 5529   27     36 130 18823  23     29 131 1721   27
      32 132 3377   6      37 133 13415  27     5 134  330    30     12 135 2533   6
      8 136  5766   47     27 137 2655   6      1 138  842    36     13 139 3452   23
      14 140 4062   27     11 141 22861  27     21 142 15142  27     43 143 1620   6
      3 144  187    27     37 145 2517   43     23 146 19182  27     22 147 23805  6
      7 148  4176   27     33 149 545    7      33 150 8650   27     20 151 5181   6
      30 152 10228  19     27 153 1600   24     6 154  198    17     19 155 304    2
      6 156  4619   7      37 157 9970   24     20 158 1905   7      29 159 2400   21
      28 160 4325   6      49 161 8545   27     35 162 3490   4      21 163 1374   21
      31 164 141    33     6 165  8186   6      15 166 389    45     23 167 337    37
      21 168 1948   36     15 169 465    45     26 170 357    37     24 171 5684   24
      34 172 3778   36     21 173 13624  12     27 174 10318  27     23 175 12951  36
      27 176 5054   47     7 177  11729  29     46 178 6210   21     44 179 10327  7
      21 180 11653  27     45 181 4431   15     33 182 562    2      49 183 3893   29
      38 184 8778   27     10 185 149    23     45 186 17755  29     27 187 543    24
      43 188 346    30     11 189 5469   44     27 190 165    16     4 191  2523   21
      16 192 656    27     17 193 686    19     2 194  7004   21     15 195 181    18
      27 196 3157   45     41 197 22711  24     7 198  11836  46     29 199 2573   15
      6 200  7404   7      25 ;
param distancia default 0 := [43, 24] 1092 [6, 27] 569 [43, 29] 118 [23, 27] 1061
[21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909 [37, 7] 274 [16,
43] 2225 [15, 37] 643 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24] 460 [36, 7]
762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267
[23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15, 40] 601 [17,
16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444
[23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29,
39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29]
1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33, 19] 326 [29, 29] 0 [27, 29] 1373
[21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459 [49, 27] 468 [27, 12] 114 [36,
23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46, 21] 2128 [27, 39]
1168 [23, 4] 2156 [19, 43] 1717 [23, 25] 1187 [21, 20] 994 [24, 23] 735 [23, 2]
1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16]
1240 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373
[6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4,
23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [15, 22]
797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624
[7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314
[49, 24] 982 [29, 1] 2029 [6, 50] 176 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31]
2267 [48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602
[24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37]
1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365
```

[31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [45, 26] 156 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [24, 4] 1474 [23, 45] 235 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [32, 36] 180 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [19, 19] 0 [24, 32] 1338 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [6, 7] 490 [19, 20] 1027 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [19, 9] 1957 [43, 19] 1717 [7, 4] 969 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [27, 4] 1099 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [23, 46] 631 [25, 27] 269 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496

[21, 36] 330 [29, 43] 118 [27, 44] 496 [7, 28] 339 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [31, 29] 2554 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [15, 18] 835 [19, 4] 817 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 20] 644 [7, 16] 851 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [29, 4] 2464 [24, 40] 396 [16, 14] 1854 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [18, 19] 961 [7, 14] 1035 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [16, 4] 142 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405

```
[18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35,
23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;
```

```
julho.dat
data;
param n := 52;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 5 4 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52;
set A := 43 24, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 16 43,
15 37, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23
20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6,
23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16,
15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 27
12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 4, 19 43, 23 25, 21 20, 24
23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 43 39, 49 35, 20 16, 25 44, 7
8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21
23, 21 34, 19 8, 15 22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15,
44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 7 2, 36 32, 16 11, 23 31, 48 24, 16 10, 27
36, 7 3, 35 7, 18 23, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 43 23,
27 37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24
20, 7 29, 15 9, 47 16, 7 27, 52 19, 44 29, 47 24, 15 23, 10 15, 24 17, 21 10, 36
21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19,
44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 24 4, 52 24, 29 27, 16 35, 24 10, 16 13, 10
19, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23
36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 48 29, 19 28, 21
42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29
24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25
21, 6 28, 29 17, 33 27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2
27, 35 19, 48 19, 16 46, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 52 6, 52 15, 6 10,
19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 15 4, 7 9, 16 1, 6 41, 21 17, 19 11, 6 14, 6
43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25,
15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 6 25, 21 7, 19 19,
24 32, 10 21, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 27 46, 15 8,
47 27, 10 27, 16 22, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15
31, 19 20, 6 7, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 10 16, 21
24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 17 7, 17 21, 15 1, 2 24, 12 27, 29 33,
32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19,
15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 52 21, 19 9, 43
19, 7 4, 37 19, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21
30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24
```

```
42, 52 36, 15 34, 16 20, 15 21, 29 3, 25 20, 32 16, 21 4, 21 38, 36 24, 19 46, 27
28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 27 4, 16 26, 29 8, 45 24, 6 21, 33 24,
49 6, 46 7, 24 22, 19 39, 7 44, 23 37, 23 46, 25 27, 21 39, 19 7, 21 35, 32 7, 19
42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 27 44, 7 28, 16 2, 19 47, 49 2,
7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 48 27, 23 17, 30 36, 27 14, 7 24, 7 17, 48
7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24,
21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19
29, 31 29, 47 19, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24,
24 3, 19 35, 27 20, 7 33, 23 32, 16 39, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21
28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 29 13, 6 15, 32 29, 37 21, 36
15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 52
7, 15 32, 15 39, 19 34, 16 45, 24 18, 10 7, 10 29, 15 16, 31 4, 23 1, 18 16, 6 4,
29 14, 52 27, 15 18, 19 4, 35 2, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23
42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 6 11, 21 33, 27 7, 43 6, 24
15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 19 37, 15 28, 43 27, 20 21, 52
29, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8,
36 29, 10 24, 24 51, 25 19, 24 46, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 24
38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 24 35, 6 17, 16 21, 20
19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32,
15 30, 27 42, 12 30, 29 4, 24 40, 16 14, 12 23, 19 49, 47 29, 27 6, 6 39, 6 44, 32
11, 36 6, 29 5, 29 45, 29 12, 18 19, 7 14, 24 11, 19 17, 27 18, 44 19, 35 24, 21
47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35,
7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 27 41, 6 46, 19 31, 16
47, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 7 40, 20 29, 21 41,
35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23
33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 52 16, 17 44, 21 21,
45 23, 10 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15
24, 15 33, 27 10, 6 45, 24 43, 15 17, 16 4, 31 6, 32 30, 6 6, 37 6, 6 18, 21 16, 18
27, 16 17, 10 6, 20 23, 7 25, 29 30, 35 23, 23 6, 21 22, 24 39;
param: K:    R     O     D:= 1  6723   43    29 2   16430  23    27 3   3126   21
       49 4   1943  29    7 5    3866   23    34 6   7619   21    33 7   1233   37
       7 8    9960  21    30 9   3407   27    26 10  13953  31    21 11  6979   25
       24 12  10192 23    24 13  14723  36    7 14   135    45    29 15  108    7
       43 16  961   45    27 17  685    6     36 18  159    49    19 19  195    52
       36 20  5832  43    27 21  5362   7     15 22  6144   15    21 23  813    25
       20 24  9485  17    27 25  21809  48    21 26  1359   24    8 27   5657   6
       20 28  4841  21    8 29   3112   29    26 30  21283  21    2 31   1864   21
       29 32  4755  32    27 33  4504   29    39 34  328    30    19 35  3012   25
       7 36   625   36    19 37  3338   27    49 38  873    24    46 39  4199   29
       24 40  294   2     21 41  6710   6     21 42  16044  6     2 43   1846   24
       25 44  15303 27    40 45  5256   27    3 46   9086   49    27 47  518    30
       21 48  11183 7     6 49   776    37    44 50  9695   24    29 51  228    7
       44 52  4842  24    14 53  3578   16    21 54  5082   16    27 55  131    19
       3 56   2107  47    21 57  551    4     27 58  2979   25    21 59  10227  27
       30 60  412   36    3 61   2296   19    6 62   7498   33    27 63  1129   23
       37 64  3664  23    43 65  4012   21    37 66  13242  24    23 67  8807   25
```

```
      27 68  11287  27     25 69  2723   24     37 70  4689   12     21 71  9800   27
      15 72  1442   2      27 73  113    15     30 74  317    36     30 75  568    21
      36 76  369    12     30 77  11982  29     43 78  6865   30     27 79  1101   23
      21 80  10807  27     44 81  3068   21     50 82  248    43     39 83  316    49
      35 84  438    19     49 85  2137   6      19 86  11177  27     6 87   1071   49
      2 88   819    25     44 89  2739   6      44 90  142    32     11 91  7048   7
      8 92   6303   46     27 93  4648   27     19 94  10491  15     7 95   8936   29
      45 96  2639   21     19 97  582    19     17 98  1134   19     11 99  10527  27
      18 100 497    30     36 101 14374  21     25 102 3218   46     24 103 2480   21
      47 104 10010  7      24 105 1573   43     45 106 630    30     32 107 976    19
      30 108 1433   4      16 109 1141   25     6 110  2364   29     9 111  7367   44
      37 112 11374  7      23 113 6558   44     24 114 510    20     24 115 10611  23
      7 116  10675  6      25 117 553    15     36 118 8170   27     51 119 3484   29
      42 120 5661   24     44 121 830    49     7 122  446    44     21 123 4590   6
      33 124 2602   21     6 125  219    7      2 126  387    36     32 127 185    17
      30 128 5217   27     36 129 18244  23     29 130 104    18     23 131 1961   27
      32 132 3933   6      37 133 12362  27     5 134  353    30     12 135 2706   6
      8 136  7158   47     27 137 103    10     27 138 3383   6      1 139  972    36
      13 140 4141   23     14 141 4603   27     11 142 13929  27     43 143 2295   6
      3 144  2705   43     23 145 19831  27     22 146 5341   27     33 147 18296  27
      20 148 6472   6      30 149 10493  19     27 150 1481   24     6 151  727    17
      19 152 640    2      6 153  5107   7      37 154 6285   24     20 155 2681   7
      29 156 3316   21     28 157 5428   6      49 158 9384   27     35 159 4560   4
      21 160 1646   21     31 161 309    52     16 162 405    17     44 163 235    33
      6 164  8210   6      15 165 524    45     23 166 660    37     21 167 2832   36
      15 168 372    37     24 169 6701   24     34 170 4256   36     21 171 20694  12
      27 172 10684  27     23 173 13370  36     27 174 6373   47     7 175  12061  29
      46 176 10661  21     44 177 8425   7      21 178 10797  27     45 179 4100   15
      33 180 1388   2      49 181 140    2      19 182 5862   29     38 183 9296   27
      10 184 122    23     45 185 21569  29     27 186 368    24     43 187 423    31
      4 188  560    30     11 189 8389   44     27 190 153    32     30 191 2985   21
      16 192 991    27     17 193 1384   19     2 194  10114  21     15 195 251    35
      2 196  155    18     27 197 3761   45     41 198 17188  46     29 199 4577   15
      6 200  5955   7      25 ;
param distancia default 0 := [43, 24] 1092 [6, 27] 569 [43, 29] 118 [23, 27] 1061
[21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909 [37, 7] 274 [16,
43] 2225 [15, 37] 643 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24] 460 [36, 7]
762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267
[23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15, 40] 601 [17,
16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444
[23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29,
39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29]
1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33, 19] 326 [29, 29] 0 [27, 29] 1373
[21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459 [49, 27] 468 [27, 12] 114 [36,
23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46, 21] 2128 [27, 39]
1168 [23, 4] 2156 [19, 43] 1717 [23, 25] 1187 [21, 20] 994 [24, 23] 735 [23, 2]
```

1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [43, 39] 359 [49, 35] 270 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [10, 15] 438 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [24, 4] 1474 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [10, 19] 556 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [21, 17] 398 [19, 11] 205 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [6, 25] 534 [21, 7] 441 [19, 19] 0 [24, 32] 1338 [10, 21] 633 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [10, 27] 66 [16, 22] 1400 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [10, 16] 1042 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [17, 7] 710 [17, 21] 401 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17]

441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [52, 21] 538 [19, 9] 1957 [43, 19] 1717 [7, 4] 969 [37, 19] 897 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [52, 36] 283 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [27, 4] 1099 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [46, 7] 1814 [24, 22] 922 [19, 39] 1598 [7, 44] 394 [23, 37] 975 [23, 46] 631 [25, 27] 269 [21, 39] 1744 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [27, 44] 496 [7, 28] 339 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [31, 29] 2554 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [23, 32] 1764 [16, 39] 2152 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [10, 7] 501 [10, 29] 1352 [15, 16] 612 [31, 4] 97 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [19, 4] 817 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [6, 11] 351 [21, 33] 317 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [10, 24] 578 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [29, 4] 2464 [24, 40] 396 [16, 14] 1854 [12, 23] 1157 [19, 49] 174 [47, 29] 3002 [27, 6] 567 [6, 39] 1738 [6, 44] 811 [32, 11] 143 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [18, 19] 961 [7, 14] 1035 [24, 11] 1201 [19, 17] 89 [27, 18] 464 [44, 19] 930 [35,

24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [7, 40] 554 [20, 29] 1142 [21, 41] 1569 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [52, 16] 388 [17, 44] 869 [21, 21] 0 [45, 23] 236 [10, 23] 1043 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [16, 4] 142 [31, 6] 687 [32, 30] 690 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [10, 6] 627 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 [24, 39] 1323 ;
end;

---

agosto.dat
data;
param n := 52;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 52;
set A := 43 24, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 15 37, 16 43, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 24 52, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 28 29, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 27 39, 23 25, 19 43, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15 22, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 23 52, 7 12, 45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 52, 6 50, 7 2, 36 32, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 47 44, 16 37, 48 6, 21 40, 43 23, 27 37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 52 19, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 52 24, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27

49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 15 27, 23 43, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 29 52, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 52 6, 52 15, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 28 6, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 6 25, 21 7, 19 19, 24 32, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 28 7, 29 21, 7 38, 15 52, 27 46, 15 8, 47 27, 16 22, 28 24, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 6 7, 19 20, 37 15, 16 25, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 43 26, 17 7, 17 21, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 52 21, 19 9, 43 19, 37 19, 27 52, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 16 52, 15 21, 29 3, 25 20, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 21 14, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 19 39, 46 7, 24 22, 24 48, 7 44, 23 37, 21 39, 23 46, 25 27, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 28 15, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 28 21, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 19 52, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 52 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 29 14, 52 27, 15 18, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 19 37, 15 28, 43 27, 20 21, 52 29, 24 8, 35 6, 21 8, 6 13, 19 41, 27 47, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 20, 7 16, 23 28, 2 21, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 35 49, 4 19, 7 6, 37 44, 19 48, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 24 40, 7 52, 16 14, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 21 52, 46 43, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 27 41, 6 46, 19 31, 16 47, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 24 41, 23 41, 17 23, 28 23, 7 40, 20 29, 21 41, 28 16, 35 15, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 28 27, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 52 16, 28 19, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 12 44, 24 43, 15 17, 31 6, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;

param: K: R O D:= 1 368 43 24 2 463 12 6 3 4290 43
29 4 18122 23 27 5 3133 21 49 6 2294 29 7 7 3328 23

34	8	1147	7	46	9	6116	21	33	10	1191	37	7	11	9898	21
30	12	4016	27	26	13	14281	31	21	14	6003	25	24	15	7352	23
24	16	14496	36	7	17	442	7	43	18	2038	45	27	19	374	6
36	20	276	49	19	21	4131	43	27	22	136	16	52	23	4693	7
15	24	8388	15	21	25	838	31	27	26	524	25	20	27	7982	17
27	28	780	24	8	29	3018	6	20	30	6432	21	8	31	114	32
21	32	2219	29	26	33	149	19	36	34	21159	21	2	35	3700	21
29	36	6283	32	27	37	3754	29	39	38	317	30	19	39	2591	25
7	40	195	36	19	41	3348	27	49	42	345	27	48	43	215	35
27	44	3265	29	24	45	333	2	21	46	8729	6	21	47	14323	6
2	48	1401	24	25	49	1338	46	7	50	15273	27	40	51	7046	27
3	52	9298	49	27	53	208	35	49	54	405	30	21	55	9278	7
6	56	677	37	44	57	12431	24	29	58	345	7	44	59	5089	24
14	60	4277	16	21	61	5455	16	27	62	1248	47	21	63	3015	25
21	64	12500	27	30	65	576	36	3	66	2027	19	6	67	7670	33
27	68	792	23	37	69	23007	15	27	70	5776	23	43	71	3928	21
37	72	9632	24	23	73	8039	25	27	74	11759	27	25	75	1886	24
37	76	3426	12	21	77	8719	27	15	78	182	2	27	79	322	36
30	80	1436	21	36	81	4141	12	30	82	11192	29	43	83	6489	30
27	84	1348	23	21	85	7785	27	44	86	3408	21	50	87	442	19
49	88	2322	6	19	89	10374	27	6	90	2004	49	2	91	670	25
44	92	2195	6	44	93	6728	7	8	94	126	36	6	95	5285	27
19	96	7823	15	7	97	10437	29	45	98	3217	21	19	99	1605	48
27	100	347	19	17	101	164	46	43	102	1130	19	11	103	8692	27
18	104	233	30	36	105	18945	27	2	106	14204	21	25	107	4024	46
24	108	3408	21	47	109	10924	7	24	110	641	43	45	111	4368	48
7	112	1888	19	30	113	1056	30	32	114	21630	24	27	115	1273	25
6	116	3551	29	9	117	5838	44	37	118	9782	7	23	119	5261	44
24	120	726	20	24	121	11472	23	7	122	10421	6	25	123	365	15
36	124	7539	27	51	125	4041	29	42	126	6708	24	44	127	1312	49
7	128	3888	6	33	129	2446	21	6	130	234	36	32	131	249	17
30	132	834	29	21	133	5751	27	36	134	13266	23	29	135	1673	27
32	136	3229	6	37	137	12636	27	5	138	422	30	12	139	127	28
21	140	4153	6	8	141	10510	47	27	142	2944	6	1	143	408	47
44	144	946	36	13	145	4791	23	14	146	4459	27	11	147	16320	27
43	148	2100	6	3	149	322	27	37	150	1782	43	23	151	20871	27
22	152	6649	27	33	153	18392	27	20	154	5647	6	30	155	10698	19
27	156	948	24	6	157	205	17	19	158	909	2	6	159	5049	7
37	160	6008	24	20	161	2812	7	29	162	3076	21	28	163	5182	6
49	164	9462	27	35	165	6339	4	21	166	2116	21	31	167	146	52
16	168	146	33	15	169	186	33	6	170	7385	6	15	171	1070	45
23	172	574	37	21	173	235	43	26	174	417	17	7	175	3124	36
15	176	257	37	24	177	6260	24	34	178	5364	36	21	179	17301	12
27	180	10690	27	23	181	17521	36	27	182	9497	47	7	183	12227	29
46	184	313	29	37	185	4311	21	44	186	9476	7	21	187	13281	27
45	188	3164	15	33	189	1583	2	49	190	5558	29	38	191	7942	27

```
      10 192 102    25     15 193 130    23     45 194 21552  29     27 195 461    12
      44 196 2083   24     43 197 357    30     11 198 7419   44     27 199 3749   21
      16 200 634    27     17 201 907    19     2 202  8355   21     15 203 171    18
      27 204 4775   45     41 205 19120  24     7 206  15163  46     29 207 4764   15
      6 208  6067   7      25 ;
param distancia default 0 := [43, 24] 1092 [6, 27] 569 [43, 29] 118 [23, 27] 1061
[21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909 [37, 7] 274 [15,
37] 643 [16, 43] 2225 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24]
460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437
[6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [15,
40] 601 [17, 16] 781 [24, 52] 1472 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12]
549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519
[20, 27] 529 [32, 27] 821 [29, 39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17,
15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33,
19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24, 25] 459
[49, 27] 468 [28, 29] 1860 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29,
47] 3001 [44, 7] 396 [46, 21] 2128 [27, 39] 1168 [23, 25] 1187 [19, 43] 1717 [21,
20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50]
1694 [23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949
[15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6,
47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21,
34] 1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34]
723 [15, 3] 253 [6, 23] 1624 [23, 52] 2009 [7, 12] 532 [45, 19] 1318 [47, 15] 1267
[44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 52] 521 [6,
50] 176 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10]
1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12,
7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [47, 44] 1826 [16, 37] 1120
[48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31,
15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18]
959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662
[7, 27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958
[21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33,
16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49]
107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23,
45] 235 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2]
278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267
[19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617
[15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222
[29, 28] 1851 [16, 38] 2153 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459
[48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35,
16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38]
248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522
[16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29,
17] 1724 [33, 27] 303 [16, 28] 509 [15, 27] 380 [23, 43] 362 [49, 16] 867 [12, 21]
549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330
[16, 46] 2536 [29, 52] 2299 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6,
```

32] 448 [21, 50] 81 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [28, 6] 153 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [6, 25] 534 [21, 7] 441 [19, 19] 0 [24, 32] 1338 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [28, 7] 341 [29, 21] 1938 [7, 38] 1432 [15, 52] 728 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [28, 24] 778 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [6, 7] 490 [19, 20] 1027 [37, 15] 639 [16, 25] 866 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [43, 26] 248 [17, 7] 710 [17, 21] 401 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [52, 21] 538 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [27, 52] 1097 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [16, 52] 387 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [23, 46] 631 [25, 27] 269 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [28, 15] 146 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [28, 21] 104 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [19, 52] 469 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612

[23, 1] 1720 [18, 16] 1438 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [19, 41] 1260 [27, 47] 1636 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 20] 644 [7, 16] 851 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [35, 49] 250 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [24, 40] 396 [7, 52] 967 [16, 14] 1854 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [21, 52] 535 [46, 43] 307 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [24, 41] 1444 [23, 41] 864 [17, 23] 1434 [28, 23] 1552 [7, 40] 554 [20, 29] 1142 [21, 41] 1569 [28, 16] 512 [35, 15] 559 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [28, 27] 512 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [52, 16] 388 [28, 19] 377 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [12, 44] 602 [24, 43] 1093 [15, 17] 366 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;
end;

---

setembro.dat

```
data;
param n := 52;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 52;
```

set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 16 43, 15 37, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 17 27, 44 6, 32 21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12, 30 19, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 24 25, 49 27, 28 29, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 23 4, 27 39, 19 43, 23 25, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 43 39, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12, 45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 43 46, 7 2, 36 32, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 6 1, 16 37, 48 6, 21 40, 43 23, 27 37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 52 19, 44 29, 47 24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 24 4, 23 45, 52 24, 29 27, 16 35, 24 10, 16 13, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 30 27, 16 7, 43 38, 24 1, 19 51, 6 32, 21 50, 51 7, 52 6, 52 15, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 15 4, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 28 6, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 19 19, 24 32, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 28 7, 29 21, 7 38, 27 46, 15 8, 47 27, 28 24, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 6 7, 19 20, 37 15, 16 25, 33 2, 45 43, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 43 26, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 52 21, 7 4, 19 9, 43 19, 37 19, 43 41, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 25 20, 32 16, 21 4, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 24 33, 23 12, 27 4, 21 14, 16 26, 29 8, 45 24, 6 21, 33 24, 49 6, 46 7, 19 39, 24 48, 7 44, 23 37, 25 27, 23 46, 21 39, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 28 15, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24, 21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 28 21, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 23 32, 16 39, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 52 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 6 4, 29 14, 52 27, 15 18, 19 4, 35 2, 29 20, 19 26, 45 41, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36,

19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 52 29, 24 8, 35 6, 21 8, 6 13, 27 47, 29 26, 19 41, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 20, 7 16, 23 28, 2 21, 47 23, 24 38, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 19 48, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 29 4, 27 42, 12 30, 24 40, 16 14, 51 23, 12 23, 19 49, 47 29, 27 6, 6 39, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 6 46, 27 41, 19 31, 16 47, 51 16, 27 32, 16 3, 29 49, 27 5, 30 12, 17 23, 23 41, 24 41, 28 23, 7 40, 20 29, 28 16, 35 15, 21 41, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 28 27, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 52 16, 28 19, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 16 4, 31 6, 32 30, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 35 23, 24 39, 23 6;

param: K: R O D:= 1 1306 12 6 2 17080 23 27 3 3089 21
49 4 6625 29 7 5 4003 23 34 6 6828 21 33 7 1462 37
7 8 9393 21 30 9 3172 27 26 10 19547 31 21 11 6255 25
24 12 11648 23 24 13 8315 36 7 14 1141 45 27 15 828 6
36 16 200 49 19 17 4682 43 27 18 4646 7 15 19 7525 15
21 20 1365 31 27 21 579 25 20 22 8508 17 27 23 1617 24
8 24 2603 6 20 25 5998 21 8 26 2070 29 26 27 412 21
4 28 135 19 36 29 21057 21 2 30 1236 21 29 31 7359 32
27 32 3202 29 39 33 615 30 19 34 2792 25 7 35 3668 27
49 36 905 24 46 37 363 27 48 38 882 29 24 39 487 2
21 40 13390 6 21 41 18684 6 2 42 1766 24 25 43 3638 46
7 44 16372 27 40 45 6358 27 3 46 11839 49 27 47 415 30
21 48 9450 7 6 49 403 37 44 50 14320 24 29 51 496 7
44 52 5167 24 14 53 908 49 21 54 7223 16 21 55 10531 16
27 56 2957 47 21 57 2392 25 21 58 12431 27 30 59 298 36
3 60 4513 19 6 61 7623 33 27 62 1361 23 37 63 6728 23
43 64 4946 21 37 65 6114 24 23 66 7039 25 27 67 15811 27
25 68 1928 24 37 69 7212 12 21 70 10750 27 15 71 835 32
7 72 892 2 27 73 185 36 30 74 589 21 36 75 10087 12
30 76 6468 30 27 77 751 23 21 78 363 43 38 79 11046 27
44 80 2699 21 50 81 561 43 39 82 375 19 49 83 2945 6
19 84 10361 27 6 85 1051 49 2 86 783 25 44 87 2511 6
44 88 7231 7 8 89 502 21 3 90 9997 46 27 91 5187 27
19 92 7689 15 7 93 8824 29 45 94 2710 21 19 95 503 48
27 96 376 19 17 97 868 19 11 98 11566 27 18 99 367 30
36 100 18074 27 2 101 14867 21 25 102 902 46 24 103 5717 21
47 104 11169 7 24 105 3792 43 45 106 6079 48 7 107 794 30
32 108 1654 19 30 109 395 21 43 110 23687 24 27 111 1293 25
6 112 3165 29 9 113 202 47 15 114 5898 44 37 115 14058 7
23 116 5971 44 24 117 498 20 24 118 12973 23 7 119 13159 6

```
      25 120 496    15     36 121 8733   27     51 122 4788   29     42 123 6203   24
      44 124 217    44     21 125 3808   6      33 126 255    43     46 127 3029   21
      6 128  1441   7      2 129  253    36     32 130 260    17     30 131 1458   29
      21 132 5086   27     36 133 172    51     27 134 13819  23     29 135 1411   27
      32 136 4401   6      37 137 12517  27     5 138  745    30     12 139 142    28
      21 140 5409   6      8 141  13539  47     27 142 3378   6      1 143  903    36
      13 144 3564   23     14 145 4466   27     11 146 21407  27     43 147 2389   6
      3 148  372    27     37 149 3039   43     23 150 7640   27     33 151 20235  27
      20 152 6428   6      30 153 182    33     2 154  13326  19     27 155 369    45
      43 156 4234   24     6 157  420    17     19 158 332    2      6 159  5844   7
      37 160 5863   24     20 161 1800   7      29 162 4868   21     28 163 6717   6
      49 164 11596  27     35 165 5970   4      21 166 4461   21     31 167 1303   52
      16 168 209    33     15 169 487    47     24 170 252    33     6 171  8322   6
      15 172 536    45     23 173 549    37     21 174 789    43     26 175 2521   36
      15 176 579    37     24 177 6704   24     34 178 4434   36     21 179 9263   27
      23 180 14995  36     27 181 8754   47     7 182  18228  29     46 183 7556   21
      44 184 9967   7      21 185 11683  27     45 186 3438   15     33 187 1434   2
      49 188 475    2      19 189 7166   29     38 190 8996   27     10 191 682    24
      43 192 405    30     11 193 237    29     14 194 7556   44     27 195 171    32
      30 196 7176   21     16 197 1242   27     17 198 768    19     2 199  7418   21
      15 200 395    35     2 201  111    18     27 202 3281   45     41 203 21597  24
      7 204  21943  46     29 205 586    43     41 206 5938   15     6 207  5607   7
      25 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [16, 43] 2225 [15, 37] 643 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7,
42] 1875 [25, 24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21]
1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502
[15, 42] 1994 [15, 40] 601 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21,
12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13]
519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [24, 12] 631 [30, 19] 398 [25, 7] 178
[17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585
[33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [24,
25] 459 [49, 27] 468 [28, 29] 1860 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2]
590 [29, 47] 3001 [44, 7] 396 [46, 21] 2128 [23, 4] 2156 [27, 39] 1168 [19, 43]
1717 [23, 25] 1187 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684
[19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [43, 39] 359 [20, 16] 1397
[25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19,
16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24]
1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [21, 25] 458 [46, 24] 1362
[16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47,
15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50]
176 [43, 46] 307 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31] 2267 [48, 24] 905
[16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2]
1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [6, 1] 88 [16, 37] 1120 [48,
6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15]
```

819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [24, 4] 1474 [23, 45] 235 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [43, 38] 113 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [28, 6] 153 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [19, 19] 0 [24, 32] 1338 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [28, 7] 341 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [28, 24] 778 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [6, 7] 490 [19, 20] 1027 [37, 15] 639 [16, 25] 866 [33, 2] 574 [45, 43] 404 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [43, 26] 248 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [52, 21] 538 [7, 4] 969 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [43, 41] 731 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [27, 4] 1099 [21, 14] 1445 [16, 26] 1976 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [33, 24] 755 [49, 6] 393 [46, 7] 1814 [19, 39] 1598 [24, 48] 917 [7, 44] 394 [23, 37] 975 [25, 27] 269

[23, 46] 631 [21, 39] 1744 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [28, 15] 146 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [28, 21] 104 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [23, 32] 1764 [16, 39] 2152 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [19, 4] 817 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [45, 41] 635 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [27, 47] 1636 [29, 26] 369 [19, 41] 1260 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 20] 644 [7, 16] 851 [23, 28] 1543 [2, 21] 592 [47, 23] 2694 [24, 38] 979 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [29, 4] 2464 [27, 42] 1624 [12, 30] 44 [24, 40] 396 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174 [47, 29] 3002 [27, 6] 567 [6, 39] 1738 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [6, 46] 2121 [27, 41] 1115 [19, 31] 928 [16, 47] 663 [51, 16] 1158 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [17, 23] 1434 [23, 41] 864 [24, 41] 1444 [28, 23] 1552 [7, 40] 554 [20, 29] 1142 [28, 16] 512 [35, 15] 559 [21, 41] 1569 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [28, 27] 512 [24,

```
26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [52, 16]
388 [28, 19] 377 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282
[46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30]
495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093
[15, 17] 366 [16, 4] 142 [31, 6] 687 [32, 30] 690 [6, 6] 0 [37, 6] 756 [6, 18] 1023
[21, 16] 405 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [35,
23] 1131 [24, 39] 1323 [23, 6] 1624 ;
end;
```

```
outubro.dat
data;
param n := 52;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 5 4 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 52;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7,
16 43, 15 37, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18 21,
16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 17 16, 17 27, 44 6, 32 21, 21 12, 19
36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 50 16, 24 12, 30 19, 25
7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51,
40 6, 24 25, 49 27, 28 29, 27 12, 36 23, 49 21, 15 2, 29 47, 50 6, 44 7, 46 21, 27
39, 23 4, 19 43, 23 25, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44
16, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4
23, 31 24, 7 36, 21 23, 21 34, 19 8, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 7 12,
45 19, 47 15, 44 37, 33 29, 19 32, 49 24, 29 1, 6 50, 7 2, 36 32, 16 11, 23 31, 48
24, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 40 27, 6 1,
47 44, 16 37, 48 6, 43 23, 27 37, 27 33, 50 29, 23 49, 31 15, 48 23, 7 41, 24 30,
29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 52 19, 44 29, 47
24, 15 23, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9,
29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 24 4, 52 24, 29
27, 16 35, 24 10, 16 13, 40 23, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 22
15, 19 14, 16 8, 16 24, 50 21, 15 25, 23 36, 31 21, 15 44, 12 29, 15 11, 15 50, 21
27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20
15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24
29, 27 50, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 23
43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 30 27, 16
7, 24 1, 19 51, 6 32, 21 50, 51 7, 52 6, 52 15, 6 10, 19 5, 6 19, 21 9, 27 19, 6
35, 21 19, 15 4, 22 6, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 22 29, 15 43, 23
3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 28 6, 15 41, 7 45,
16 33, 19 44, 25 6, 30 29, 15 38, 7 23, 23 7, 6 25, 21 7, 19 19, 24 32, 6 42, 49 7,
12 24, 6 33, 44 21, 6 12, 23 5, 28 7, 29 21, 7 38, 27 46, 15 8, 47 27, 28 24, 23
14, 16 5, 27 21, 31 7, 22 24, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 6 7, 19 20,
```

37 15, 16 25, 40 16, 45 43, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 22 19, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 22 21, 15 10, 52 21, 19 9, 43 19, 7 4, 37 19, 43 41, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 50 7, 50 19, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 15 21, 29 3, 50 24, 25 20, 32 16, 21 4, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 29 36, 40 7, 24 33, 23 12, 21 14, 27 4, 16 26, 40 21, 29 8, 45 24, 6 21, 40 24, 33 24, 49 6, 19 39, 46 7, 24 48, 7 44, 23 37, 21 39, 23 46, 25 27, 19 7, 21 35, 32 7, 19 42, 25 23, 36 30, 23 35, 30 7, 21 36, 29 43, 28 15, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 22 27, 49 2, 7 5, 46 27, 24 16, 21 3, 22 23, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 21 32, 7 10, 17 24, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 22 16, 23 29, 50 23, 20 6, 6 37, 28 21, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 40 19, 40 29, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 46 23, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 52 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 31 4, 23 1, 18 16, 6 4, 29 14, 52 27, 15 18, 19 4, 35 2, 29 20, 19 26, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 34, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 52 29, 24 8, 35 6, 21 8, 6 13, 27 47, 29 26, 19 41, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 20, 7 16, 23 28, 2 21, 22 7, 24 38, 47 23, 31 23, 6 2, 23 30, 7 34, 35 49, 4 19, 7 6, 37 44, 19 48, 24 35, 6 17, 30 3, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 29 4, 16 14, 51 23, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 6 46, 27 41, 19 31, 16 47, 51 16, 27 32, 16 3, 29 49, 27 5, 30 12, 23 41, 24 41, 17 23, 28 23, 50 27, 20 29, 40 15, 28 16, 35 15, 21 41, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 28 27, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 52 16, 28 19, 21 21, 45 23, 6 38, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 16 4, 50 15, 31 6, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6;

param: K: R O D:= 1 4158 12 6 2 5014 43 29 3 17810 23
27 4 3996 21 49 5 5472 29 7 6 3348 23 34 7 5572 21
33 8 2495 37 7 9 10442 21 30 10 132 50 21 11 5204 27
26 12 7519 25 24 13 11932 23 24 14 9137 36 7 15 404 7
43 16 1176 45 27 17 990 6 36 18 109 49 19 19 5701 43
27 20 5203 7 15 21 8316 15 21 22 5188 31 27 23 776 25
20 24 12670 17 27 25 867 24 8 26 2083 6 20 27 6354 21
8 28 162 32 21 29 2778 29 26 30 171 19 36 31 329 32
6 32 23384 21 2 33 2492 21 29 34 7342 32 27 35 5954 29

```
39 36  614   30    19 37  3208  25    7 38   4334  27    49 39  1765  24
46 40  366   27    48 41  130   35    27 42  1166  29    24 43  330   2
21 44  16537 6     2 45   1694  24    25 46  870   49    6 47   1500  46
7 48   6990  27    3 49   9811  49    27 50  119   35    49 51  555   30
21 52  10911 7     6 53   346   37    44 54  13946 24    29 55  107   30
3 56   5915  24    14 57  2941  49    21 58  7185  16    21 59  11196 16
27 60  2059  47    21 61  937   4     27 62  2927  25    21 63  13367 27
30 64  581   36    3 65   5838  19    6 66   8940  33    27 67  899   23
37 68  6181  23    43 69  5423  21    37 70  12223 24    23 71  6710  25
27 72  20822 27    25 73  2399  24    37 74  6946  12    21 75  9779  27
15 76  1796  2     27 77  401   36    30 78  1046  21    36 79  10754 12
30 80  10979 29    43 81  7145  30    27 82  543   23    21 83  10536 27
44 84  4242  21    50 85  635   19    49 86  131   22    27 87  3361  6
19 88  8784  27    6 89   1183  49    2 90   748   25    44 91  3708  6
44 92  7849  7     8 93   10681 46    27 94  4577  27    19 95  12154 15
7 96   6661  29    45 97  3043  21    19 98  884   48    27 99  164   19
17 100 2530  31    24 101 1458  19    11 102 12344 27    18 103 794   21
23 104 464   30    36 105 107   37    27 106 18428 27    2 107  16340 21
25 108 2226  46    24 109 4385  21    47 110 13678 7     24 111 1206  43
45 112 1043  30    32 113 1203  19    30 114 1422  25    6 115  3492  29
9 116  1988  47    15 117 1068  19    21 118 20306 44    37 119 9836  7
23 120 488   20    24 121 14246 23    7 122  11301 6     25 123 860   15
36 124 9605  27    51 125 5072  29    42 126 6935  24    44 127 4036  6
33 128 3739  21    6 129  1675  7     2 130  332   36    32 131 295   17
30 132 7415  27    36 133 156   51    27 134 12389 23    29 135 2873  27
32 136 6232  6     37 137 13985 27    5 138  1207  30    12 139 220   28
21 140 5594  6     8 141  20041 47    27 142 4257  6     1 143  119   40
27 144 459   47    44 145 1216  36    13 146 5158  23    14 147 4400  27
11 148 23965 27    21 149 23577 27    43 150 2746  6     3 151  1197  31
7 152  295   27    37 153 2975  43    23 154 6387  27    33 155 21195 27
20 156 7340  6     30 157 15683 19    27 158 154   45    43 159 224   24
6 160  1083  2     6 161  7801  7     37 162 7386  24    20 163 2004  7
29 164 5029  21    28 165 8042  6     49 166 11307 27    35 167 9258  4
21 168 3796  21    31 169 301   52    16 170 258   33    15 171 1422  46
23 172 2984  47    24 173 245   33    6 174  8195  6     15 175 497   45
23 176 191   37    21 177 4916  36    15 178 1111  37    24 179 8366  24
34 180 3281  36    21 181 10976 27    23 182 14738 36    27 183 12421 47
7 184  15423 29    46 185 507   29    37 186 7329  21    44 187 15466 7
21 188 12151 27    45 189 5063  15    33 190 1395  2     49 191 630   2
19 192 8323  29    38 193 11495 27    10 194 17472 29    27 195 988   24
43 196 1734  31    4 197  774   30    11 198 6022  21    16 199 1228  27
17 200 1005  19    2 201  7849  21    15 202 273   35    2 203  134   18
27 204 17589 46    29 205 5610  43    41 206 4942  15    6 207  6214  7
25 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
```

[37, 7] 274 [16, 43] 2225 [15, 37] 643 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [7, 42] 1875 [25, 24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502 [15, 42] 1994 [17, 16] 781 [17, 27] 444 [44, 6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [50, 16] 414 [24, 12] 631 [30, 19] 398 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16, 9] 2465 [24, 31] 1585 [33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7] 395 [21, 51] 754 [40, 6] 789 [24, 25] 459 [49, 27] 468 [28, 29] 1860 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [50, 6] 174 [44, 7] 396 [46, 21] 2128 [27, 39] 1168 [23, 4] 2156 [19, 43] 1717 [23, 25] 1187 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 50] 176 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [40, 27] 226 [6, 1] 88 [47, 44] 1826 [16, 37] 1120 [48, 6] 106 [43, 23] 361 [27, 37] 536 [27, 33] 300 [50, 29] 2004 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [23, 45] 235 [24, 4] 1474 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [40, 23] 835 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [22, 15] 787 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [50, 21] 80 [15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [22, 6] 977 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [22, 29] 998 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [28, 6] 153 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [7, 23]

1248 [23, 7] 1249 [6, 25] 534 [21, 7] 441 [19, 19] 0 [24, 32] 1338 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [28, 7] 341 [29, 21] 1938 [7, 38] 1432 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [28, 24] 778 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [22, 24] 922 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [6, 7] 490 [19, 20] 1027 [37, 15] 639 [16, 25] 866 [40, 16] 1204 [45, 43] 404 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [22, 19] 840 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [22, 21] 983 [15, 10] 447 [52, 21] 538 [19, 9] 1957 [43, 19] 1717 [7, 4] 969 [37, 19] 897 [43, 41] 731 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [50, 7] 507 [50, 19] 392 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [15, 21] 204 [29, 3] 1693 [50, 24] 944 [25, 20] 530 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [29, 36] 2051 [40, 7] 554 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [27, 4] 1099 [16, 26] 1976 [40, 21] 796 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [40, 24] 395 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [23, 46] 631 [25, 27] 269 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [28, 15] 146 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [22, 27] 416 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [22, 23] 708 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [21, 32] 511 [7, 10] 501 [17, 24] 958 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [22, 16] 1391 [23, 29] 442 [50, 23] 1695 [20, 6] 1092 [6, 37] 760 [28, 21] 104 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [40, 19] 727 [40, 29] 1144 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [46, 23] 630 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [31, 4] 97 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [19, 4] 817 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 34] 371 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6,

```
11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15]
94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896
[15, 28] 146 [43, 27] 1247 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [21,
8] 313 [6, 13] 387 [27, 47] 1636 [29, 26] 369 [19, 41] 1260 [20, 7] 645 [21, 29]
1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [25, 19] 639
[24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 20] 644 [7,
16] 851 [23, 28] 1543 [2, 21] 592 [22, 7] 851 [24, 38] 979 [47, 23] 2694 [31, 23]
2245 [6, 2] 496 [23, 30] 1163 [7, 34] 878 [35, 49] 250 [4, 19] 813 [7, 6] 489 [37,
44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [30, 3] 231 [16, 21] 410 [20, 19]
1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24,
45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30]
318 [27, 42] 1624 [12, 30] 44 [29, 4] 2464 [16, 14] 1854 [51, 23] 926 [12, 23] 1157
[19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5]
1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [19, 17] 89
[24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979
[43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568 [7,
26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [19, 45] 1316 [27, 31]
1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336
[6, 46] 2121 [27, 41] 1115 [19, 31] 928 [16, 47] 663 [51, 16] 1158 [27, 32] 821
[16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [23, 41] 864 [24, 41] 1444 [17,
23] 1434 [28, 23] 1552 [50, 27] 640 [20, 29] 1142 [40, 15] 600 [28, 16] 512 [35,
15] 559 [21, 41] 1569 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21]
745 [43, 15] 1621 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356
[24, 6] 994 [7, 31] 1080 [7, 37] 275 [28, 27] 512 [24, 26] 918 [17, 6] 307 [6, 49]
391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [52, 16] 388 [28, 19] 377 [21, 21] 0
[45, 23] 236 [6, 38] 1739 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27,
1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820
[15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [16, 4] 142 [50,
15] 269 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461
[16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131
[23, 6] 1624 ;
end;
```

```
novembro.dat
data;
param n := 52;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 52;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7,
15 37, 16 43, 21 48, 16 15, 15 46, 25 24, 7 42, 36 7, 45 29, 15 19, 30 24, 18 21,
16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 40, 15 42, 17 16, 24 52, 17 27, 44 6, 32
```

21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 30 19, 24 12, 25 7, 36 2, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 33 19, 29 29, 27 29, 21 11, 33 7, 21 51, 40 6, 24 25, 49 27, 27 12, 36 23, 49 21, 15 2, 29 47, 44 7, 46 21, 23 4, 27 39, 23 25, 19 43, 21 20, 24 23, 23 2, 27 15, 23 39, 19 23, 23 50, 23 21, 44 16, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6, 27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 21 25, 46 24, 16 31, 27 34, 15 3, 6 23, 23 52, 7 12, 45 19, 47 15, 33 29, 19 32, 49 24, 29 1, 6 52, 6 50, 7 2, 36 32, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18 23, 7 48, 24 2, 12 7, 23 51, 6 8, 7 39, 40 27, 6 1, 47 44, 16 37, 48 6, 21 40, 43 23, 27 37, 27 33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29, 15 9, 47 16, 7 27, 52 19, 44 29, 47 24, 15 23, 10 15, 24 17, 21 10, 36 21, 2 15, 27 23, 24 24, 29 46, 33 16, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19, 44 15, 29 38, 29 10, 7 7, 24 13, 24 4, 23 45, 52 24, 29 27, 16 35, 24 10, 16 13, 40 23, 10 19, 19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 22 15, 19 14, 16 8, 16 24, 15 25, 23 36, 15 44, 31 21, 12 29, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 51 19, 48 29, 19 28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 29 23, 35 16, 29 34, 36 19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 15 27, 23 43, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 35 19, 48 19, 16 46, 29 52, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 51 7, 52 6, 52 15, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 15 4, 22 6, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 22 29, 15 43, 23 3, 37 27, 27 27, 27 2, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 6 25, 21 7, 19 19, 24 32, 10 21, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 29 21, 7 38, 15 52, 27 46, 15 8, 47 27, 10 27, 23 14, 16 5, 27 21, 31 7, 22 24, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 19 20, 6 7, 37 15, 16 25, 40 16, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 10 16, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 22 19, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 7 49, 27 45, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 22 21, 15 10, 52 21, 7 4, 19 9, 43 19, 37 19, 43 41, 27 52, 24 7, 46 29, 7 19, 18 7, 43 7, 46 45, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 46 42, 15 34, 16 20, 16 52, 15 21, 29 3, 25 20, 32 16, 21 4, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 40 7, 29 36, 24 33, 23 12, 27 4, 21 14, 16 26, 40 21, 29 8, 45 24, 6 21, 40 24, 33 24, 49 6, 19 39, 46 7, 24 48, 7 44, 23 37, 21 39, 23 46, 25 27, 19 7, 21 35, 32 7, 25 23, 19 42, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 22 27, 49 2, 7 5, 46 27, 24 16, 21 3, 22 23, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 30 32, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24, 21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 22 16, 23 29, 20 6, 6 37, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 40 19, 40 29, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 46 23, 19 52, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 47 7, 2 7, 52 7, 15 32, 15 39, 19 34, 16 45, 24 18, 10 7, 10 29, 15 16, 23 1, 18 16, 6 4, 29 14, 52 27, 15 18, 19 4, 35 2, 29 20, 19 26, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 27 7, 43 6, 24 15, 23 24, 16

```
44, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 20 21, 52 29, 24
8, 35 6, 6 13, 21 8, 27 47, 29 26, 19 41, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29,
10 24, 24 51, 25 19, 24 46, 27 48, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 22
7, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 19 48, 24 35, 6
17, 16 21, 20 19, 19 3, 27 30, 23 44, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41,
12 16, 7 32, 15 30, 29 4, 12 30, 27 42, 24 40, 7 52, 16 14, 51 23, 12 23, 19 49, 6
39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 21 52, 46 43,
19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 4 16,
21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6
31, 17 30, 6 46, 27 41, 19 31, 16 47, 51 16, 27 32, 16 3, 29 49, 27 5, 30 12, 17
23, 24 41, 23 41, 7 40, 40 15, 20 29, 35 15, 21 41, 16 23, 29 44, 27 43, 6 3, 35
21, 43 15, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 24 26, 17 6, 6 49,
29 50, 18 6, 21 31, 52 16, 21 21, 45 23, 10 23, 6 38, 46 44, 27 13, 6 34, 46 15, 6
5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 6 45, 27 10, 24 43, 15 17, 16
4, 31 6, 32 30, 6 6, 37 6, 6 18, 21 16, 18 27, 16 17, 10 6, 20 23, 7 25, 29 30, 24
39, 35 23, 23 6;
param: K:    R      O      D:= 1  4894   12     6 2    5107   43     29 3   17046  23
       27 4   3892   21     49 5   2231   29     7 6    3647   23     34 7   4617   21
       33 8   3026   37     7 9    9096   21     30 10  4022   27     26 11  15740  31
       21 12  9669   25     24 13  16286  23     24 14  9113   36     7 15   473    7
       43 16  1173   45     27 17  495    6      36 18  388    49     19 19  253    46
       42 20  4944   43     27 21  432    16     52 22  4941   7      15 23  6776   15
       21 24  4757   31     27 25  885    25     20 26  13199  17     27 27  1034   24
       8 28   2634   6      20 29  5134   21     8 30   2603   29     26 31  201    21
       4 32   174    19     36 33  21274  21     2 34   2498   21     29 35  7254   32
       27 36  4016   29     39 37  488    30     19 38  3955   25     7 39   410    36
       2 40   218    24     51 41  4130   27     49 42  842    24     46 43  365    27
       48 44  192    35     27 45  1407   29     24 46  425    2      21 47  9389   6
       21 48  17138  6      2 49   1963   24     25 50  1137   49     6 51   8267   27
       3 52   4472   46     7 53   17060  27     40 54  12553  49     27 55  742    30
       21 56  11405  7      6 57   416    37     44 58  13166  24     29 59  737    7
       44 60  141    27     12 61  5722   24     14 62  585    49     21 63  8355   16
       21 64  12555  16     27 65  2605   47     21 66  2872   25     21 67  16152  27
       30 68  835    36     3 69   5238   19     6 70   9604   33     27 71  1283   23
       37 72  22292  15     27 73  10523  23     43 74  4593   21     37 75  12868  24
       23 76  7450   25     27 77  20190  27     25 78  3007   24     37 79  11736  12
       21 80  736    19     7 81   9906   27     15 82  1755   32     7 83   138    2
       27 84  406    36     30 85  1515   21     36 86  3315   12     30 87  13116  29
       43 88  9027   30     27 89  2310   23     21 90  11896  27     44 91  2785   21
       50 92  332    19     49 93  201    22     27 94  1503   6      19 95  8853   27
       6 96   1001   49     2 97   1241   25     44 98  4040   6      44 99  9330   7
       8 100  5651   46     27 101 5317   27     19 102 4571   15     7 103  7288   29
       45 104 3309   21     19 105 3761   48     27 106 470    19     17 107 2222   31
       24 108 273    46     43 109 1019   19     11 110 11274  27     18 111 433    30
       36 112 209    37     27 113 12284  21     25 114 2335   46     24 115 5525   21
       47 116 12388  7      24 117 166    43     45 118 2509   48     7 119  1131   19
```

```
      30 120 724    30     32 121 332    21     43 122 660    25     6 123  4012   29
      9 124  1672   47     15 125 13060  7      23 126 23484  44     24 127 286    20
      24 128 15291  23     7 129  383    15     36 130 10086  6      25 131 10022  27
      51 132 4717   29     42 133 7546   24     44 134 2252   49     7 135  3822   6
      33 136 3276   21     6 137  534    7      2 138  124    36     32 139 275    17
      30 140 6607   27     36 141 228    51     27 142 18706  23     29 143 1909   27
      32 144 3906   6      37 145 16944  27     5 146  1682   30     12 147 4198   6
      8 148  728    12     7 149  20987  47     27 150 116    10     27 151 3815   6
      1 152  195    40     27 153 1052   47     44 154 1209   36     13 155 4443   23
      14 156 3892   27     11 157 20225  27     21 158 761    21     40 159 18848  27
      43 160 1567   31     7 161  2793   6      3 162  914    27     37 163 3081   43
      23 164 6896   27     33 165 15175  27     20 166 5660   6      30 167 11471  19
      27 168 1503   24     6 169  228    17     19 170 493    2      6 171  5670   7
      37 172 7619   24     20 173 2737   7      29 174 4711   21     28 175 6323   6
      49 176 12040  27     35 177 5148   4      21 178 5962   21     31 179 447    52
      16 180 253    33     15 181 291    46     23 182 4660   47     24 183 119    33
      6 184  8705   6      15 185 630    45     23 186 296    37     21 187 730    17
      7 188  4036   36     15 189 168    46     44 190 483    37     24 191 7428   24
      34 192 4943   36     21 193 12189  27     23 194 13904  36     27 195 13124  47
      7 196  20473  29     46 197 4399   21     44 198 17644  7      21 199 5432   15
      33 200 11983  27     45 201 1875   2      49 202 227    2      19 203 7452   29
      38 204 11845  27     10 205 155    23     45 206 18665  29     27 207 1164   24
      43 208 507    30     11 209 132    32     30 210 10358  21     16 211 1153   27
      17 212 662    19     2 213  8743   21     15 214 126    35     2 215  294    18
      27 216 19419  24     7 217  4072   15     6 218  13133  46     29 219 6526   43
      41 220 6645   7      25 221 167    46     45 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [15, 37] 643 [16, 43] 2225 [21, 48] 110 [16, 15] 611 [15, 46] 1933 [25,
24] 460 [7, 42] 1875 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24] 622 [18, 21]
1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371 [7, 50] 502
[15, 40] 601 [15, 42] 1994 [17, 16] 781 [24, 52] 1472 [17, 27] 444 [44, 6] 878 [32,
21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587 [30, 15]
317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [30, 19] 398 [24, 12] 631
[25, 7] 178 [36, 2] 507 [17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6] 1412 [16,
9] 2465 [24, 31] 1585 [33, 19] 326 [29, 29] 0 [27, 29] 1373 [21, 11] 443 [33, 7]
395 [21, 51] 754 [40, 6] 789 [24, 25] 459 [49, 27] 468 [27, 12] 114 [36, 23] 1762
[49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7] 396 [46, 21] 2128 [23, 4] 2156 [27,
39] 1168 [23, 25] 1187 [19, 43] 1717 [21, 20] 994 [24, 23] 735 [23, 2] 1334 [27,
15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694 [23, 21] 1630 [44, 16] 1240 [20,
16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15, 14] 1251 [15, 7] 373 [6, 26]
1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47] 1137 [49, 29] 1675 [4, 23] 2154
[31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34] 1288 [19, 8] 621 [21, 25] 458 [46,
24] 1362 [16, 31] 253 [27, 34] 723 [15, 3] 253 [6, 23] 1624 [23, 52] 2009 [7, 12]
532 [45, 19] 1318 [47, 15] 1267 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1]
2029 [6, 52] 521 [6, 50] 176 [7, 2] 942 [36, 32] 182 [16, 11] 650 [23, 31] 2267
```

[48, 24] 905 [16, 10] 1050 [27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12, 7] 532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [40, 27] 226 [6, 1] 88 [47, 44] 1826 [16, 37] 1120 [48, 6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15] 819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959 [15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7, 27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [10, 15] 438 [24, 17] 958 [21, 10] 642 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0 [29, 46] 322 [33, 16] 726 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35] 1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352 [7, 7] 0 [24, 13] 1396 [24, 4] 1474 [23, 45] 235 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24, 10] 577 [16, 13] 458 [40, 23] 835 [10, 19] 556 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16] 498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [22, 15] 787 [19, 14] 1375 [16, 8] 721 [16, 24] 1286 [15, 25] 345 [23, 36] 1760 [15, 44] 703 [31, 21] 617 [12, 29] 1466 [15, 11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51, 19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20] 1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [29, 23] 440 [35, 16] 1125 [29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [15, 27] 380 [23, 43] 362 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [35, 19] 433 [48, 19] 330 [16, 46] 2536 [29, 52] 2299 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [15, 4] 730 [22, 6] 977 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [22, 29] 998 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 27] 0 [27, 2] 523 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [6, 25] 534 [21, 7] 441 [19, 19] 0 [24, 32] 1338 [10, 21] 633 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [29, 21] 1938 [7, 38] 1432 [15, 52] 728 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [10, 27] 66 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [22, 24] 922 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [40, 16] 1204 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [10, 16] 1042 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [22, 19] 840 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [7, 49] 858 [27, 45] 851 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [22, 21] 983 [15, 10] 447 [52, 21] 538 [7, 4] 969 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [43, 41] 731 [27, 52] 1097 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [46, 45] 715 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7,

11] 839 [24, 42] 1492 [46, 42] 464 [15, 34] 1093 [16, 20] 1403 [16, 52] 387 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 4] 523 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [40, 7] 554 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [27, 4] 1099 [21, 14] 1445 [16, 26] 1976 [40, 21] 796 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [40, 24] 395 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [23, 46] 631 [25, 27] 269 [19, 7] 749 [21, 35] 743 [32, 7] 941 [25, 23] 1187 [19, 42] 2090 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [22, 27] 416 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [22, 23] 708 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [30, 32] 692 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [22, 16] 1391 [23, 29] 442 [20, 6] 1092 [6, 37] 760 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [40, 19] 727 [40, 29] 1144 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [46, 23] 630 [19, 52] 469 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [10, 7] 501 [10, 29] 1352 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [6, 4] 600 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [19, 4] 817 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [27, 7] 445 [43, 6] 1811 [24, 15] 831 [23, 24] 734 [16, 44] 1170 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [6, 13] 387 [21, 8] 313 [27, 47] 1636 [29, 26] 369 [19, 41] 1260 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [10, 24] 578 [24, 51] 232 [25, 19] 639 [24, 46] 1362 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [22, 7] 851 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [27, 30] 106 [23, 44] 875 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [29, 4] 2464 [12, 30] 44 [27, 42] 1624 [24, 40] 396 [7, 52] 967 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [21, 52] 535 [46, 43] 307 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [4, 16] 132 [21, 26] 1568

```
[7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [19, 45] 1316 [27,
31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30]
336 [6, 46] 2121 [27, 41] 1115 [19, 31] 928 [16, 47] 663 [51, 16] 1158 [27, 32] 821
[16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [17, 23] 1434 [24, 41] 1444 [23,
41] 864 [7, 40] 554 [40, 15] 600 [20, 29] 1142 [35, 15] 559 [21, 41] 1569 [16, 23]
2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [45, 16]
1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080
[7, 37] 275 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21,
31] 633 [52, 16] 388 [21, 21] 0 [45, 23] 236 [10, 23] 1043 [6, 38] 1739 [46, 44]
1502 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346
[36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [6, 45]
1410 [27, 10] 66 [24, 43] 1093 [15, 17] 366 [16, 4] 142 [31, 6] 687 [32, 30] 690
[6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [18, 27] 461 [16, 17] 779 [10, 6]
627 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131 [23, 6] 1624
;
end;
```

```
dezembro.dat
data;
param n := 53;
set T := truck carreta;
param U := truck 12000, carreta 24000;
param c := truck 1.0, carreta 1.4;
set V := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 17 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49 48 50 51 52 53;
set A := 43 24, 51 15, 6 27, 43 29, 23 27, 21 49, 29 7, 21 46, 7 46, 29 18, 37 7, 7
53, 16 43, 15 37, 21 48, 16 15, 15 46, 7 42, 25 24, 36 7, 45 29, 15 19, 30 24, 18
21, 16 18, 6 36, 23 20, 31 27, 7 15, 7 50, 15 42, 15 40, 17 16, 24 52, 17 27, 44 6,
32 21, 21 12, 19 36, 32 6, 23 47, 21 2, 30 15, 15 13, 20 27, 32 27, 29 39, 24 12,
30 19, 21 53, 25 7, 17 15, 37 16, 15 29, 45 6, 16 9, 24 31, 5 16, 33 19, 29 29, 27
29, 21 11, 33 7, 21 51, 40 6, 24 25, 49 27, 28 29, 27 12, 36 23, 49 21, 15 2, 29
47, 44 7, 46 21, 27 39, 19 43, 23 25, 21 20, 24 23, 23 2, 6 53, 27 15, 23 39, 19
23, 23 50, 23 21, 44 16, 20 16, 25 44, 7 8, 24 21, 15 14, 15 7, 6 26, 19 16, 30 6,
27 16, 6 47, 49 29, 4 23, 31 24, 7 36, 21 23, 21 34, 19 8, 15 22, 21 25, 46 24, 16
31, 27 34, 15 3, 6 23, 23 52, 7 12, 45 19, 47 15, 15 53, 44 37, 33 29, 19 32, 49
24, 29 1, 6 52, 6 50, 7 2, 16 11, 23 31, 48 24, 16 10, 27 36, 7 3, 35 7, 18 23, 7
48, 24 2, 12 7, 23 51, 6 8, 7 39, 40 27, 6 1, 16 37, 48 6, 21 40, 43 23, 27 37, 27
33, 23 49, 31 15, 48 23, 7 41, 24 30, 29 6, 43 16, 19 18, 15 20, 21 1, 24 20, 7 29,
15 9, 47 16, 7 27, 52 19, 44 29, 47 24, 15 23, 24 17, 45 26, 21 10, 36 53, 36 21, 2
15, 27 23, 24 24, 29 46, 33 16, 5 40, 6 51, 7 21, 27 9, 29 35, 16 34, 2 49, 2 19,
44 15, 29 38, 29 10, 7 7, 24 13, 23 45, 52 24, 29 27, 16 35, 24 10, 16 13, 40 23,
19 2, 21 15, 29 51, 36 16, 37 23, 15 6, 15 47, 19 14, 16 8, 16 24, 15 25, 23 36, 31
21, 15 44, 12 29, 5 21, 15 11, 15 50, 21 27, 45 15, 29 28, 16 38, 51 19, 48 29, 19
28, 21 42, 21 13, 48 21, 6 20, 16 42, 20 15, 16 19, 5 23, 29 23, 35 16, 29 34, 36
```

19, 27 49, 29 24, 7 51, 23 38, 27 3, 30 21, 24 29, 27 50, 19 22, 24 14, 16 27, 47 21, 4 27, 25 21, 6 28, 16 48, 29 17, 33 27, 16 28, 23 43, 15 27, 49 16, 12 21, 24 37, 15 45, 15 49, 2 27, 5 19, 35 19, 48 19, 29 52, 16 46, 30 27, 16 7, 24 1, 19 51, 6 32, 21 50, 51 7, 52 6, 52 15, 6 10, 19 5, 6 19, 21 9, 27 19, 6 35, 21 19, 29 53, 7 9, 16 1, 6 41, 19 11, 21 17, 6 14, 6 43, 29 22, 15 43, 23 3, 37 27, 27 2, 27 27, 29 11, 2 29, 30 23, 2 16, 37 29, 29 25, 28 6, 15 41, 7 45, 16 33, 19 44, 25 6, 30 29, 15 38, 6 40, 7 23, 23 7, 21 7, 6 25, 24 32, 19 19, 6 42, 49 7, 12 24, 6 33, 44 21, 6 12, 23 5, 28 7, 29 21, 7 38, 15 52, 27 46, 15 8, 47 27, 16 22, 28 24, 23 14, 16 5, 27 21, 31 7, 25 29, 46 6, 31 16, 29 16, 24 47, 15 31, 19 20, 6 7, 37 15, 16 25, 40 16, 45 43, 46 16, 29 32, 33 23, 24 28, 17 19, 23 10, 21 24, 6 29, 44 23, 30 16, 12 19, 23 15, 33 6, 43 26, 17 7, 17 21, 51 6, 15 1, 2 24, 12 27, 29 33, 32 23, 12 15, 29 37, 21 44, 49 23, 23 19, 27 45, 7 49, 25 15, 35 29, 23 8, 24 19, 15 15, 30 11, 44 27, 52 23, 15 51, 27 17, 4 29, 16 50, 47 6, 15 10, 52 21, 19 9, 43 19, 37 19, 43 41, 27 52, 24 7, 46 29, 7 19, 18 7, 43 7, 12 6, 32 24, 27 38, 16 29, 15 5, 21 30, 27 26, 32 15, 45 21, 16 49, 49 15, 29 19, 19 13, 32 19, 7 43, 49 19, 7 11, 24 42, 15 34, 16 20, 16 52, 15 21, 29 3, 25 20, 32 16, 21 38, 36 24, 19 46, 27 28, 21 5, 6 9, 40 7, 29 36, 24 33, 23 12, 21 14, 24 53, 16 26, 40 21, 29 8, 45 24, 6 21, 40 24, 5 7, 33 24, 49 6, 19 39, 46 7, 24 22, 5 6, 24 48, 7 44, 23 37, 21 39, 25 27, 23 46, 19 7, 21 35, 32 7, 19 42, 25 23, 19 40, 36 30, 23 35, 30 7, 21 36, 29 43, 28 15, 51 29, 27 44, 7 28, 23 48, 16 2, 19 47, 49 2, 7 5, 46 27, 24 16, 21 3, 19 25, 16 32, 15 48, 48 27, 23 17, 30 36, 51 24, 27 14, 46 20, 7 24, 7 17, 48 7, 16 6, 19 30, 19 10, 21 43, 29 9, 29 2, 7 1, 19 38, 29 15, 16 12, 17 24, 21 32, 7 10, 25 16, 20 24, 15 36, 23 18, 15 26, 27 51, 7 35, 24 44, 21 6, 17 29, 19 29, 51 27, 31 29, 29 48, 47 19, 23 29, 20 6, 6 37, 28 21, 23 26, 36 13, 27 11, 2 23, 7 18, 21 18, 19 24, 24 3, 19 35, 27 20, 7 33, 19 53, 16 39, 23 32, 23 16, 16 30, 43 21, 2 6, 19 15, 18 15, 40 19, 40 29, 21 28, 4 21, 27 35, 24 36, 33 15, 16 41, 23 22, 46 23, 19 52, 29 13, 6 15, 32 29, 37 21, 36 15, 4 7, 16 40, 15 12, 6 16, 37 24, 19 12, 24 34, 23 9, 19 50, 27 24, 27 53, 47 7, 2 7, 52 7, 15 32, 15 39, 19 34, 16 45, 24 18, 15 16, 23 1, 18 16, 29 14, 52 27, 15 18, 35 2, 29 20, 19 26, 7 13, 16 16, 45 7, 19 1, 23 42, 16 36, 19 33, 23 40, 23 34, 29 40, 24 50, 23 11, 21 33, 6 11, 43 6, 27 7, 24 15, 16 44, 23 24, 48 15, 45 27, 18 29, 23 13, 46 19, 51 21, 19 37, 15 28, 43 27, 5 29, 20 21, 52 29, 24 8, 35 6, 21 8, 6 13, 27 47, 19 41, 29 26, 20 7, 21 29, 31 19, 16 51, 27 8, 36 29, 24 51, 16 53, 25 19, 24 46, 30 25, 27 48, 7 47, 35 27, 18 24, 7 16, 7 20, 23 28, 2 21, 24 38, 47 23, 31 23, 6 2, 23 30, 27 40, 7 34, 4 19, 7 6, 37 44, 19 48, 30 2, 24 35, 6 17, 16 21, 20 19, 19 3, 23 44, 27 30, 36 3, 19 6, 6 24, 24 45, 21 37, 27 25, 29 41, 12 16, 7 32, 15 30, 27 42, 12 30, 24 40, 7 52, 16 14, 51 23, 12 23, 19 49, 6 39, 47 29, 27 6, 6 44, 36 6, 29 5, 29 45, 29 12, 6 48, 18 19, 7 14, 21 52, 19 17, 24 11, 27 18, 44 19, 35 24, 21 47, 24 49, 43 45, 21 45, 23 23, 24 27, 5 24, 4 16, 21 26, 7 26, 4 24, 4 15, 19 21, 15 35, 7 22, 19 45, 27 31, 44 24, 33 21, 29 31, 29 42, 6 31, 17 30, 27 41, 6 46, 19 31, 16 47, 51 16, 6 22, 27 32, 16 3, 29 49, 27 5, 30 12, 17 23, 24 41, 23 41, 28 23, 7 40, 5 27, 40 15, 20 29, 28 16, 35 15, 21 41, 16 23, 29 44, 27 43, 6 3, 35 21, 43 15, 27 22, 45 16, 48 16, 6 30, 19 27, 23 33, 24 6, 7 31, 7 37, 28 27, 24 26, 17 6, 6 49, 29 50, 18 6, 21 31, 52 16, 28 19, 21 21, 45 23, 6 38, 5 15, 27 13, 6 34, 46 15, 6 5, 27 1, 24 5, 36 27, 24 9, 7 30, 4 6, 15 24, 15 33, 27 10, 6 45, 24 43, 15 17, 31 6, 6 6, 37 6, 6 18, 21 16, 23 53, 18 27, 16 17, 20 23, 7 25, 29 30, 24 39, 35 23, 23 6, 21 22;

```
param: K:    R      O      D:= 1  3700   12     6 2    6502   43     29 3   13416  23
       27 4   2595   21     49 5   6462   29     7 6    3356   23     34 7   295    16
       24 8   5689   21     33 9   2413   37     7 10   7536   21     30 11  2995   27
       26 12  13984  31     21 13  7578   25     24 14  698    21     27 15  9156   23
       24 16  8050   36     7 17   780    7      43 18  893    45     27 19  933    30
       24 20  412    6      36 21  241    49     19 22  4968   43     27 23  189    16
       52 24  4989   7      15 25  9496   15     21 26  2782   31     27 27  849    25
       20 28  7356   17     27 29  1422   24     8 30   3124   6      20 31  4222   21
       8 32   3411   29     26 33  129    19     36 34  17197  21     2 35   2232   21
       29 36  5812   32     27 37  4709   29     39 38  242    30     19 39  4354   27
       49 40  403    24     46 41  458    30     25 42  299    27     48 43  191    35
       27 44  637    2      21 45  12569  6      21 46  17954  6      2 47   1930   24
       25 48  1648   46     7 49   17210  27     40 50  6331   27     3 51   8276   49
       27 52  687    30     21 53  10964  7      6 54   1126   37     44 55  9754   24
       29 56  352    30     2 57   6261   24     14 58  194    49     21 59  4537   16
       21 60  11877  16     27 61  1592   47     21 62  2330   25     21 63  16209  27
       30 64  562    36     3 65   3499   19     6 66   6248   33     27 67  912    23
       37 68  1334   15     27 69  7805   23     43 70  3580   21     37 71  13461  24
       23 72  6842   25     27 73  20199  27     25 74  3063   24     37 75  3981   12
       21 76  752    19     7 77   8642   27     15 78  282    32     7 79   1480   2
       27 80  242    36     30 81  1194   21     36 82  1484   12     30 83  13995  29
       43 84  10320  30     27 85  3709   23     21 86  9942   27     44 87  3564   21
       50 88  335    19     49 89  3267   6      19 90  5641   27     6 91   2622   49
       2 92   1025   25     44 93  3714   6      44 94  7909   7      8 95   4981   46
       27 96  336    36     6 97   4223   27     19 98  3754   15     7 99   9126   29
       45 100 2482   21     19 101 3890   48     27 102 125    19     17 103 3516   31
       24 104 1074   19     11 105 10836  27     18 106 342    30     36 107 12416  21
       25 108 2872   46     24 109 5982   21     47 110 698    46     20 111 9416   7
       24 112 526    43     45 113 10225  48     7 114  980    19     30 115 1410   21
       43 116 22034  24     27 117 1478   25     6 118  3194   29     9 119  1389   47
       15 120 441    19     21 121 16495  44     37 122 12025  7      23 123 316    20
       24 124 6939   23     7 125  445    15     36 126 13565  6      25 127 13482  27
       51 128 4737   29     42 129 5032   24     44 130 285    49     7 131  4292   6
       33 132 4387   21     6 133  207    17     30 134 4536   27     36 135 179    51
       27 136 17668  23     29 137 537    27     46 138 2018   27     32 139 4352   6
       37 140 15515  27     5 141  845    30     12 142 130    28     21 143 5096   6
       8 144  15254  47     27 145 4061   6      1 146  117    40     27 147 993    36
       13 148 4948   23     14 149 3729   27     11 150 15361  27     21 151 21430  27
       43 152 2466   6      3 153  3232   31     7 154  1990   43     23 155 23247  27
       22 156 7164   27     33 157 13044  27     20 158 5746   6      30 159 12557  19
       27 160 107    45     43 161 2027   24     6 162  933    17     19 163 555    2
       6 164  6799   7      37 165 9583   24     20 166 878    7      29 167 3914   21
       28 168 5949   6      49 169 10500  27     35 170 4584   4      21 171 6056   21
       31 172 391    52     16 173 161    33     15 174 5165   47     24 175 244    33
       6 176  8866   6      15 177 852    45     23 178 382    37     21 179 250    43
       26 180 294    17     7 181  5010   36     15 182 176    45     26 183 546    37
```

```
      24 184 7144   24     34 185 823    36     53 186 3927   36     21 187 17130  12
      27 188 8558   27     23 189 16475  36     27 190 8225   47     7 191  17361  29
      46 192 228    29     37 193 7495   21     44 194 248    5      40 195 6877   7
      21 196 4936   15     33 197 10227  27     45 198 1087   2      49 199 6523   29
      38 200 8536   27     10 201 125    25     15 202 318    23     45 203 18845  29
      27 204 487    24     43 205 751    30     11 206 2747   44     27 207 10123  21
      16 208 1094   27     17 209 401    19     2 210  7873   21     15 211 257    35
      2 212  262    18     27 213 11228  24     7 214  4130   15     6 215  11892  46
      29 216 6538   43     41 217 4924   7      25 ;
param distancia default 0 := [43, 24] 1092 [51, 15] 663 [6, 27] 569 [43, 29] 118
[23, 27] 1061 [21, 49] 483 [29, 7] 1625 [21, 46] 2127 [7, 46] 1814 [29, 18] 909
[37, 7] 274 [7, 53] 990 [16, 43] 2225 [15, 37] 643 [21, 48] 110 [16, 15] 611 [15,
46] 1933 [7, 42] 1875 [25, 24] 460 [36, 7] 762 [45, 29] 526 [15, 19] 320 [30, 24]
622 [18, 21] 1030 [16, 18] 1437 [6, 36] 267 [23, 20] 704 [31, 27] 1190 [7, 15] 371
[7, 50] 502 [15, 42] 1994 [15, 40] 601 [17, 16] 781 [24, 52] 1472 [17, 27] 444 [44,
6] 878 [32, 21] 514 [21, 12] 549 [19, 36] 218 [32, 6] 444 [23, 47] 2693 [21, 2] 587
[30, 15] 317 [15, 13] 519 [20, 27] 529 [32, 27] 821 [29, 39] 480 [24, 12] 631 [30,
19] 398 [21, 53] 558 [25, 7] 178 [17, 15] 368 [37, 16] 1119 [15, 29] 1744 [45, 6]
1412 [16, 9] 2465 [24, 31] 1585 [5, 16] 1305 [33, 19] 326 [29, 29] 0 [27, 29] 1373
[21, 11] 443 [33, 7] 395 [21, 51] 754 [40, 6] 789 [24, 25] 459 [49, 27] 468 [28,
29] 1860 [27, 12] 114 [36, 23] 1762 [49, 21] 487 [15, 2] 590 [29, 47] 3001 [44, 7]
396 [46, 21] 2128 [27, 39] 1168 [19, 43] 1717 [23, 25] 1187 [21, 20] 994 [24, 23]
735 [23, 2] 1334 [6, 53] 544 [27, 15] 377 [23, 39] 684 [19, 23] 1549 [23, 50] 1694
[23, 21] 1630 [44, 16] 1240 [20, 16] 1397 [25, 44] 343 [7, 8] 129 [24, 21] 949 [15,
14] 1251 [15, 7] 373 [6, 26] 1562 [19, 16] 699 [30, 6] 507 [27, 16] 981 [6, 47]
1137 [49, 29] 1675 [4, 23] 2154 [31, 24] 1494 [7, 36] 762 [21, 23] 1630 [21, 34]
1288 [19, 8] 621 [15, 22] 797 [21, 25] 458 [46, 24] 1362 [16, 31] 253 [27, 34] 723
[15, 3] 253 [6, 23] 1624 [23, 52] 2009 [7, 12] 532 [45, 19] 1318 [47, 15] 1267 [15,
53] 751 [44, 37] 122 [33, 29] 1667 [19, 32] 314 [49, 24] 982 [29, 1] 2029 [6, 52]
521 [6, 50] 176 [7, 2] 942 [16, 11] 650 [23, 31] 2267 [48, 24] 905 [16, 10] 1050
[27, 36] 763 [7, 3] 604 [35, 7] 812 [18, 23] 602 [7, 48] 457 [24, 2] 1040 [12, 7]
532 [23, 51] 926 [6, 8] 362 [7, 39] 1603 [40, 27] 226 [6, 1] 88 [16, 37] 1120 [48,
6] 106 [21, 40] 796 [43, 23] 361 [27, 37] 536 [27, 33] 300 [23, 49] 1365 [31, 15]
819 [48, 23] 1521 [7, 41] 1550 [24, 30] 623 [29, 6] 1932 [43, 16] 2225 [19, 18] 959
[15, 20] 903 [21, 1] 102 [24, 20] 130 [7, 29] 1625 [15, 9] 1862 [47, 16] 662 [7,
27] 440 [52, 19] 468 [44, 29] 1313 [47, 24] 1942 [15, 23] 1435 [24, 17] 958 [45,
26] 156 [21, 10] 642 [36, 53] 315 [36, 21] 336 [2, 15] 593 [27, 23] 1065 [24, 24] 0
[29, 46] 322 [33, 16] 726 [5, 40] 101 [6, 51] 803 [7, 21] 444 [27, 9] 1492 [29, 35]
1421 [16, 34] 1696 [2, 49] 107 [2, 19] 280 [44, 15] 715 [29, 38] 193 [29, 10] 1352
[7, 7] 0 [24, 13] 1396 [23, 45] 235 [52, 24] 1402 [29, 27] 1369 [16, 35] 1124 [24,
10] 577 [16, 13] 458 [40, 23] 835 [19, 2] 278 [21, 15] 203 [29, 51] 1234 [36, 16]
498 [37, 23] 976 [15, 6] 198 [15, 47] 1267 [19, 14] 1375 [16, 8] 721 [16, 24] 1286
[15, 25] 345 [23, 36] 1760 [31, 21] 617 [15, 44] 703 [12, 29] 1466 [5, 21] 896 [15,
11] 483 [15, 50] 268 [21, 27] 574 [45, 15] 1222 [29, 28] 1851 [16, 38] 2153 [51,
19] 789 [48, 29] 1830 [19, 28] 378 [21, 42] 2189 [21, 13] 459 [48, 21] 112 [6, 20]
1043 [16, 42] 2597 [20, 15] 902 [16, 19] 697 [5, 23] 737 [29, 23] 440 [35, 16] 1125
```

[29, 34] 809 [36, 19] 221 [27, 49] 463 [29, 24] 1172 [7, 51] 419 [23, 38] 248 [27, 3] 337 [30, 21] 513 [24, 29] 1173 [27, 50] 637 [19, 22] 840 [24, 14] 522 [16, 27] 983 [47, 21] 1066 [4, 27] 1099 [25, 21] 461 [6, 28] 153 [16, 48] 518 [29, 17] 1724 [33, 27] 303 [16, 28] 509 [23, 43] 362 [15, 27] 380 [49, 16] 867 [12, 21] 549 [24, 37] 238 [15, 45] 1221 [15, 49] 485 [2, 27] 528 [5, 19] 827 [35, 19] 433 [48, 19] 330 [29, 52] 2299 [16, 46] 2536 [30, 27] 108 [16, 7] 850 [24, 1] 1039 [19, 51] 787 [6, 32] 448 [21, 50] 81 [51, 7] 419 [52, 6] 518 [52, 15] 727 [6, 10] 636 [19, 5] 825 [6, 19] 225 [21, 9] 2057 [27, 19] 504 [6, 35] 651 [21, 19] 317 [29, 53] 2186 [7, 9] 1743 [16, 1] 373 [6, 41] 1477 [19, 11] 205 [21, 17] 398 [6, 14] 1440 [6, 43] 1811 [29, 22] 999 [15, 43] 1622 [23, 3] 1384 [37, 27] 532 [27, 2] 523 [27, 27] 0 [29, 11] 1917 [2, 29] 1628 [30, 23] 1163 [2, 16] 972 [37, 29] 1414 [29, 25] 1495 [28, 6] 153 [15, 41] 1496 [7, 45] 1102 [16, 33] 725 [19, 44] 929 [25, 6] 506 [30, 29] 1471 [15, 38] 1551 [6, 40] 790 [7, 23] 1248 [23, 7] 1249 [21, 7] 441 [6, 25] 534 [24, 32] 1338 [19, 19] 0 [6, 42] 2183 [49, 7] 860 [12, 24] 630 [6, 33] 311 [44, 21] 833 [6, 12] 544 [23, 5] 736 [28, 7] 341 [29, 21] 1938 [7, 38] 1432 [15, 52] 728 [27, 46] 1563 [15, 8] 244 [47, 27] 1639 [16, 22] 1400 [28, 24] 778 [23, 14] 213 [16, 5] 1304 [27, 21] 573 [31, 7] 1057 [25, 29] 1496 [46, 6] 2122 [31, 16] 224 [29, 16] 2346 [24, 47] 2011 [15, 31] 841 [19, 20] 1027 [6, 7] 490 [37, 15] 639 [16, 25] 866 [40, 16] 1204 [45, 43] 404 [46, 16] 2536 [29, 32] 2054 [33, 23] 1359 [24, 28] 845 [17, 19] 89 [23, 10] 1044 [21, 24] 878 [6, 29] 1932 [44, 23] 875 [30, 16] 921 [12, 19] 436 [23, 15] 1434 [33, 6] 311 [43, 26] 248 [17, 7] 710 [17, 21] 401 [51, 6] 852 [15, 1] 295 [2, 24] 1043 [12, 27] 116 [29, 33] 1665 [32, 23] 1739 [12, 15] 354 [29, 37] 1413 [21, 44] 762 [49, 23] 1385 [23, 19] 1548 [27, 45] 851 [7, 49] 858 [25, 15] 347 [35, 29] 1422 [23, 8] 1299 [24, 19] 1021 [15, 15] 0 [30, 11] 555 [44, 27] 492 [52, 23] 1985 [15, 51] 663 [27, 17] 441 [4, 29] 2462 [16, 50] 411 [47, 6] 1136 [15, 10] 447 [52, 21] 538 [19, 9] 1957 [43, 19] 1717 [37, 19] 897 [43, 41] 731 [27, 52] 1097 [24, 7] 512 [46, 29] 324 [7, 19] 713 [18, 7] 716 [43, 7] 1503 [12, 6] 543 [32, 24] 1336 [27, 38] 1180 [16, 29] 2346 [15, 5] 701 [21, 30] 512 [27, 26] 1003 [32, 15] 585 [45, 21] 1418 [16, 49] 864 [49, 15] 487 [29, 19] 1838 [19, 13] 269 [32, 19] 313 [7, 43] 1503 [49, 19] 175 [7, 11] 839 [24, 42] 1492 [15, 34] 1093 [16, 20] 1403 [16, 52] 387 [15, 21] 204 [29, 3] 1693 [25, 20] 530 [32, 16] 606 [21, 38] 1745 [36, 24] 1199 [19, 46] 2028 [27, 28] 510 [21, 5] 896 [6, 9] 2051 [40, 7] 554 [29, 36] 2051 [24, 33] 754 [23, 12] 1157 [21, 14] 1445 [24, 53] 1495 [16, 26] 1976 [40, 21] 796 [29, 8] 1643 [45, 24] 812 [6, 21] 102 [40, 24] 395 [5, 7] 583 [33, 24] 755 [49, 6] 393 [19, 39] 1598 [46, 7] 1814 [24, 22] 922 [5, 6] 890 [24, 48] 917 [7, 44] 394 [23, 37] 975 [21, 39] 1744 [25, 27] 269 [23, 46] 631 [19, 7] 749 [21, 35] 743 [32, 7] 941 [19, 42] 2090 [25, 23] 1187 [19, 40] 726 [36, 30] 610 [23, 35] 1131 [30, 7] 496 [21, 36] 330 [29, 43] 118 [28, 15] 146 [51, 29] 1235 [27, 44] 496 [7, 28] 339 [23, 48] 1520 [16, 2] 968 [19, 47] 1354 [49, 2] 105 [7, 5] 582 [46, 27] 1559 [24, 16] 1356 [21, 3] 397 [19, 25] 637 [16, 32] 608 [15, 48] 94 [48, 27] 466 [23, 17] 1434 [30, 36] 611 [51, 24] 231 [27, 14] 881 [46, 20] 1331 [7, 24] 511 [7, 17] 708 [48, 7] 459 [16, 6] 480 [19, 30] 396 [19, 10] 547 [21, 43] 1817 [29, 9] 384 [29, 2] 1624 [7, 1] 533 [19, 38] 1646 [29, 15] 1742 [16, 12] 958 [17, 24] 958 [21, 32] 511 [7, 10] 501 [25, 16] 869 [20, 24] 130 [15, 36] 398 [23, 18] 629 [15, 26] 1373 [27, 51] 293 [7, 35] 812 [24, 44] 138 [21, 6] 99 [17, 29] 1725 [19, 29] 1839 [51, 27] 289 [31, 29] 2554 [29, 48] 1828 [47, 19] 1353 [23, 29] 442

[20, 6] 1092 [6, 37] 760 [28, 21] 104 [23, 26] 186 [36, 13] 106 [27, 11] 684 [2, 23] 1337 [7, 18] 716 [21, 18] 1029 [19, 24] 1019 [24, 3] 854 [19, 35] 434 [27, 20] 533 [7, 33] 394 [19, 53] 447 [16, 39] 2152 [23, 32] 1764 [23, 16] 2038 [16, 30] 921 [43, 21] 1817 [2, 6] 498 [19, 15] 321 [18, 15] 834 [40, 19] 727 [40, 29] 1144 [21, 28] 101 [4, 21] 526 [27, 35] 393 [24, 36] 1267 [33, 15] 122 [16, 41] 1949 [23, 22] 708 [46, 23] 630 [19, 52] 469 [29, 13] 2099 [6, 15] 197 [32, 29] 2029 [37, 21] 712 [36, 15] 407 [4, 7] 966 [16, 40] 1204 [15, 12] 355 [6, 16] 482 [37, 24] 238 [19, 12] 435 [24, 34] 364 [23, 9] 628 [19, 50] 392 [27, 24] 524 [27, 53] 954 [47, 7] 1506 [2, 7] 944 [52, 7] 965 [15, 32] 579 [15, 39] 1549 [19, 34] 1218 [16, 45] 1824 [24, 18] 436 [15, 16] 612 [23, 1] 1720 [18, 16] 1438 [29, 14] 651 [52, 27] 1098 [15, 18] 835 [35, 2] 219 [29, 20] 1142 [19, 26] 1469 [7, 13] 891 [16, 16] 0 [45, 7] 1104 [19, 1] 304 [23, 42] 760 [16, 36] 495 [19, 33] 328 [23, 40] 835 [23, 34] 371 [29, 40] 1143 [24, 50] 1007 [23, 11] 1626 [21, 33] 317 [6, 11] 351 [43, 6] 1811 [27, 7] 445 [24, 15] 831 [16, 44] 1170 [23, 24] 734 [48, 15] 94 [45, 27] 849 [18, 29] 910 [23, 13] 1809 [46, 19] 2028 [51, 21] 750 [19, 37] 896 [15, 28] 146 [43, 27] 1247 [5, 29] 1045 [20, 21] 990 [52, 29] 2275 [24, 8] 562 [35, 6] 651 [21, 8] 313 [6, 13] 387 [27, 47] 1636 [19, 41] 1260 [29, 26] 369 [20, 7] 645 [21, 29] 1938 [31, 19] 905 [16, 51] 1163 [27, 8] 443 [36, 29] 2053 [24, 51] 232 [16, 53] 557 [25, 19] 639 [24, 46] 1362 [30, 25] 324 [27, 48] 463 [7, 47] 1506 [35, 27] 396 [18, 24] 435 [7, 16] 851 [7, 20] 644 [23, 28] 1543 [2, 21] 592 [24, 38] 979 [47, 23] 2694 [31, 23] 2245 [6, 2] 496 [23, 30] 1163 [27, 40] 231 [7, 34] 878 [4, 19] 813 [7, 6] 489 [37, 44] 122 [19, 48] 329 [30, 2] 454 [24, 35] 910 [6, 17] 306 [16, 21] 410 [20, 19] 1029 [19, 3] 168 [23, 44] 875 [27, 30] 106 [36, 3] 382 [19, 6] 225 [6, 24] 927 [24, 45] 812 [21, 37] 712 [27, 25] 274 [29, 41] 852 [12, 16] 958 [7, 32] 943 [15, 30] 318 [27, 42] 1624 [12, 30] 44 [24, 40] 396 [7, 52] 967 [16, 14] 1854 [51, 23] 926 [12, 23] 1157 [19, 49] 174 [6, 39] 1738 [47, 29] 3002 [27, 6] 567 [6, 44] 811 [36, 6] 265 [29, 5] 1045 [29, 45] 525 [29, 12] 1465 [6, 48] 104 [18, 19] 961 [7, 14] 1035 [21, 52] 535 [19, 17] 89 [24, 11] 1201 [27, 18] 464 [44, 19] 930 [35, 24] 910 [21, 47] 1060 [24, 49] 979 [43, 45] 404 [21, 45] 1415 [23, 23] 0 [24, 27] 521 [5, 24] 346 [4, 16] 132 [21, 26] 1568 [7, 26] 1254 [4, 24] 1402 [4, 15] 727 [19, 21] 318 [15, 35] 560 [7, 22] 851 [19, 45] 1316 [27, 31] 1210 [44, 24] 137 [33, 21] 317 [29, 31] 2575 [29, 42] 516 [6, 31] 711 [17, 30] 336 [27, 41] 1115 [6, 46] 2121 [19, 31] 928 [16, 47] 663 [51, 16] 1158 [6, 22] 985 [27, 32] 821 [16, 3] 778 [29, 49] 1655 [27, 5] 331 [30, 12] 44 [17, 23] 1434 [24, 41] 1444 [23, 41] 864 [28, 23] 1552 [7, 40] 554 [5, 27] 327 [40, 15] 600 [20, 29] 1142 [28, 16] 512 [35, 15] 559 [21, 41] 1569 [16, 23] 2038 [29, 44] 1312 [27, 43] 1252 [6, 3] 291 [35, 21] 745 [43, 15] 1621 [27, 22] 416 [45, 16] 1826 [48, 16] 520 [6, 30] 507 [19, 27] 504 [23, 33] 1356 [24, 6] 994 [7, 31] 1080 [7, 37] 275 [28, 27] 512 [24, 26] 918 [17, 6] 307 [6, 49] 391 [29, 50] 2002 [18, 6] 1024 [21, 31] 633 [52, 16] 388 [28, 19] 377 [21, 21] 0 [45, 23] 236 [6, 38] 1739 [5, 15] 701 [27, 13] 772 [6, 34] 1282 [46, 15] 1932 [6, 5] 890 [27, 1] 664 [24, 5] 346 [36, 27] 765 [24, 9] 1360 [7, 30] 495 [4, 6] 596 [15, 24] 820 [15, 33] 122 [27, 10] 66 [6, 45] 1410 [24, 43] 1093 [15, 17] 366 [31, 6] 687 [6, 6] 0 [37, 6] 756 [6, 18] 1023 [21, 16] 405 [23, 53] 1896 [18, 27] 461 [16, 17] 779 [20, 23] 704 [7, 25] 178 [29, 30] 1471 [24, 39] 1323 [35, 23] 1131 [23, 6] 1624 [21, 22] 991 ;

end;