CONNIE TENIN SU

Heurísticas construtivas para o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto

São Paulo
2017

CONNIE TENIN SU

Heurísticas construtivas para o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2017

CONNIE TENIN SU

Heurísticas construtivas para o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de concentração:
Engenharia de Sistemas Logísticos

Orientador: Prof. Dr. Miguel Cezar Santoro

São Paulo
2017

Catálogo-na-publicação

Su, Connie Tenin

Heurísticas construtivas para o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto / C. T. Su -- São Paulo, 2017.

137 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.Programação da produção 2.Scheduling 3.Heurística I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

Dedico este trabalho à minha família, por
sempre me apoiar e me incentivar.

AGRADECIMENTOS

Ao Prof. Dr. Miguel Cezar Santoro, por toda orientação, ajuda, paciência e disponibilidade ao longo deste trabalho e por sempre acreditar no meu potencial.

Ao meu marido Ronaldo, por sempre me apoiar, me incentivar e me ajudar quando necessário e por ser uma fonte de admiração constante.

Ao meu filho Alexandre, que nasceu e cresceu junto com este trabalho e todo dia me incentiva a ser uma pessoa melhor.

Aos meus pais, Wen e Ya Hui, e aos meus sogros, Haruo e Luiza, por toda ajuda e apoio e por todas as vezes que cuidaram do Alexandre com muito carinho.

Ao meu irmão Wiliam, pelo apoio.

À Dra. Denise Sato Yamashita, por compartilhar as instâncias de testes utilizadas em seu trabalho.

Aos meus colegas de programa e aos meus amigos, pela companhia e pelo incentivo ao longo deste período.

RESUMO

Este trabalho propõe uma heurística construtiva determinística e uma heurística construtiva probabilística para resolver o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*). Os algoritmos combinam a flexibilidade da atividade com a flexibilidade do recurso para selecionar a próxima atividade a ser programada. A data de início de uma atividade é a data mais cedo em que sua execução não gera o maior pico de utilização dos recursos ou a data mais cedo na qual o custo total do projeto for menor. A melhor versão das heurísticas foi obtida após o teste de várias regras de prioridade, conforme a revisão bibliográfica realizada. As heurísticas propostas foram testadas em 360 instâncias de testes e seus resultados foram comparados aos obtidos pela formulação matemática baseada em *strip packing* e restrições disjuntivas implementada no programa CPLEX. A heurística construtiva determinística gera uma solução viável rapidamente, porém de baixa qualidade. Já a heurística construtiva probabilística gera soluções ótimas ou próximas da ótima para problemas pequenos ou para problemas fáceis e gera soluções muito melhores do que o CPLEX na metade do tempo computacional para os problemas médios e grandes ou para problemas difíceis. Dado os bons resultados obtidos e à implementação no programa VBA for Microsoft Excel, a heurística construtiva probabilística proposta é um método bom e prático para resolução do RACPT.

Palavras-chave: Programação de projetos. Custo de disponibilidade de recursos. Custo de penalidade por atraso. Heurística construtiva.

ABSTRACT

This work proposes a deterministic constructive heuristic and a probabilistic constructive heuristic for solving the resource availability cost problem with tardiness (RACPT). The algorithms combine the flexibility of an activity with the flexibility of a resource to select the next activity to be scheduled. The start time of the activity is the earliest date in which the activity's execution does not create resources usage peak or the earliest date with the lowest total project cost. We tested several priority rules according to the literature review and determined the best version of the heuristics. Afterwards, we tested the proposed heuristics in 360 instances and compared its results with the solutions obtained by the optimization software CPLEX. The RACPT implementation on CPLEX utilized a mathematical formulation based on strip packing concepts and disjunctive constraints. The computational results showed that the deterministic constructive heuristic generates feasible solutions of poor quality in low computational time. The probabilistic constructive heuristic achieved better results. For small instances or easy problems, it found optimal or near-optimal solutions. For medium and large instances or hard problems, it obtained better results than CPLEX in half-computational time. We believe that the probabilistic constructive heuristic is a good and practical method for solving the RACPT. The proposed algorithm produced good results in reasonable computational time and was implemented on the popular software VBA for Microsoft Excel.

Keywords: Project scheduling. Resource availability cost. Tardiness cost. Constructive heuristic.

LISTA DE FIGURAS

Figura 1 – Projeto exemplo com 12 atividades representadas nos nós. O valor à esquerda é o número da atividade i , o valor do meio é sua duração d_i e o valor à direita é sua necessidade de recursos $ri1$. Os arcos indicam as relações de precedência.....	13
Figura 2 – Perfil de utilização do recurso 1 do projeto exemplo no programa com data mais cedo de início.....	14
Figura 3 – Solução ótima do projeto exemplo como RCPSP, considerando disponibilidade do recurso 1 igual a 4 unidades.....	14
Figura 4 – Perfil de utilização do recurso 1 do projeto exemplo após nivelamento de recursos.....	16
Figura 5 – Tipos de problema de programação de projetos.	17
Figura 6 – Principais artigos sobre RACP, RACPT e MMRACP.	25
Figura 7 – Pseudocódigo da heurística construtiva determinística.....	56
Figura 8 – Pseudocódigo do procedimento ACHA_MELHOR_S.....	57
Figura 9 – Pseudocódigo do procedimento SIMULA_FO.....	57
Figura 10 – Pseudocódigo do procedimento BACKWARD_FORWARD.	58
Figura 11 – Exemplo de ordenação da lista de atividades candidatas C combinando FA_i e FR_k	59
Figura 12 – Exemplo de probabilidades com viés para uma lista com 6 atividades candidatas.....	59
Figura 13 – Pseudocódigo da heurística construtiva probabilística.	62

LISTA DE TABELAS

Tabela 1 – Datas mais cedo, mais tarde e a folga total das atividades do projeto exemplo calculadas pelo CPM. As atividades em azul estão no caminho crítico do projeto.	13
Tabela 2 – Combinações de regra de prioridade e regra de desempate selecionadas.	65
Tabela 3 – Tempo limite de execução da heurística probabilística e do CPLEX (em segundos).....	66
Tabela 4 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 4 recursos.....	67
Tabela 5 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 6 recursos.....	68
Tabela 6 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 8 recursos.....	69
Tabela 7 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 4 recursos.....	70
Tabela 8 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 6 recursos.....	71
Tabela 9 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 8 recursos.....	72
Tabela 10 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 4 recursos.....	73
Tabela 11 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 6 recursos.....	74
Tabela 12 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 8 recursos.....	75
Tabela 13 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 120 atividades e $DF=1,2$	76
Tabela 14 – Resultados da comparação entre as regras de prioridade para a heurística construtiva probabilística – instâncias de teste com 30 atividades e 8 recursos (900 segundos).....	77
Tabela 15 – Resultados da comparação entre as heurísticas e o CPLEX (30 atividades e 4 recursos).	78

Tabela 16 – Resultados da comparação entre as heurísticas e o CPLEX (30 atividades e 6 recursos).	79
Tabela 17 – Resultados da comparação entre as heurísticas e o CPLEX (30 atividades e 8 recursos).	80
Tabela 18 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 4 recursos).	81
Tabela 19 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 6 recursos).	82
Tabela 20 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 8 recursos).	83
Tabela 21 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 4 recursos).	84
Tabela 22 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 6 recursos).	85
Tabela 23 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 8 recursos).	86
Tabela 24 – Resultados da comparação entre as heurísticas e o CPLEX (7.200 segundos) – instâncias de teste com 90 atividades e $DF=1,2$	87
Tabela 25 – Resultados da comparação entre as heurísticas e o CPLEX (3.600 segundos) – instâncias de teste com 120 atividades e $DF=1,2$	88
Tabela 26 – Resultados da comparação entre as heurísticas e o CPLEX (7.200 segundos) – instâncias de teste com 120 atividades e $DF=1,2$	89
Tabela 27 – Resumo da comparação entre as regras de prioridade para a heurística construtiva determinística.....	90
Tabela 28 – Resumo da comparação entre as regras de prioridade para a heurística construtiva probabilística (instâncias com 30 atividades e 8 recursos).	91
Tabela 29 – Resumo da comparação entre as heurísticas e o CPLEX.	92
Tabela 30 – Resumo da comparação entre a heurística construtiva probabilística (900 s) e o CPLEX (1.800 s) – instâncias de teste com 30 atividades.	94
Tabela 31 – Resumo da comparação entre a heurística construtiva probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 60 atividades.	95
Tabela 32 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 90 atividades.	96

Tabela 33 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (7.200 s) – instâncias de teste com 90 atividades e DF=1,2.	97
Tabela 34 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 120 atividades e DF=1,2.	97
Tabela 35 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (3.600 s) e o CPLEX (7.200 s) – instâncias de teste com 120 atividades e DF=1,2.	98

SUMÁRIO

1	INTRODUÇÃO	12
1.1	MOTIVAÇÃO	17
1.2	DESCRIÇÃO DO PROBLEMA	19
1.3	OBJETIVO, JUSTIFICATIVA E MÉTODO	21
1.4	ORGANIZAÇÃO DO TRABALHO	22
2	REVISÃO DA LITERATURA	24
2.1	PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS (RACP)	27
2.1.1	Métodos exatos para o RACP	27
2.1.2	Métodos heurísticos para o RACP	29
2.2	PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS E COM MÚLTIPLOS MODOS DE EXECUÇÃO (MMRACP)	41
2.2.1	Métodos exatos para o MMRACP	41
2.2.2	Métodos heurísticos para o MMRACP	42
2.3	PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS E CUSTO DE PENALIDADE POR ATRASO NO TÉRMINO DO PROJETO (RACPT)	46
3	MODELAGEM MATEMÁTICA PARA O PROBLEMA	49
3.1	MÉTODO OTIMIZANTE	49
3.2	MÉTODOS HEURÍSTICOS	52
3.2.1	Heurística construtiva determinística	52
3.2.2	Heurística construtiva probabilística	58
4	EXPERIMENTOS COMPUTACIONAIS	63
4.1	INSTÂNCIAS DE TESTES	63
4.2	PROCEDIMENTO EXPERIMENTAL	64
4.3	RESULTADOS	66

5	DISCUSSÃO	90
6	CONCLUSÕES	100
	REFERÊNCIAS BIBLIOGRÁFICAS	102
	APÊNDICE A – Código fonte para a heurística construtiva em VBA	106
	APÊNDICE B – Código fonte para os experimentos no CPLEX através do IBM ILOG CPLEX Optimization Studio	135

1 INTRODUÇÃO

De acordo com Pinedo (2010), a programação da produção começou a ganhar importância no começo do século XX, com Henry Gantt e outros pioneiros. Porém, os primeiros estudos foram publicados somente na década de 50. Na década de 60, foram publicados diversos trabalhos considerando programação dinâmica e programação inteira. Nos anos 70, o foco principal era a hierarquia de complexidade dos problemas de programação. Já na década de 80, houve um aumento no interesse em problemas de programação estocásticos, mas também houve muitos trabalhos em diversas direções. O avanço dos computadores e sua crescente utilização em empresas levaram ao desenvolvimento de sistemas de programação que gerassem programas passíveis de serem usados na prática.

Dentre as principais classes de problemas de programação, há a programação de projetos, que consiste em determinar a menor data de término de um projeto composto por n atividades sujeitas a um conjunto de relações de precedência $(i, j) \in H$ do tipo fim – início e a um número ilimitado de recursos em paralelo (PINEDO, 2010). Cada atividade i possui uma duração d_i determinística e necessita de um conjunto de recursos renováveis k ($k = 1, \dots, m$) para ser executada, sendo que a quantidade necessária de cada recurso r_{ik} é constante ao longo da execução de cada atividade. Os recursos renováveis são aqueles que ficam disponíveis para uso a cada período de tempo, como por exemplo, trabalhadores e máquinas.

Um dos primeiros métodos para resolver esse problema de programação de projetos sem restrição de recursos é o método do caminho crítico (CPM – *Critical Path Method*), no qual obtém-se a menor data de término para o projeto (*makespan*) ao programar suas atividades na data mais cedo de início possível respeitando as relações de precedência. Da mesma forma, pode-se calcular a data mais tarde de início e fim possível das atividades, partindo da última atividade do projeto com data fim igual a menor data de término do projeto. Com as datas mais cedo e mais tarde de início de cada atividade, é possível calcular sua folga total, que mede o quanto cada atividade pode ter seu início atrasado sem atrasar o término do projeto. As atividades com folga total igual a zero estão no caminho crítico, pois qualquer atraso na sua data de início acarreta em atraso na conclusão do projeto. A Figura 1 mostra um projeto exemplo

com 12 atividades e 1 tipo de recurso. A Tabela 1 traz as datas mais cedo, mais tarde e a folga total das atividades deste projeto exemplo.

Figura 1 – Projeto exemplo com 12 atividades representadas nos nós. O valor à esquerda é o número da atividade i , o valor do meio é sua duração d_i e o valor à direita é sua necessidade de recursos r_{i1} . Os arcos indicam as relações de precedência.

Fonte: Klein (2000).

Tabela 1 – Datas mais cedo, mais tarde e a folga total das atividades do projeto exemplo calculadas pelo CPM. As atividades em azul estão no caminho crítico do projeto.

Atividade	i	1	2	3	4	5	6	7	8	9	10	11	12
Data início mais cedo	ES _i	0	0	0	0	3	5	6	7	7	7	12	16
Data fim mais cedo	EF _i	0	3	5	1	6	7	10	12	13	11	16	16
Data início mais tarde	LS _i	0	2	0	4	5	5	8	7	10	12	12	16
Data fim mais tarde	LF _i	0	5	5	5	8	7	12	12	16	16	16	16
Folga total	TSL _i	0	2	0	4	2	0	2	0	3	5	0	0

Fonte: Klein (2000).

Considerando o programa com as atividades iniciando na data mais cedo, é possível calcular uma matriz de utilização de recurso do projeto exemplo. Assim, no intervalo de tempo $[0,1[$, ocorre o pico de utilização do recurso, pois as atividades 2, 3 e 4 estão em execução e geram uma demanda total de 8 unidades do recurso 1. A Figura 2 mostra o diagrama de Gantt com o perfil de utilização do recurso do projeto exemplo no programa mais cedo.

Figura 2 – Perfil de utilização do recurso 1 do projeto exemplo no programa com data mais cedo de início.

No entanto, o mais comum é que haja restrição na disponibilidade de recursos (por exemplo, devido a custos), gerando o problema de programação de projetos com restrição de recursos (RCPSP – *Resource Constrained Project Scheduling Problem*), um dos problemas mais estudados em programação de projetos. No RCPSP, dado uma quantidade limite para cada recurso, o objetivo é determinar qual a menor duração possível para o projeto (*makespan*), respeitando as relações de precedência. As variáveis de decisão são os instantes de término (ou início) das atividades.

A Figura 3 mostra o programa ótimo do projeto exemplo no formato de diagrama de Gantt caso a disponibilidade do recurso 1 seja igual a 4 unidades. A data de término do projeto passa a ser igual a 22, maior do que o programa na data mais cedo em 6 unidades.

Figura 3 – Solução ótima do projeto exemplo como RCPSP, considerando disponibilidade do recurso 1 igual a 4 unidades.

Há também situações em que a data de término do projeto é determinada a priori, por exemplo, quando a data de término e altas penalidades por atraso são definidas em contrato (KLEIN, 2000). Nesses casos, é importante buscar o nivelamento de recursos, pois há um custo alto para disponibilizar picos de recursos (por exemplo, custo de contratação e demissão de funcionários) ou então altas variações podem ser inviáveis (por exemplo, compra ou aluguel de máquinas de grande porte). Além disso, é importante otimizar o uso dos recursos, evitando momentos de pico e de ociosidade, melhorando a eficiência do projeto como um todo. O uso nivelado de recursos também beneficia o processo de compra de materiais, dado a menor variação ao longo do tempo. Assim, surgem os problemas de nivelamento de recursos (RLP – *Resource Leveling Problem*).

No RLP, o objetivo é obter a menor variação no consumo de recursos ao longo do tempo, ou seja, obter um perfil de utilização de recursos mais próximo possível de um retângulo. Existem diversas funções objetivos para o RLP, sendo a mais tradicional minimizar a soma dos quadrados da utilização de recursos em cada período de tempo e considerando todos os tipos de recursos. As variáveis de decisão são as quantidades utilizadas de cada recurso (que variam com o tempo) e o instante de término (ou início) de cada atividade.

Um caso especial de problemas de nivelamento de recursos (RLP) ocorre quando o objetivo é minimizar o pico de utilização de recursos, e consequentemente o custo total de recursos do projeto. Este caso especial é chamado de problema de programação de projetos com custo de disponibilidade de recursos (RACP – *Resource Availability Cost Problem*). As variáveis de decisão do RACP são as quantidades disponíveis de cada recurso, constantes ao longo do projeto, e o instante de término (ou início) de cada atividade. As restrições são idênticas às do RLP, com a diferença de que a quantidade utilizada é trocada pela quantidade disponível de cada recurso.

Möhring (1984) mostrou as relações de dualidade do RACP com o problema de programação de projetos com restrição de recursos (RCPSP). A formulação matemática dos dois problemas é similar, com diferença na função objetivo e na restrição do RACP que limita a duração do projeto à data de entrega e que não existe no RCPSP.

Hsu e Kim (2005) explicam que o RACP é importante porque há muitos projetos reais cujas datas de entrega são definidas por contrato ou são decididas pela alta diretoria

da empresa. Assim, com a definição da data de entrega, o objetivo da programação de projetos passa a ser programar as atividades de forma a minimizar o custo de execução do projeto, ou de certa forma, minimizar o pico de utilização dos recursos. Um exemplo de aplicação prática do RACP é o projeto de desenvolvimento de um veículo (desenho e engenharia), que motivou o trabalho de Hsu e Kim (2005). Möhring (1984) mostrou que o RACP é um problema NP-difícil. Drexel e Kimms (2001) mencionam que a solução do RACP para diferentes datas de entrega gera *tradeoffs* interessantes entre tempo (prazo) e custo, o que é uma informação importante para a precificação de um projeto.

No projeto exemplo, mantendo a data de término do projeto na data mais cedo, é possível alterar a data de início das atividades com folga total maior que zero de forma a diminuir o pico de utilização dos recursos e melhorar o nivelamento dos recursos. Por exemplo, ao postergar a data de início das atividades 4, 5, 7 e 10, pode-se obter um programa com pico de utilização de recurso igual a 6, diminuindo o pico em 2 unidades, com a mesma data de término (vide Figura 4).

Figura 4 – Perfil de utilização do recurso 1 do projeto exemplo após nivelamento de recursos.

Por outro lado, há situações em que o custo por atraso no término do projeto é inferior ao custo dos recursos, de modo que o custo total do projeto pode ser reduzido quando o atraso no término gera uma economia de recursos. Esse é o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*), no qual o custo de atraso é adicionado na função objetivo.

O RACP também é um caso particular do problema de programação de projetos com custo de disponibilidade de recursos com múltiplos modos de execução (MMRACP – *Multi-Mode Resource Availability Cost Problem*), no qual uma atividade pode ser executada de vários modos. Cada modo requer uma quantidade de recursos e possui durações diferentes. As variáveis de decisão do MMRACP são as quantidades disponíveis de cada recurso, o instante de término de cada atividade e o modo de execução dela.

A Figura 5 apresenta um resumo dos tipos de problema de programação de projetos apresentados.

Figura 5 – Tipos de problema de programação de projetos.

Tipo de problema	Nome	Função objetivo	Variáveis de decisão
RCPSP	Problema de programação de projetos com restrição de recursos	Menor <i>makespan</i>	Instante de término das atividades
RLP	Problema de nivelamento de recursos	Menor variação no consumo de recursos ao longo do tempo	Quantidades utilizadas de cada recurso em cada instante de tempo e o instante de término das atividades
RACP	Problema de programação de projetos com custo de disponibilidade de recursos	Menor custo total de recursos	Quantidades disponíveis de cada recurso (constantes ao longo do projeto) e o instante de término das atividades
RACPT	Problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto	Menor custo total (recursos mais atraso)	Quantidades disponíveis de cada recurso (constantes ao longo do projeto) e o instante de término das atividades
MMRACP	Programação de projetos com custo de disponibilidade de recursos com múltiplos modos de execução	Menor custo total de recursos	Quantidades disponíveis de cada recurso (constantes ao longo do projeto), o instante de término das atividades e seu modo de execução

Fonte: Elaboração própria.

1.1 MOTIVAÇÃO

A pesquisa conduzida por Liberatore; Pollack-Johnson e Smith (2001) mostra que 83% dos profissionais de gestão de projetos utilizam *software* de gestão de projetos para planejamento e controle. Mostra também que na construção civil, o nivelamento de recursos é utilizado por 58% para planejamento e por 44% para controle. Segundo Kastor e Sirakoulis (2009), a maioria dos *softwares* de gestão de projetos emprega

heurísticas baseadas em regras de prioridade (*priority rule-based*) para nivelamento de recursos. Este tipo de heurística é composto pelo esquema de programação (*scheduling scheme*), que pode ser em série ou em paralelo, e pela regra de prioridade. Os *softwares* de gestão de projetos não revelam qual esquema de programação ou regra de prioridade é utilizado.

Kastor e Sirakoulis (2009) avaliaram a eficiência no nivelamento de recursos de três pacotes populares de programação de projetos (Primavera P6.0, Microsoft Project 2007 e Open Workbench 1.1.6) comparando com a duração de dois projetos reais de construção civil. O Primavera Project Planner permite ao usuário escolher a regra de prioridade, porém não explica qual a utilidade de alterar ou utilizar mais de uma regra. O Microsoft Project e o Open Workbench utilizam somente o método padrão embutido. Considerando a configuração padrão dos três *softwares*, o Primavera P6 obteve o melhor resultado, com desvio médio em relação ao método do caminho crítico (CPM - *Critical Path Method*) de 41,11%, seguido pelo Microsoft Project com desvio de 46,14% e pelo Open Workbench com desvio de 167,79%. Além disso, o Primavera P6 é o mais efetivo por permitir que o usuário cadastre inúmeras regras de prioridade. O Microsoft Project não permite restrição de capacidade para materiais e o usuário tem que considerá-los como recursos humanos. O Open Workbench foi muito lento para gerar a programação. As versões anteriores dos três programas foram avaliadas e produziram os mesmos resultados. Os autores sugerem que os *softwares* de gestão de projetos devem incorporar métodos mais eficientes para programação com restrição de recursos; buscar a redução de inconvenientes como o *phantom float*, que é a diferença entre a folga total restante teórica e a real; e tenham outros objetivos além de minimizar o instante mínimo de término do projeto (*makespan*), como maximizar o valor presente líquido.

Através deste e de outros estudos similares podemos ver que os métodos padrão dos *softwares* de gestão de projetos apresentam resultados muito distantes do ótimo e, portanto, devem ser melhorados.

Além disso, há poucos trabalhos publicados sobre o RACP, o RACPT e o MMRACP apesar da relevância prática destes problemas, principalmente se compararmos com o RCPSp e até com o RLP, mostrando oportunidade para o desenvolvimento de novos métodos de solução. Em particular, para o RACPT foram propostas somente duas meta-heurísticas, faltando o desenvolvimento de heurísticas construtivas.

1.2 DESCRIÇÃO DO PROBLEMA

Este trabalho estudará o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*). Este problema é uma variação do problema de programação de projetos com custo de disponibilidade de recursos (RACP – *Resource Availability Cost Problem*), conhecido também como problema de investimento em recursos (RIP – *Resource Investment Problem*) ou como problema do tempo escasso, introduzido por Möhring (1984).

O RACPT consiste em: seja um projeto caracterizado por um conjunto de n atividades sem interrupção, que possuem um único modo de execução, duração d_i determinística e sujeitas ao conjunto de relações de precedência $(i, j) \in H$ do tipo fim – início sem intervalo, representadas em uma rede acíclica com atividades nos nós. As atividades fictícias 0 e $n + 1$ representam o início e o término do projeto, com duração igual a zero e necessidade de recursos igual a zero. Cada atividade i necessita de r_{ik} unidades do recurso renovável de tipo k ($k = 1, \dots, m$) para ser executada e essa quantidade necessária de cada recurso r_{ik} é constante ao longo da execução de cada atividade. Cada recurso tem disponibilidade a_k constante ao longo do projeto e uma função de custo linear e não decrescente $c_k(a_k)$. Há um custo C_T constante para cada unidade de tempo de atraso T em relação à data de entrega D do projeto, sendo que Mx é a data limite para término do projeto ($Mx \geq D$). O problema consiste em: qual a programação das atividades, respeitando as relações de precedência, que gera o menor custo total do projeto? O custo total é igual à soma do custo de disponibilidade dos recursos $c_k(a_k)$ e do custo de penalidade por atraso $C_T(T)$. As variáveis de decisão do RACPT são as quantidades disponíveis de cada recurso a_k e as datas de início S_i (ou datas de término F_i) das atividades.

O RACPT foi estudado pela primeira vez por Shadrokh e Kianfar (2007). Eles apresentaram a seguinte formulação matemática, como um problema de programação linear inteira mista.

Além dos parâmetros e das variáveis já apresentados, considere também as variáveis binárias x_{it} , que determinam se a atividade i começa a ser executada no instante t

($x_{it} = 1$). Caso contrário o valor de x_{it} é igual a zero. Assim, a data de início de uma atividade i é definida por: $S_i = \sum_{t=0}^{Mx} tx_{it}$. O modelo matemático consiste de:

$$\min \sum_{k=1}^m c_k a_k + C_T \max\{0, S_{n+1} - D\} \quad (1.1)$$

sujeito a:

$$\sum_{t=0}^{Mx} tx_{jt} \geq d_i + \sum_{t=0}^{Mx} tx_{it}, \quad \forall (i, j) \in H \quad (1.2)$$

$$\sum_{i=1}^n \sum_{u=\max(t-d_i+1, 0)}^t r_{ik} x_{iu} \leq a_k, \quad t = 0, \dots, Mx - 1; k = 1, \dots, m \quad (1.3)$$

$$\sum_{t=0}^{Mx} x_{it} = 1, \quad i = 0, \dots, n + 1 \quad (1.4)$$

$$x_{00} = 1 \quad (1.5)$$

$$x_{it} \in \{0, 1\}, \quad i = 0, \dots, n + 1, t = 0, \dots, Mx \quad (1.6)$$

$$a_k \geq 0, \quad k = 1, \dots, m \quad (1.7)$$

A função objetivo (1.1) visa minimizar o custo total de disponibilidade de todos os recursos mais a penalidade por atraso e contém também o cálculo do atraso. Se o projeto for concluído antes da data de entrega D , o atraso T é igual a zero, caso contrário, o atraso T é igual a diferença entre a data de término do projeto S_{n+1} e a data de entrega D . As restrições (1.2) levam em consideração as relações de precedência entre os pares de atividades (i, j) , onde j é uma atividade imediatamente sucessora de i . As restrições (1.3) limitam o consumo total de cada recurso em cada período de tempo à capacidade disponível do recurso. O conjunto de restrições (1.4) garantem que cada atividade i tenha somente uma única data de início. A restrição (1.5) define que a atividade fictícia 0 inicia sua execução no instante 0. Os conjuntos de restrições (1.6) e (1.7) definem o domínio das variáveis de decisão. Todos os valores são inteiros, com exceção dos parâmetros de custo unitário.

O segundo trabalho desenvolvido para o RACPT é o de Van Peteghem e Vanhoucke (2015). Eles apresentaram apenas um modelo conceitual, similar ao de Shadrokh e Kianfar (2007):

$$\min \sum_{k=1}^m c_k a_k + C_T \max\{0, S_{n+1} - D\} \quad (1.8)$$

sujeito a:

$$S_i + d_i \leq S_j, \quad \forall (i, j) \in H \quad (1.9)$$

$$r_k(S, t) \leq a_k, \quad k = 1, \dots, m; t = 0, \dots, Mx - 1 \quad (1.10)$$

$$S_0 = 0 \quad (1.11)$$

A função objetivo (1.8) minimiza o custo total de disponibilidade dos recursos mais a penalidade por atraso e é igual a de Shadrokh e Kianfar (2007). As restrições (1.9) garantem as relações de precedência entre os pares de atividades (i, j) . As restrições (1.10) limitam o consumo total de cada recurso em cada período de tempo $r_k(S, t)$ à capacidade disponível do recurso. Por fim, a restrição (1.11) define que a atividade fictícia 0 inicia sua execução no instante 0.

1.3 OBJETIVO, JUSTIFICATIVA E MÉTODO

O objetivo deste trabalho é desenvolver heurísticas construtivas (determinística e probabilística) para resolver o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*), que foi definido no item 1.2. Espera-se que as heurísticas propostas gerem uma solução ótima ou próxima da ótima.

A ideia principal das heurísticas consiste em combinar a flexibilidade da atividade com a flexibilidade dos recursos para selecionar a próxima atividade a ser programada. A flexibilidade da atividade é determinada através de regras de prioridade. Quanto maior a prioridade da atividade conforme a regra escolhida, menos flexível é a atividade. Já a flexibilidade do recurso mede sua intensidade de utilização em relação à sua necessidade média e ao seu custo unitário. Quanto maior a intensidade de utilização e quanto maior o custo unitário, menos flexível é o recurso. A data de início da atividade é a data mais cedo em que a execução da atividade não gera o maior pico de utilização de um ou mais recursos ou a data mais cedo em que o custo total do projeto for menor.

Primeiramente, serão realizados testes para encontrar a melhor regra de prioridade para as heurísticas. Em seguida, os resultados dos modelos desenvolvidos serão validados e comparados com soluções ótimas ou com limitantes superiores e inferiores obtidos no programa CPLEX. A implementação no CPLEX utiliza uma formulação matemática de programação linear inteira mista baseado em conceitos de *strip packing* e restrições disjuntivas. Os testes computacionais serão realizados nas mesmas instâncias de Yamashita; Armentano e Laguna (2006).

A justificativa deste trabalho é contribuir com métodos de aplicação prática para enriquecer o conjunto de técnicas disponíveis para resolver o RACPT. Apesar da aplicação prática deste tipo de problema, apenas duas meta-heurísticas foram desenvolvidas para resolvê-lo. A implementação em um *software* popular como o Microsoft Excel facilita a sua utilização.

Com relação a metodologia, este trabalho utiliza a modelagem quantitativa para descrever o RACPT, mais especificamente um modelo determinístico de programação matemática linear inteira mista:

Modelos quantitativos são modelos abstratos descritos em linguagem matemática e computacional, que utilizam técnicas analíticas (matemáticas, estatísticas) e experimentais (simulação) para calcular valores numéricos das propriedades do sistema em questão (MORABITO; PUREZA, 2012, p. 171).

Morabito e Pureza (2012) apresentaram os quatro tipos de metodologia de pesquisa com base em modelagem quantitativa para gestão de produção e operações. Este trabalho utiliza a pesquisa axiomática quantitativa normativa, pois o modelo do RACPT é idealizado, ou seja, não é empírico, e o objetivo é desenvolver um novo método de resolução do tipo heurística construtiva, que apresente desempenho melhor do que outro método já existente (nesse caso, os métodos do *software* de otimização CPLEX).

1.4 ORGANIZAÇÃO DO TRABALHO

No capítulo 2, é apresentada uma revisão da literatura sobre RACP, MMRACP e RACPT. São apresentados os principais trabalhos publicados e uma descrição do método de solução proposto.

No capítulo 3, é apresentada a modelagem matemática para o RACPT. Primeiro é mostrado o método otimizante, cuja formulação matemática é baseada em *strip*

packing e restrições disjuntivas. Esta formulação foi utilizada nos testes realizados no *software* de otimização CPLEX. Em seguida, são apresentados a descrição e o pseudocódigo dos métodos heurísticos desenvolvidos neste trabalho: uma heurística construtiva determinística e uma heurística construtiva probabilística.

Já o capítulo 4 traz uma descrição das instâncias de testes e dos experimentos computacionais e apresenta os resultados obtidos.

O capítulo 5 mostra as discussões decorrentes dos resultados obtidos, com uma avaliação de desempenho dos métodos propostos.

Por fim, o capítulo 6 contém as conclusões deste trabalho.

2 REVISÃO DA LITERATURA

Há relativamente poucos trabalhos sobre o RACP, o MMRACP e o RACPT. Foram desenvolvidos tanto métodos exatos quanto heurísticos para resolvê-los. A Figura 6 mostra os principais trabalhos desenvolvidos e as próximas seções trazem um detalhamento dos principais trabalhos publicados e mencionam outros trabalhos menos relevantes.

Outros trabalhos interessantes são as *surveys* de Neumann; Schwindt e Zimmermann (2002) e Hartmann e Briskorn (2010), que mostram os principais trabalhos desenvolvidos sobre o RCPSP e suas variantes, dentre elas o RACP.

Figura 6 – Principais artigos sobre RACP, RACPT e MMRACP.

Problema	Autor(es)	Data	Método	Características do método	Observações
RACP	Möhring	1984	Método exato	Solução ótima é determinada por um subconjunto de todos os pedidos parciais viáveis. Baseado nos algoritmos da teoria de grafos.	Teste: projeto com 16 atividades e 4 recursos.
RACP	Demeulemeester	1995	Método exato	Determina a quantidade mínima necessária de disponibilidade para cada recurso (ponto eficiente) e resolve o RCPSP com <i>branch-and-bound</i> considerando esses pontos eficientes, até que o <i>makespan</i> seja menor ou igual a data de entrega.	Mais rápido que o de Möhring (1984). Resolve problemas com 20 atividades e 6 tipos de recursos.
RACP	Rangaswamy	1998	Método exato	Algoritmo <i>branch-and-bound</i> .	Resultados não conclusivos.
RACP	Rodrigues e Yamashita	2010	Método exato	Baseado em Demeulemeester (1995). Solução inicial é obtida através de uma heurística. Propõe novos limitantes para a ramificação, diminuindo o espaço de busca.	Teste: projetos com 30 atividades e 4 recursos. Mais rápido que o de Demeulemeester (1995).
RACP	Drex1 e Kimms	2001	Heurística	Dois limitantes inferiores (relaxação Lagrangeana e geração de colunas), que geram soluções viáveis (limitantes superiores).	Teste: projetos com 30 atividades e 4 recursos. Método de geração de colunas é melhor.
RACP	Yamashita; Armentano e Laguna	2006	Heurística	Busca dispersa (<i>scatter search</i>) e método de combinação de soluções baseado em <i>path relinking</i> .	Teste: projetos com 30 a 120 atividades e 4 a 8 recursos e nas instâncias de Drex1 e Kimms (2001).
RACP	Ranjbar; Kianfar e Shadrokh	2008	Heurística	<i>Path relinking</i> e algoritmo genético.	Teste: nas instâncias de Yamashita; Armentano e Laguna (2006). <i>Path relinking</i> é melhor.
RACP	Van Peteghem e Vanhoucke	2013	Heurística	Algoritmo de sistema imunológico artificial (similar ao algoritmo genético, mas sem a etapa de cruzamento).	Teste: projetos com 30 a 120 atividades e 4 recursos.
RACP	Zhu et al.	2017	Heurística	Heurística de busca multi-início iterativa. Dividiram o RACP em dois subproblemas (problema de sequenciamento e problema de decisão de recursos).	Teste: projetos com 30 a 120 atividades e 4 a 8 recursos. Melhor regra: maior necessidade de recurso acumulada.

continua

conclusão

Problema	Autor(es)	Data	Método	Características do método	Observações
RACP	Neumann e Zimmermann	1999	Heurística	Heurística polinomial baseada em regras de prioridade.	Teste: projetos com 10 a 500 atividades e 1 a 5 recursos. Melhor regra: mínima ordem de processamento.
RACP	Neumann e Zimmermann	2000	Método exato e heurística	Algoritmo <i>branch-and-bound</i> , <i>branch-and-bound</i> truncado e heurística de melhoria baseada na busca tabu.	Testes: nas instâncias de Neumann e Zimmermann (1999). Heurística de melhoria (busca tabu) é melhor.
MMRACP	Yamashita e Morabito	2009	Método exato	Combinaram o algoritmo exato de Demeulemeester (1995) para o RACP e o algoritmo exato de Sprecher e Drexler (1998) para o MMRCPPSP, gerando também curvas com <i>tradeoff</i> entre data de término do projeto e custo dos recursos.	Teste: projetos com 15 atividades, 4 recursos e 3 modos de execução.
MMRACP	Hsu e Kim	2005	Heurística	Regras de prioridade e de liberação aplicadas dentro de um esquema de geração de programa em série. Seleção simultânea da atividade a ser programada, seu modo de execução e seu instante de início.	Teste: projetos com 12 a 30 atividades e 4 recursos.
RACPT	Shadrokh e Kianfar	2007	Heurística	Algoritmo genético. Representam a solução como uma lista de sequência de atividades e uma lista de capacidade disponível de recursos e usam esquema de geração de programa para converter em um programa.	Teste: projetos com 10 a 200 atividades e 4 recursos e nas instâncias de Drexler e Kimms (2001).
RACP e RACPT	Van Peteghem e Vanhoucke	2015	Heurística	Algoritmo de otimização de ervas invasivas (estratégia de busca baseada no comportamento natural das ervas em colonizar e achar um local adequado para crescimento e reprodução).	Teste: projetos com 30 a 120 atividades e 4 recursos.

Fonte: Elaboração própria.

2.1 PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS (RACP)

2.1.1 Métodos exatos para o RACP

Möhring (1984) desenvolveu um método exato baseado no procedimento que Radermacher apud Möhring (1984) criou para solucionar o RCPSP. O método de Möhring (1984) considera pedidos parciais viáveis para o RACP. Um pedido parcial viável estende um pedido parcial composto pelas relações de precedência e respeita a data de entrega do projeto. A solução ótima do custo de disponibilidade para todas as possibilidades de requisições de recursos é determinada por um subconjunto de todos os pedidos parciais viáveis. Möhring (1984) apresentou todos os menores subconjuntos que possuem esta propriedade, chamados de sistemas minimamente dominados. O método proposto é baseado nos algoritmos da teoria de grafos, para comparabilidade de grafos e reconhecimento e orientação de intervalos de grafos. O método proposto foi testado em um projeto pequeno de construção de ponte, com 16 atividades e 4 tipos de recursos.

Demeulemeester (1995) desenvolveu um algoritmo exato baseado no algoritmo *depth-first* (busca em profundidade) *branch-and-bound* construído para o RCPSP em Demeulemeester e Herroelen (1992). No primeiro passo, calcula-se o limitante inferior para a capacidade disponível de cada recurso, igual a maior necessidade desse recurso para executar qualquer uma das atividades do projeto. No passo seguinte, o limitante inferior de cada recurso é melhorado: considera-se que a disponibilidade de um recurso é igual ao limitante inferior e que a disponibilidade dos demais recursos é infinita. Então é verificado se o problema de decisão RCPSP possui uma solução viável usando o procedimento *branch-and-bound* e considerando a data de entrega do projeto. Se sim, então o limitante inferior não pode ser melhorado, se não, aumenta-se a disponibilidade do recurso em uma unidade e verifica-se novamente a solução do problema de decisão. No terceiro passo, são gerados os conjuntos eficientes parciais, compostos por dois elementos: o número mínimo de unidades de um recurso qualquer (segundo elemento) necessário para gerar um programa viável quando há somente uma certa quantidade disponível do primeiro tipo de recurso (primeiro elemento), considerando os limitantes inferiores calculados. Depois, todos os

conjuntos eficientes parciais são combinados para criar os conjuntos de pontos eficientes (soluções candidatas). Por fim, é resolvido um problema de decisão RCPSP com o conjunto de pontos eficientes mais barato. Se o *makespan* for menor ou igual a data de entrega do projeto, a solução ótima foi encontrada, se não, a disponibilidade de um recurso é aumentada em uma unidade, até que a primeira solução viável seja encontrada.

Demeulemeester (1995) comparou a performance de seu método exato com o de Möhring (1984) e mostrou que o seu procedimento é mais rápido. O autor também realizou testes em um conjunto de problemas adaptado de Patterson (1984), novamente com bons resultados. Os testes computacionais mostraram que o algoritmo exato proposto consegue resolver problemas com no máximo 6 tipos de recursos e 15 a 20 atividades.

Rangaswamy apud Yamashita; Armentano e Laguna (2006) propôs um algoritmo *branch-and-bound* para o RACP. Os testes computacionais foram realizados nas mesmas instâncias de Demeulemeester (1995), mas os resultados não são conclusivos devido a diferenças entre os equipamentos utilizados nos testes.

Rodrigues e Yamashita (2010) apresentaram um algoritmo exato híbrido (chamado de MMBA) para resolver o RACP, baseado no método de Demeulemeester (1995) (chamado de MBA). A solução inicial é obtida através de uma heurística e foram desenvolvidos novos limitantes para o esquema de ramificação, diminuindo o número de subproblemas resolvidos. Os autores partem do princípio que, se um vetor de disponibilidade de recursos gera uma solução viável, qualquer outro vetor com valores iguais ou maiores também gera uma solução viável. Da mesma forma, se um vetor de recursos gerar uma solução inviável, todos os outros vetores menores ou iguais geram soluções inviáveis. Assim, todos os vetores de recursos inviáveis podem ser usados para cortar o espaço de busca de soluções.

Rodrigues e Yamashita (2010) testaram o algoritmo MMBA proposto contra o MBA de Demeulemeester (1995) em instâncias de 30 atividades e 4 tipos de recursos da PSPLIB de Kolisch e Sprecher (1997), que foram modificadas no programa ProGen de Kolisch; Sprecher e Drexler (1995). O MMBA requer menos tempo computacional nas instâncias processadas em mais de 10 segundos, enquanto que o MBA é mais rápido nas instâncias que requerem menos de 3 segundos. Nas instâncias de teste mais difíceis, eles identificaram que a dificuldade em resolver o RCPSP era a causa

para o aumento no tempo computacional. No geral, o MMBA reduz o tempo de processamento em relação ao MBA em 24%. Os autores fizeram testes que mostraram que essa redução no tempo vem dos cortes introduzidos no esquema de ramificação, combinado com a solução heurística inicial.

2.1.2 Métodos heurísticos para o RACP

As primeiras heurísticas para o RACP foram propostas por Drexl e Kimms (2001). Eles propuseram dois limitantes inferiores para o RACP, cujas heurísticas também são capazes de gerar soluções viáveis e por isso geram também limitantes superiores. Os autores apresentaram também alguns limitantes simples de serem calculados. Um limitante inferior simples pode ser calculado como o maior valor entre a maior necessidade de uma única atividade e a necessidade total do recurso distribuída igualmente ao longo da duração total do projeto (média de utilização do recurso). Um limitante superior simples é calculado considerando a quantidade de recursos necessária para executar todas as atividades ao mesmo tempo. Um outro limitante superior simples é a menor quantidade de recursos necessária para programar as atividades nos instantes de início mais cedo.

O primeiro limitante proposto é calculado através de relaxação Lagrangeana combinado com otimização de sub gradiente para determinar o valor dos multiplicadores. Dado um conjunto de multiplicadores Lagrangeanos, o subproblema restante é decomposto e pode ser resolvido em tempo (pseudo) polinomial e por isso esse método é muito rápido. O segundo limitante é obtido através de técnicas de geração de colunas. Uma coluna corresponde a uma variável de decisão que representa um programa (*schedule*). A heurística começa com uma coluna, que corresponde ao programa de início mais cedo, e a cada iteração tenta acrescentar uma coluna que, se adicionada, possa reduzir o valor da função objetivo. Se não houver mais nenhuma coluna que possa ser acrescentada, o procedimento para o valor da função objetivo obtido é um limitante inferior. O subproblema de gerar uma coluna também pode ser resolvido em tempo pseudo polinomial, porém o tempo de processamento depende fortemente do número de colunas que precisam ser geradas.

Nos testes computacionais, Drexl e Kimms (2001) utilizaram instâncias de teste com 30 atividades e 4 recursos da biblioteca PSPLIB, proposta por Kolisch e Sprecher

(1997) e gerada pelo programa ProGen de Kolisch; Sprecher e Drexl (1995), e compararam os dois limitantes inferiores propostos com o limitante inferior simples. O método de geração de coluna gera resultados melhores do que o método de relaxação Lagrangeana, porém ambos não conseguem melhorar o limitante inferior simples quando a data de entrega é maior ou igual a 1,3 vezes a data de término mais cedo do projeto. Os autores também utilizaram as instâncias de teste de Möhring (1984) com 16 atividades e 4 recursos e constataram que o método de geração de colunas gera bons limitantes inferiores, próximos da solução ótima, e tem resultados melhores que o método de relaxação Lagrangeana. Um problema do método de geração de colunas é que seu tempo de execução pode ser muito maior do que o método de relaxação Lagrangeana dependendo das características do projeto.

Yamashita; Armentano e Laguna (2006) desenvolveram uma heurística de busca dispersa (*scatter search*), chamada SSD, com atualização dinâmica do conjunto de referência, uso de memória baseada em frequência dentro do gerador de diversificação e um método de combinação de soluções baseado em *path relinking*. A heurística começa com o método gerador de diversificação para construir um conjunto de soluções melhoradas. Esse método gerador consiste em: para cada recurso, são criados subintervalos com os possíveis valores de disponibilidade de cada recurso. Uma solução é construída sorteando-se um subintervalo e depois um valor de disponibilidade para cada recurso. Aplica-se a heurística de melhoria na solução construída. Uma matriz de frequência é atualizada a cada solução gerada e a probabilidade de um subintervalo ser sorteado é inversamente proporcional a sua frequência, para aumentar a diversidade.

A heurística de melhoria é composta de duas fases: viabilidade e melhoria. Se a solução for inviável, aplica-se a fase de viabilidade. Uma solução é inviável se o *makespan* do programa gerado pela heurística desenvolvida por Tormos e Lova (2003) para resolver o RCPSP for maior do que a data de entrega. Esta heurística, chamada de RSBF, é um método híbrido *multi-pass* que combina amostragem aleatória dentro de um método de programação *backward-forward* e foi escolhida por apresentar uma boa relação entre qualidade da solução e tempo de processamento. No algoritmo de viabilidade, é identificada a atividade com a maior diferença entre o instante de início das atividades no programa inviável e o instante de início mais tarde. A quantidade disponível de recursos é adicionada da quantidade necessária para

executar essa atividade. O RSBF é um procedimento para tentar eliminar a folga de disponibilidade de um dos recursos chamado SEP são executados novamente. Se o novo programa for viável, o procedimento continua para a fase de melhoria, caso contrário a fase de viabilidade é aplicada somente mais uma vez, para não onerar o tempo de processamento do algoritmo. Na fase de melhoria, a quantidade disponível de cada recurso é diminuída em uma unidade e o RSBF e o SEP são executados novamente. O objetivo dessa fase é encontrar uma solução viável com custo menor.

O segundo passo da heurística de busca dispersa é a construção do conjunto de referência, composto por um subconjunto de referência de alta qualidade, medido pelo valor da função objetivo, e por um subconjunto de referência diverso, composto pelas soluções que maximizam a mínima distância para o conjunto de referência. As soluções do conjunto de referência são combinadas por *path relinking*, onde a solução que maximiza a mínima distância entre todas as soluções é a solução inicial. É gerado um caminho a partir da solução inicial em direção a todas as outras soluções, com o objetivo de achar uma solução melhor ou mais diversa. A melhor versão do algoritmo de busca dispersa SSD é a que possui subconjunto de referência de alta qualidade com tamanho 7, subconjunto de referência diverso com tamanho 3, atualização do conjunto de referência por qualidade e por diversificação e atualização dinâmica do conjunto de referência.

Os testes computacionais foram realizados em 2 grupos de instâncias de testes. No primeiro grupo, foram criadas instâncias de teste com 30, 60, 90 e 120 atividades e 4, 6 e 8 recursos no programa ProGen de Kolisch; Sprecher e Drexl (1995) e a performance do algoritmo de busca dispersa foi comparada com duas heurísticas multi-início (aleatório e baseado em frequência), criadas com o objetivo de avaliar se a estratégia avançada de busca dispersa gera resultados melhores do que essas simples heurísticas. No segundo grupo de testes, foram utilizadas as mesmas instâncias de Drexl e Kimms (2001), com 20 e 30 atividades e 2, 4, 6 e 8 recursos, de forma a comparar o SSD com os limitantes inferiores e superiores propostos por estes autores e também com a solução exata obtida ao aplicar o algoritmo de planos de corte de Demeulemeester (1995). Os resultados mostraram o bom desempenho da heurística SSD proposta. O SSD também foi capaz de calcular a solução ótima em todas as instâncias de teste de Möhring (1984).

Ranjbar; Kianfar e Shadrokh (2008) apresentaram as meta-heurísticas *path relinking* e algoritmo genético para o RACP. Ao contrário de outros métodos, os autores não transformam o problema em um RCPSP, mas representam o problema como uma lista de prioridade de atividades e usam um esquema de geração de programa (SGS), baseado na lógica de Burgess e Killebrew (1962), para convertê-la em um programa. O procedimento de *path relinking* começa com a geração da população inicial através do método de geração de diversificação e do procedimento para transformar uma lista de atividades viável em termos de precedência (MPFAL). O método de geração de diversificação consiste em gerar listas de atividades aleatoriamente, sendo que a probabilidade de uma atividade ser escolhida em uma posição é inversamente proporcional à sua frequência de escolha. O próximo passo é construir o conjunto de referência com as soluções de elite, tanto por qualidade quanto por diversidade. Cada par de soluções é combinado pelo método de *path relinking* e duas das novas soluções são adicionadas ao conjunto de referência, ou seja, a atualização do conjunto de referência é estática.

Já o algoritmo genético começa com a geração da população inicial, também através do método de geração de diversificação e do procedimento MPFAL. A nova geração de soluções é construída cruzando cada elemento da população inicial com um outro elemento selecionado aleatoriamente. Os dois filhos são gerados por cruzamento em dois pontos e por mutação, ambos baseados em Shadrokh e Kianfar (2007). O filho com o menor *makespan* é adicionado a população. Ao final dos cruzamentos, metade da população é excluída, sendo que a probabilidade de exclusão é proporcional ao *makespan* do indivíduo.

Os experimentos computacionais foram feitos em 144 instâncias de testes de Yamashita; Armentano e Laguna (2006) com 30, 60, 90 e 120 atividades e 4, 6 e 8 recursos. O algoritmo *path relinking* proposto apresentou desempenho melhor do que o algoritmo genético e conseguiu melhorar 7 das 144 melhores soluções de Yamashita; Armentano e Laguna (2006).

Van Peteghem e Vanhoucke (2013) desenvolveram um algoritmo de sistema imunológico artificial (AIS), baseado no sistema imunológico dos vertebrados, com uma nova função de aptidão, uma função de probabilidade para a composição de listas de capacidade de recursos e uma função de densidade de K-médias para estimular a diversidade das soluções e evitar a convergência prematura. O AIS

assemelha-se ao algoritmo genético sem a etapa de cruzamento. Assim como em Shadrokh e Kianfar (2007), um indivíduo da população é composto por uma lista de sequência de atividades e uma lista de capacidade disponível de recursos. O esquema de geração de programa (SGS) em série de Kelley Jr. (1963) é utilizado para transformar o indivíduo em um programa. O algoritmo começa com um procedimento de pré-processamento, no qual calcula-se a mínima e a máxima disponibilidade para cada tipo de recurso e uma solução heurística limitante superior, que é obtida ao aplicar o procedimento de nivelamento de recursos de Burgess e Killebrew (1962). O próximo passo é a geração da população inicial, na qual os indivíduos são gerados aleatoriamente de forma a gerar maior diversidade e demandar menor esforço computacional. Para cada indivíduo, são geradas aleatoriamente uma lista de atividades viável em termos de precedência e uma lista de capacidade de recursos viável. Aplica-se o SGS na população e calcula-se o valor de aptidão para cada indivíduo. A função de aptidão proposta consiste de duas partes: a primeira indica o custo relativo do indivíduo em relação ao menor custo, que é baseado na mínima disponibilidade para cada tipo de recurso. A segunda parte avalia a viabilidade quanto à data de entrega, através da relação entre a data de término do indivíduo e a data de entrega do projeto.

A próxima fase do algoritmo é o processo de seleção de clonagem. Os anticorpos (programas) disponíveis para proliferação são selecionados segundo uma avaliação de densidade de K-médias. Nesse método, K anticorpos são selecionados aleatoriamente, representando os centroides dos grupos. Os demais anticorpos são alocados no grupo cujo centroide é o mais próximo. Após designar todos os anticorpos, os centroides são recalculados como a média das posições dos indivíduos do grupo. O processo é repetido até os centroides estabilizarem. Em cada um dos K grupos, o anticorpo com o melhor valor de aptidão é selecionado e disponibilizado para proliferação.

Na fase de proliferação, os K anticorpos selecionados são clonados, dobrando o tamanho da população. Quanto menor o valor de aptidão do anticorpo, maior é o número de clones gerados. Depois, acontece o processo de hiper-mutação. Na lista de atividades, quanto menor o valor de aptidão, menor a taxa de hiper-mutação e menor o número de mutações. A mutação consiste em explorar a vizinhança da solução. Na lista de capacidade de recursos, a mutação gera uma nova lista, com

base em uma função de probabilidade criada pelos autores. Com essa fórmula, as disponibilidades de recurso que geram programas viáveis com relação à data de entrega do projeto possuem maior probabilidade de serem selecionados. Essa função acumula os resultados de todas as gerações. Os novos anticorpos, compostos por uma nova lista de atividades e uma nova lista de capacidade, são avaliados pelo SGS e são melhorados por um procedimento de busca local. Se a duração do programa é menor ou igual a data de entrega do programa, é aplicada a busca local de nivelamento de recursos, para tentar reduzir o custo total dos recursos. Caso contrário, é executada a busca local *forward-backward*, para diminuir o instante de término do programa. Após o processo de clonagem e mutação, os melhores anticorpos da população são mantidos e uma nova população aleatória é gerada, utilizando-se uma nova semente. As etapas anteriores são executadas novamente até atingir o critério de parada.

Van Peteghem e Vanhoucke (2013) testaram a meta-heurística proposta em instâncias de teste com 30 atividades e 4 tipos de recursos, chamadas de conjunto RACP30, criadas pelos autores utilizando o programa RanGen de Vanhoucke et al. (2008). Primeiro eles encontraram os melhores parâmetros para o algoritmo e depois testaram o desempenho dos diversos componentes propostos. A função de aptidão proposta foi comparada com a utilizada em Shadrokh e Kianfar (2007) e os resultados gerados foram melhores. Quando o método de K-médias é acrescentado, verifica-se mais uma melhoria nos resultados. Por fim, o acréscimo da busca local *forward-backward* melhora os resultados significativamente. O AIS foi comparado ao algoritmo genético de Shadrokh e Kianfar (2007) nas instâncias de teste de 30 atividades e 4 recursos da biblioteca PSPLIB de Kolisch e Sprecher (1997), gerando resultados melhores em menor tempo computacional, e ao algoritmo de otimização de ervas invasivas (IWO) de Van Peteghem e Vanhoucke apud Van Peteghem e Vanhoucke (2013) no conjunto RACP30 e nas instâncias de teste de 30, 60, 90 e 120 atividades e 4 recursos da biblioteca PSPLIB, também gerando resultados melhores.

Zhu et al. (2017) dividiram o RACP em dois subproblemas (problema de sequenciamento e problema de decisão de recursos) e propuseram uma heurística de busca multi-início iterativa (chamada de MSIS). A sequência das atividades é obtida através de um procedimento de busca iterativa e a decisão dos recursos é realizada através do procedimento de ajuste de recursos em dois estágios (TSRAP) e do

procedimento *backward* para eliminação de picos (BPEP). A representação da solução é a mesma de Shadrokh e Kianfar (2007), com uma lista de atividades e uma lista de capacidade disponível de recursos.

O primeiro passo da heurística consiste em gerar a população inicial de soluções. Calcula-se a prioridade de cada atividade de acordo com uma regra de prioridade combinada com um número gerado aleatoriamente. Em seguida, constrói-se uma solução inicial viável e aplica-se o procedimento *backward* para eliminação de picos (BPEP), no qual as atividades são atrasadas seguindo uma estratégia definida com o objetivo de reduzir o pico de utilização de recursos.

Os passos seguintes são repetidos até atingir o critério de parada. As soluções da população inicial são melhoradas pelo método de busca local. O grupo de elite é atualizado com as soluções que atendem os critérios de aceitação e a melhor solução é armazenada como a melhor solução global. Se houver atualização no grupo de elite, então aplica-se o método de *path relinking* combinado com BPEP em todos os pares de soluções e em seguida aplica-se o método de busca local nas novas soluções geradas. Novamente, atualiza-se o grupo de elite com as soluções que atendem os critérios de aceitação e também a melhor solução global. Quando o grupo de elite ou a melhor solução não forem atualizados por um dado número de gerações, então é gerada uma nova população inicial. Caso contrário, aplica-se o procedimento de geração de novos pontos (NSPG) nas soluções atuais como preparação para a próxima iteração. O procedimento NSPG é baseado no cruzamento em dois pontos de Shadrokh e Kianfar (2007) e também contém o BPEP para melhorar as soluções e diversificar a população.

O procedimento de ajuste de recursos em dois estágios (TSRAP) é executado dentro do método de busca local. No primeiro estágio, reduz-se a disponibilidade de cada recurso em uma unidade, em ordem decrescente de custo unitário, enquanto for possível gerar uma solução viável. No segundo estágio, diminui-se a disponibilidade dos recursos caros e aumenta-se a disponibilidade dos recursos baratos.

Os testes computacionais foram realizados nas instâncias de 30, 60, 90 e 120 atividades e 4 recursos da PSPLIB de Kolisch e Sprecher (1997) e em instâncias com 30 atividades e 4, 6 e 8 recursos geradas no programa RanGen. O critério de parada utilizado foi o número de programas gerados, conforme Kolisch e Hartmann (2006). Primeiramente, os autores calibraram os parâmetros da heurística proposta MSIS nas

instâncias de testes da PSPLIB. Foram testadas três regras de prioridade, com base em trabalhos anteriores desenvolvidos para o RCPSP: maior necessidade de recurso acumulada, *rank* posicional ponderado e data mais tarde de término da atividade, sendo que a regra de melhor desempenho foi a de maior necessidade de recurso acumulada.

Em seguida, os autores compararam o desempenho da heurística MSIS com a heurística de busca dispersa (SS) de Yamashita; Armentano e Laguna (2006) e com o algoritmo genético (GA) e a heurística de *path relinking* (PR) de Ranjbar; Kianfar e Shadrokh (2008). Cada instância de teste foi executada cinco vezes e o valor médio das cinco replicações é considerado o resultado obtido pelo algoritmo. O indicador de desempenho é o desvio do resultado obtido em relação ao melhor resultado obtido pelas quatro heurísticas. Nas instâncias da PSPLIB, a heurística MSIS apresentou resultados mais efetivos e mais eficientes do que a heurística SS quando o número de atividades é 120 e do que as heurísticas GA e PR quando a instância possui 60, 90 ou 120 atividades, sendo que essas diferenças são estatisticamente significativas. Os testes realizados nas instâncias RanGen mostraram resultados similares.

Já Neumann e Zimmermann (1999) apresentaram heurísticas construtivas polinomiais para problemas de nivelamento de recursos com diferentes tipos de função objetivo e com intervalos mínimo e máximo entre as atividades do projeto. Os autores apresentaram uma heurística polinomial baseada em regras de prioridade para o problema sem restrição de recursos e duas generalizações desta heurística para problemas com restrição de recursos explícita. Foram estudadas três classes de função objetivo: minimização da soma do máximo custo de recursos por período de tempo (RACP), minimização do desvio do consumo dos recursos em relação a uma dada disponibilidade e minimização da variação da utilização dos recursos ao longo do tempo. As heurísticas propostas utilizam os seguintes conceitos: custo de penalidade, que é o custo dos recursos adicionais necessários para programar o início de uma atividade em certo instante de tempo; atividades quase críticas, que são as que possuem folga total maior que zero e menor que a sua duração; conjunto ativo básico, que são as atividades obrigatoriamente em processamento em certo instante de tempo de modo que o projeto atenda as durações mínima e máxima do projeto; matriz de utilização de recursos básica, que considera a utilização de recursos ao longo do tempo das atividades do conjunto ativo básico; e conjunto de decisão, que é

composto pelos possíveis instantes de início da atividade (entre os instantes mais cedo e mais tarde) e que também são instantes em que há uma mudança de patamar (descontinuidade) na matriz de utilização de recursos.

A heurística polinomial baseada em regras de prioridade para o problema sem restrição de recursos pode ser utilizada para as três classes de função objetivo e para três diferentes matrizes de utilização de recursos (considerando somente as atividades já programadas, adicionando as atividades quase críticas e considerando todas as atividades do projeto; sendo que as três não consideram a atividade que está sendo avaliada). Para escolher a próxima atividade a ser programada, são consideradas todas as atividades não programadas, mesmo aquelas cujas atividades precedentes ainda não foram programadas.

O procedimento começa com a programação de todas as atividades críticas. Em seguida, seleciona-se uma atividade não programada com base na regra de prioridade e na regra de desempate, que podem ser: mínimo paralelismo (número de atividades que podem estar em processamento ao mesmo tempo em que a atividade em questão), mínima folga total, mínimo instante mais tarde de início, maior demanda total de recursos e mínima ordem de processamento. Em seguida, calcula-se a matriz de utilização de recursos, o conjunto de decisão, os valores da função de custo de penalidade e o instante de início que minimiza o custo de penalidade. A atividade selecionada é programada nesse instante e os instantes mais cedo e mais tarde e a folga total das atividades não programadas são recalculados. O conjunto atualizado de atividades críticas é programado e seleciona-se a próxima atividade ainda não programada. O procedimento é repetido até que todas as atividades sejam programadas.

Nos testes computacionais, foram testadas quatro regras de prioridade em dois conjuntos de instâncias de testes gerados no programa ProGen/max de Schwindt apud Neumann e Zimmermann (1999). O primeiro conjunto *A* é composto por projetos com 10, 15 e 20 atividades e 1, 3 e 5 recursos e o segundo conjunto *B* é composto por instâncias com 100, 200 e 500 atividades e 1, 3 e 5 recursos. Para o conjunto *A*, os autores calcularam o desvio do valor da função objetivo em relação ao menor valor encontrado por um procedimento *branch-and-bound* em desenvolvimento. Considerando a melhor solução das quatro regras de prioridade, as heurísticas geraram soluções com valores 6% a 25% maiores que o mínimo para o RACP. Para

o conjunto B , as quatro regras de prioridade foram comparadas à heurística de Brinkmann e Neumann apud Neumann e Zimmermann (1999), por ser a única heurística encontrada na literatura para o problema sem restrição de recursos. O melhor resultado para o RACP foi obtido pela regra de prioridade “mínima ordem de processamento” combinada com a regra de desempate “mínimo instante mais tarde de início”. Os testes também mostraram que as heurísticas propostas geralmente obtêm bons programas viáveis em um tempo computacional razoável.

Neumann e Zimmermann (2000) abordaram os problemas de nivelamento de recursos e de valor presente líquido com restrições temporais generalizadas e com e sem restrição de recursos. Eles desenvolveram um procedimento *branch-and-bound*, um *branch-and-bound* truncado, heurísticas construtivas e uma heurística de melhoria baseada na busca tabu para resolver esses problemas. Assim como em Neumann e Zimmermann (1999), foram estudados três tipos de função objetivo para o problema de nivelamento de recursos, dentre eles, o problema de minimização da soma do máximo custo de recursos por período de tempo (RACP).

No procedimento *branch-and-bound* para o problema sem restrição de recursos, a ideia básica é reduzir o conjunto de soluções viáveis programando atividades sucessivamente. Cada nó corresponde a um programa parcial. Os filhos de um nó são obtidos ao programar a atividade com a menor folga total e o menor índice nas suas possíveis datas de início (entre a data mais cedo e a data mais tarde). O próximo nó a ser analisado é escolhido através da combinação de busca em profundidade primeiro e de menor limitante inferior, ou seja, após cada ramificação, analisa-se o nó recém gerado com o menor limitante inferior. Quando um nó é sondado por limitante, todos os seus irmãos também são sondados.

Para resolver problemas maiores, foi criado o *branch-and-bound* truncado baseado em técnicas de *beam search* filtrado, no qual o número de filhos de um nó é reduzido a um limite chamado de largura da busca B . Somente os B nós mais promissores (com os menores valores de limitante inferior) são selecionados para continuar a ramificação e os demais são descartados em definitivo.

As heurísticas construtivas aplicam-se somente aos problemas com restrição de recursos, o que não é o caso do RACP. Já na heurística de melhoria baseada na busca tabu, primeiro são encontrados os picos de utilização de cada recurso e então altera-se a data de início de uma atividade com o objetivo de reduzir o pico de um ou

mais recursos. Para evitar que o pico de um recurso recém reduzido seja aumentado novamente, todas as soluções vizinhas com pico maior recebem o status de tabu. O critério de aspiração cancela o status de tabu caso a solução represente uma nova melhor solução. Caso não seja encontrada uma nova melhor solução após 20 iterações, são executados métodos de intensificação e diversificação, para melhorar o processo de busca.

Os testes foram realizados nos mesmos dois conjuntos de instâncias de testes de Neumann e Zimmermann (1999). As instâncias foram geradas no programa ProGen/max de Schwindt apud Neumann e Zimmermann (2000) e consistem de: conjunto *A*, composto por projetos com 10, 15 e 20 atividades e 1, 3 e 5 recursos; e conjunto *B*, composto por instâncias com 100, 200 e 500 atividades e 1, 3 e 5 recursos. Para o problema sem restrição de recursos, que inclui o RACP, a heurística de melhoria baseada na busca tabu apresenta os melhores resultados, com desvio de até 9% do melhor resultado encontrado (até 3% para o RACP). Já para pequenas instâncias do problema com restrição de recursos (até 100 atividades), os procedimentos *branch-and-bound* e *branch-and-bound* truncado são recomendados, enquanto que para instâncias maiores, é melhor utilizar a heurística Algoritmo Reparador, que repara um programa que é viável no tempo, mas não em recursos.

Outros trabalhos realizados sobre o RACP englobam o de Jędrzejowicz e Ratajczak-Ropel (2012), que propuseram uma abordagem baseada no conceito de times assíncronos (*A-Team*), um tipo de sistemas de múltiplos agentes. A implementação é realizada no mediador (*middleware*) E-JABAT, que gera soluções para problemas de otimização combinatória usando um conjunto de agentes de otimização, cada um representando um algoritmo de melhoria. Os testes computacionais foram realizados nas mesmas instâncias de Yamashita; Armentano e Laguna (2006) e os resultados foram comparáveis.

Koulinas e Anagnostopoulos (2013) desenvolveram uma hiper-heurística baseada na busca tabu (THH). A hiper-heurística é uma nova abordagem multi-nível para otimização de problemas, na qual uma camada superior de heurística gerencia outros métodos de solução, ao invés de atuar diretamente nas soluções. A THH controla um conjunto de heurísticas de baixo nível, que alteram a prioridade das atividades dentro do Microsoft Project e modificam o programa gerado. As heurísticas mais eficientes são executadas primeiro. A THH também é composta por uma busca tabu, que proíbe

a seleção de heurísticas com desempenho recente ruim. Os autores testaram o algoritmo proposto em um projeto exemplo com 11 atividades.

Afshar-Nadjafi (2014a) estudou o RACP com data de contratação e de liberação para os recursos e com custo de *setup* de recrutamento dependente do tempo. O autor propôs uma formulação de programação inteira mista e uma heurística baseada no GRASP (*Greedy Randomized Adaptive Search Procedure*) para resolvê-lo, sem transformar o problema em RCPSP. Os testes computacionais foram realizados em instâncias de testes com até 90 atividades e com até 4 recursos criadas no programa ProGen de Kolisch; Sprecher e Drexl (1995). A heurística foi executada dez vezes e foi comparada à solução obtida pelo programa Lingo, gerando boas soluções em um tempo computacional baixo.

Qi et al. (2014) defenderam que transformar a resolução do RACP em uma resolução iterativa de vários RCPSPs é um método indireto, complicado e ineficiente. Então os autores apresentaram dois métodos mais eficientes de resolução do RACP, usando a data de início de cada atividade para codificar um programa. O primeiro método é uma nova heurística para procurar a melhor solução de forma eficiente. O segundo método é chamado de pseudo otimização por enxame de partículas (PSO - *Particle Swarm Optimization*), que combina PSO com o procedimento de *path relinking*.

Meng et al. (2016) apresentaram um algoritmo de busca dispersa (*scatter search*) combinado com busca tabu para resolver o RACP, chamado de TSS. Esse algoritmo segue o método direto de resolução de Qi et al. (2014), onde a solução é representada pela data de início das atividades. A data de início de uma atividade está no intervalo entre a data mais cedo e a data mais tarde de início, obtidas pelo CPM considerando a data de entrega do projeto. A vizinhança da busca tabu é construída selecionando aleatoriamente algumas atividades e alterando sua data de início, dentro do seu intervalo de possibilidades. Os testes computacionais foram realizados nas instâncias com 30, 60, 90 e 120 atividades e 4 recursos da PSPLIB, gerados no programa ProGen de Kolisch; Sprecher e Drexl (1995). Primeiramente foram realizados testes para calibrar os parâmetros do algoritmo. Em seguida, o desempenho do TSS foi comparado ao algoritmo de busca dispersa de Yamashita; Armentano e Laguna (2006). O TSS gerou soluções melhores, porém seu tempo de processamento foi maior.

2.2 PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS E COM MÚLTIPLOS MODOS DE EXECUÇÃO (MMRACP)

2.2.1 Métodos exatos para o MMRACP

Yamashita e Morabito (2009) combinaram o algoritmo exato de Demeulemeester (1995), construído para o RACP, e o algoritmo exato de Sprecher e Drexl (1998), desenvolvido para o RCPSP com múltiplos modos de execução (MMRCPSP), para obter soluções exatas para o MMRACP, gerando também curvas com o *tradeoff* entre data de término do projeto e custo dos recursos. Devido à dificuldade de resolver este problema, o método proposto consegue resolver apenas problemas de tamanho moderado. Os autores fizeram duas modificações no algoritmo de Demeulemeester (1995): resolveram uma sequência de MMRCPPSPs ao invés de RCPSPs e adicionaram um procedimento para gerar as curvas de *tradeoff*.

O primeiro passo do algoritmo é determinar as datas de entrega do projeto mínima e máxima. No passo seguinte, o valor atual da data de entrega recebe a data de entrega máxima e calcula-se o limitante inferior para a capacidade disponível de cada recurso. Então um problema MMRCPPSP é resolvido com o algoritmo *branch and bound* de Sprecher e Drexl (1998). Neste algoritmo, é gerada uma árvore de busca, os nós são programas parciais e a cada ramificação, escolhe-se a próxima atividade a ser programada e seu modo de execução. Se for encontrada uma solução viável para o MMRCPPSP considerando a data de entrega atual do projeto, então a solução ótima foi encontrada. Caso contrário, são criados os conjuntos de pontos eficientes e o MMRCPPSP é resolvido para o conjunto de pontos eficientes mais barato, até que uma solução viável seja encontrada. Para gerar as curvas com o *tradeoff* entre data de término do projeto e custo dos recursos, diminui-se o valor da data de entrega atual e repete-se todo o procedimento, encontrando uma solução ótima para o novo valor de data de entrega.

Foram realizados testes computacionais em instâncias de teste geradas no programa ProGen, de Kolisch e Sprecher (1997), com 15 atividades, 4 tipos de recursos, 3 modos de execução e considerando funções objetivo linear e não linear, e os

resultados foram comparados com a resolução do programa CPLEX utilizando a linguagem GAMS. O algoritmo proposto é mais rápido, resolvendo mais instâncias do que o GAMS/CPLEX dentro do tempo limite de 3.600 segundos. O algoritmo é computacionalmente viável para situações em que é razoável considerar a quantidade disponível de recursos como números inteiros e em quantidade relativamente pequena, como dezenas de trabalhadores.

Outro trabalho que desenvolveu um método exato para o MMRACP é o de Sabzehparvar; Seyed-Hosseini e Nouri (2008), que apresentaram um modelo exato para o MMRACP com relações de precedência generalizadas. O modelo foi inspirado nos problemas de empacotamento e foi resolvido no programa Lingo.

2.2.2 Métodos heurísticos para o MMRACP

Hsu e Kim (2005) desenvolveram a primeira heurística para o MMRACP, composta por regras de prioridade e liberação aplicadas dentro de um esquema de geração de programa em série. O algoritmo começa com a inicialização do limite de investimento em recursos com um valor baixo. O próximo passo é a construção de um programa completo através do esquema de geração de programa em série. Neste esquema, o programa é construído iterativamente, atividade por atividade e a regra de prioridade é utilizada para determinar qual será a próxima atividade a ser programada. Essa única função de prioridade combina as funções de prioridade de folga e de investimento transformadas, avaliando todas as alternativas de atividades, modos e tempos das atividades ainda não programadas e cujas predecessoras já estão programadas, e possibilita a seleção simultânea da atividade a ser programada, seu modo e seu instante de início. A função de prioridade de folga transformada de uma atividade é um valor normalizado entre zero e um e relaciona os valores de instante de término da atividade se iniciada em um instante de tempo em um modo (FT), instante de término limite (TFT), que é o tempo mais tarde calculado pelo método do caminho crítico (CPM) considerando o modo mais longo para todas as atividades, e o instante de término mais tarde (LFT), calculado pelo CPM considerando o modo mais curto para todas as atividades. Se o FT de uma atividade for maior do que seu LFT, o projeto não será concluído dentro da data de término e o valor da função de prioridade de folga transformada é infinito. Se o FT de uma atividade for menor ou igual ao seu

TFT, então as atividades sucessoras podem ser executadas em qualquer modo e o valor da função é zero. Por fim, se o FT de uma atividade for maior do que seu TFT, as atividades sucessoras só podem ser realizadas em alguns modos e o valor da função é positivo e maior do que zero. Assim, um valor alto da função de prioridade de folga transformada de uma atividade significa que é mais difícil programar suas atividades sucessoras. Já a função de prioridade de investimento transformada avalia o investimento adicional em recursos ao programar uma atividade e também é um valor normalizado entre zero e um. Se o programa parcial for inviável, o limite de investimento em recursos é aumentado até um programa viável ser encontrado.

Nos testes computacionais, o desempenho do algoritmo proposto foi comparado ao de outras quinze combinações de duas regras de prioridade de Kolisch apud Hsu e Kim (2005) e de Neumann e Zimmermann (1999), sendo uma regra para selecionar a atividade e outra regra para selecionar o modo e o instante de início. As instâncias de teste são da biblioteca PSPLIB de Kolisch e Sprecher (1997) para o RCPSP e possuem entre 12 a 30 atividades e 4 tipos de recursos. Na média, o algoritmo proposto de uma regra é melhor do que todas as combinações de duas regras, gerando soluções com o valor da função objetivo de 25% a 50% menor. O melhor conjunto de duas regras é a combinação de mínima folga e mínimo investimento extra. Os autores também fizeram testes para determinar a melhor forma de combinar as duas funções de prioridade.

Outros trabalhos desenvolvidos sobre o MMRACP incluem Coughlan; Lübbecke e Schulz (2010) e Coughlan; Lübbecke e Schulz (2015), que estudaram o MMRACP com períodos de disponibilidade (ou turnos de trabalho), chamado também de problema de programação de *turnaround*. Cada recurso está disponível para uso somente em um conjunto de períodos de disponibilidade, também chamado de turnos. Devido a granularidade no planejamento, cada atividade necessita de apenas um recurso para sua execução. Os autores apresentaram uma formulação de programação inteira mista, um algoritmo *branch-and-price* em Coughlan; Lübbecke e Schulz (2010) e um algoritmo *branch-and-price-and-cut* em Coughlan; Lübbecke e Schulz (2015). Os testes computacionais foram realizados em instâncias de testes com 30 e 50 atividades, 2 recursos, 3 modos de execução e 5 tipos de calendário, adaptadas da PSPLIB de Kolisch e Sprecher (1997), e os resultados do algoritmo foram comparados aos obtidos pelo CPLEX com um tempo limite de 1800 segundos.

Os algoritmos propostos foram capazes de calcular a solução ótima para problemas com até 50 atividades e apresentaram desempenho melhor do que o CPLEX, principalmente nas instâncias maiores.

Chen; Li e Cai (2012) desenvolveram um esquema de enumeração da árvore de precedência e uma heurística de três passos (tipos de regras) para seleção da atividade, seleção do modo de execução e decisão da data de início. Além disso, as combinações dessas regras são comparadas e as probabilidades de escolha são determinadas com base em um método estocástico de probabilidade baseado em arrependimento (RBPS). Os testes computacionais realizados em instâncias de testes adaptadas de Kolisch e Sprecher (1997) com 10 e 20 atividades mostraram que o método RBPS é superior ao de Hsu e Kim (2005), que por sua vez é melhor do que todas as combinações das regras de três passos testadas.

Afshar-Nadjafi (2014) estudou o MMRACP com data de contratação e de liberação para os recursos e mostrou uma formulação de programação inteira mista e um algoritmo de *simulated annealing*, sem transformar o problema em RCPSP. As instâncias de teste foram criadas no programa ProGen, de Drexler et al. (2000) e os resultados foram comparados com os obtidos pelo programa Lingo. O método proposto obteve rapidamente soluções com pequeno desvio em relação ao ótimo.

Afshar-Nadjafi e Arani (2014) propuseram uma formulação de programação inteira mista e um algoritmo genético com busca local baseada em conhecimento para o MMRACP com preempção, recursos renováveis e não renováveis e custos de adiantamento e de atraso para cada atividade. Os autores criaram instâncias de teste no programa ProGen, de Drexler et al. (2000) e compararam os resultados com a solução ótima obtida no CPLEX, para projetos com 20 atividades, e com a melhor solução gerada aleatoriamente, para problemas com até 80 atividades, obtendo bons resultados em um tempo computacional aceitável.

Colak e Azizoglu (2014) formularam o MMRACP com um único tipo de recurso como um modelo de programação inteira linear mista. Problemas pequenos (até 20 atividades) foram resolvidos otimamente no programa CPLEX com a linguagem GAMS. Para problemas maiores, foram desenvolvidos uma heurística de duas fases (construção e melhoria) e quatro limitantes inferiores. Os testes mostraram que os limitantes inferiores geram valores próximos da solução ótima e que a heurística possui desempenho melhor do que a heurística de Hsu e Kim (2005).

Qi et al. (2014b) estudaram o MMRACP com recursos renováveis e não renováveis e propuseram um método de otimização por enxame de partículas modificado (MPSO) combinado com um procedimento de *path relinking* e desenvolveram uma heurística para melhorar o valor da solução. Os testes computacionais realizados em instâncias de testes da PSPLIB com até 30 atividades, 2 recursos renováveis, 2 recursos não renováveis e 3 modos de execução mostraram que o MPSO tem desempenho melhor do que o algoritmo de otimização por enxame de partículas (PSO) e do que o algoritmo genético (GA).

Qi et al. (2015) apresentaram um esquema de geração de programa chamado FAL (*forward activity list*) e uma heurística de otimização por enxame de partículas modificada (MPSO), que combina as vantagens da otimização por enxame de partículas e da busca dispersa. Os testes realizados nas instâncias da PSPLIB de Kolisch e Sprecher (1997) mostraram que o método proposto, combinando MPSO e FAL, tem melhor desempenho do que o algoritmo genético de Ranjbar; Kianfar e Shadrokh (2008).

Javanmard et al. (2017) abordaram um novo problema chamado de problema de investimento em recursos com multi-habilidades e com preempção (PMSRIP - *Preemptive Multi-Skilled Resource Investment Problem*), no qual é necessário um conjunto de habilidades para executar uma atividade. Cada recurso humano possui multi-habilidades e cada nível de habilidade resulta em um tempo diferente para execução da atividade. Os autores desenvolveram duas formulações matemáticas de programação inteira mista, um algoritmo genético e um algoritmo baseado em otimização por enxame de partículas para resolver esse problema. Na primeira formulação, o nível de cada habilidade definido para cada atividade permanece inalterado durante o processamento da atividade, enquanto que na segunda formulação, o nível da habilidade pode ser alterado para outro nível após a preempção. Os dois modelos propostos e os dois algoritmos foram testados em problemas pequenos (até 30 atividades), médios (50 e 60 atividades) e grandes (80 e 90 atividades) da PSPLIB ou gerados aleatoriamente no programa RanGen. Nos problemas pequenos, os algoritmos propostos foram comparados à solução ótima gerada no programa GAMS. Nos três tamanhos de problemas, o segundo modelo matemático e o algoritmo genético apresentaram os melhores resultados.

2.3 PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM CUSTO DE DISPONIBILIDADE DE RECURSOS E CUSTO DE PENALIDADE POR ATRASO NO TÉRMINO DO PROJETO (RACPT)

Shadrokh e Kianfar (2007) desenvolveram um algoritmo genético para o RACP com custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness* ou RIPT - *Resource Investment project scheduling Problem, Tardiness permitted with penalty*). O custo por atraso é constante para cada unidade de tempo de atraso em relação a data de término. Um indivíduo da população é composto por uma lista de sequência de atividades, que determina a sequência na qual as atividades são programadas, e uma lista de capacidade disponível de recursos. Os esquemas de geração de programa em série (SSS) e em paralelo (PSS) são usados para transformar o indivíduo em um programa.

A primeira geração de indivíduos (cromossomos) é criada aleatoriamente e calcula-se o valor da função de inaptidão, que é igual ao custo total do programa. Cada nova geração é construída utilizando quatro operadores: cruzamento, mutação, imigração e busca local. No operador de cruzamento, são criados aleatoriamente pares de pais e em cada par, o cruzamento pode ocorrer com uma probabilidade. Caso ocorra o cruzamento, são produzidos dois filhos. Dependendo do valor de inaptidão e de um valor aleatório, ou o filho vai para busca local com uma probabilidade, vai para a próxima geração e o pai é excluído, ou o filho é excluído e o pai é considerado para mutação com uma probabilidade e vai para a busca local com outra probabilidade. Se o cruzamento não acontecer, cada indivíduo do par pode sofrer mutação de acordo com uma probabilidade e depois pode ocorrer a busca local dependendo de outra probabilidade. A última etapa da criação de uma nova geração é a imigração, na qual o cromossomo imigrante é gerado aleatoriamente. Um indivíduo da população atual é selecionado aleatoriamente. Novamente, dependendo do valor de inaptidão e de um valor aleatório, ou o imigrante substitui o indivíduo na população ou é descartado.

Os autores criaram instâncias de teste utilizando o programa ProGen de Kolisch; Sprecher e Drexel (1995) com 20 atividades e 4 recursos, calcularam a solução ótima usando o procedimento de enumeração de disponibilidade de recursos e o *branch-and-bound* de Demeulemeester e Herroelen (1992) e ajustaram os vários parâmetros

do algoritmo genético para o RACPT e para o RACP. Para o RACPT, o custo por atraso é igual a $1/8$ da soma dos custos unitários dos recursos

O desempenho do algoritmo com os parâmetros ajustados foi testado em instâncias de 10 e 14 atividades e 4 recursos, também criadas no ProGen, gerando resultados satisfatórios para o RACP e o RACPT. A heurística proposta também foi testada nas mesmas instâncias com 30 atividades e 4 recursos de Drexler e Kimms (2001), para as quais conseguiu gerar limitantes superiores melhores para o RACP. Para avaliar a eficiência do algoritmo em instâncias maiores, os autores também criaram instâncias de teste com 60, 90 e 120 atividades e 4 recursos e compararam a melhor solução obtida na população inicial com a melhor solução obtida após 4 segundos de execução do algoritmo, mas não mencionaram quais foram os resultados obtidos. O efeito somente da busca local foi avaliado em instâncias com 120, 150 e 200 atividades e 4 recursos e foi constatado que o resultado melhorou de 7% a 8%. O algoritmo proposto também foi testado nas instâncias de 16 atividades e 4 recursos de Möhring (1984), que foram consideradas muito simples.

Van Peteghem e Vanhoucke (2015) desenvolveram um algoritmo de otimização de ervas invasivas (IWO) para resolver o RACP e o RACPT. O algoritmo IWO é uma nova estratégia de busca baseada no comportamento natural das ervas invasivas em colonizar e achar um local adequado para crescimento e reprodução. As ervas são capazes de adaptar-se ao ambiente, mudando seu comportamento e tornando-se melhores e mais fortes, devido à sua biologia e ecologia. No algoritmo, cada indivíduo da colônia de ervas é composto por uma lista de atividades e uma lista de capacidade de recursos. O indivíduo é transformado em um programa através do esquema de geração de programa (SGS) em série de Kelley Jr. (1963).

O procedimento começa com uma etapa de pré-processamento, que é a mesma descrita em Van Peteghem e Vanhoucke (2013), e cujo objetivo é reduzir o espaço de busca da lista de capacidade de recursos, através do cálculo da disponibilidade mínima e máxima de cada recurso e de uma melhor solução inicial. A próxima etapa consiste em gerar a população inicial aleatoriamente, para garantir a diversificação de indivíduos. Depois, vem a etapa da reprodução, na qual o número de sementes de cada indivíduo é calculado com base no seu valor de aptidão. Dessa forma, mesmo indivíduos inviáveis possuem chance de sobreviver e gerar novas sementes, similar ao mecanismo que ocorre na natureza. A função de aptidão para o RACP é a mesma

de Van Peteghem e Vanhoucke (2013) e para o RACPT, é a mesma de Shadrokh e Kianfar (2007).

A próxima etapa é a dispersão espacial, na qual as sementes geradas são distribuídas aleatoriamente pelo espaço de busca. Para cada semente, são executadas duas fases: um procedimento de mutação é aplicado na lista de atividades e depois há a geração de uma nova lista de capacidade de recursos. Cada nova semente é avaliada usando o esquema de geração de programa em série e então aplica-se um procedimento de busca local na solução obtida. Na busca local, se o *makespan* for menor ou igual a data de entrega, é aplicada a busca local de nivelamento de recursos, caso contrário, é aplicada uma busca local *forward-backward*. Por fim, na etapa de exclusão competitiva, todas as ervas (pais e filhos) são ordenadas pelo valor de aptidão e somente as melhores são mantidas, assim como as ervas mais diversas.

Os testes computacionais foram realizados em instâncias com 30 atividades e 4 recursos geradas no programa RanGen de Vanhoucke et al. (2008), chamadas de RACP30, e nas instâncias de 30, 60, 90 e 120 atividades e 4 recursos da biblioteca PSPLIB de Kolisch e Sprecher (1997). Para o RACPT, o custo por atraso é igual a um percentual da soma dos custos unitários dos recursos, sendo o percentual igual a: 0; 12,5; 25; 37,5; 50; 62,5; 75; 87,5; 100 e um valor grande M . O critério de parada é o número de programas gerados (neste caso, igual a 5.000), seguindo a recomendação de Kolisch e Hartmann (2006). Para o RACP nas instâncias RACP30, o IWO consegue melhorar o limitante superior calculado seguindo a fórmula de Drexl e Kimms (2001). Além disso, quando os custos unitários dos recursos são iguais a 1, o IWO tem desempenho melhor do que o algoritmo de sistema imunológico artificial (AIS) de Van Peteghem e Vanhoucke (2013). Quando os custos unitários dos recursos variam entre 1 e 5, o AIS tem desempenho melhor do que o IWO. Nas instâncias PSPLIB, o IWO tem desempenho melhor do que o AIS e do que o algoritmo genético de Shadrokh e Kianfar (2007). Já para o RACPT, nas instâncias RACP30, quanto menor o valor do custo por atraso, menor o valor da função objetivo em relação ao RACP e maior o percentual de vezes que a data de término do projeto é maior do que a data de entrega.

3 MODELAGEM MATEMÁTICA PARA O PROBLEMA

3.1 MÉTODO OTIMIZANTE

O modelo matemático utilizado neste trabalho segue o proposto por Santoro (2016) para o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*), baseado nos conceitos de *strip packing* e com restrições disjuntivas. O modelo matemático apresentado em Shadrokh e Kianfar (2007) (vide item 1.2) utiliza as variáveis de decisão binárias x_{it} ($i = 0, \dots, n + 1, t = 0, \dots, Mx$), que determinam se a atividade i começa a ser executada no instante t ($x_{it} = 1$). Caso contrário o valor de x_{it} é igual a zero. Então a data de início de uma atividade i é definida por: $S_i = \sum_{t=0}^{Mx} tx_{it}$. Assim, este modelo matemático considera o tempo como uma variável discreta. No entanto, para problemas grandes ou para problemas reais, onde o tempo é contínuo, o número de variáveis de decisão x_{it} torna-se muito grande, inviabilizando sua implementação. O modelo apresentado em Van Peteghem e Vanhoucke (2015) é apenas um modelo conceitual, que também não foi implementado pelos autores.

Um projeto é caracterizado por um conjunto de n atividades executadas sem interrupção, que possuem um único modo de execução, duração d_i determinística e são sujeitas a um conjunto de relações de precedência $(i, j) \in H$ do tipo fim – início sem intervalo, representadas em uma rede acíclica com atividades nos nós. Cada atividade i ($i = 1, \dots, n$) necessita de r_{ik} unidades do recurso renovável de tipo k ($k = 1, \dots, m$) para ser executada e essa quantidade r_{ik} necessária de cada recurso é constante ao longo da execução de cada atividade. Cada recurso tem disponibilidade a_k constante ao longo do projeto e uma função de custo linear e não decrescente $c_k(a_k)$. Há um custo C_T constante para cada unidade de tempo de atraso do término do projeto em relação à data de entrega D . O problema consiste em: qual a programação das atividades que gera o menor custo total do projeto e que respeite as relações de precedência? O custo total é igual à soma do custo de disponibilidade dos recursos $c_k(a_k)$ e do custo de penalidade por atraso $C_T(T)$.

O problema é formulado como um problema de programação linear inteira mista. Ao considerar o RACPT como um problema de *strip packing*, considera-se que a duração das atividades corresponde ao comprimento do item (coordenada horizontal) e a necessidade de recursos é equivalente à largura do item (coordenada vertical). Os parâmetros, as variáveis de decisão e o modelo matemático do RACPT são apresentados abaixo:

Parâmetros:

D = data de entrega do projeto

d_i = duração da atividade i

r_{ik} = quantidade necessária de recurso k para executar a atividade i

c_k = custo unitário do recurso k

C_T = custo unitário do atraso

$Mx = \sum_{i=1}^n d_i$, = valor grande necessário para as restrições disjuntivas

$NmMax_k = \sum_{i=1}^n r_{ik}$ = valor grande necessário para as restrições disjuntivas

Variáveis:

F_i = data de término da atividade i

PCT = data de término do projeto

$z1_{ij} = 1$ se a atividade i precede a atividade j ; 0 caso contrário ($i < j$)

$z2_{ij} = 1$ se a atividade j precede a atividade i ; 0 caso contrário ($i < j$)

$z3_{ij} = 1$ se a atividade i está abaixo da atividade j ; 0 caso contrário ($i < j$)

$z4_{ij} = 1$ se a atividade i está acima da atividade j ; 0 caso contrário ($i < j$)

a_k = quantidade disponível do recurso k

y_{ik} = coordenada vertical da atividade i no recurso k

T = atraso do término do projeto em relação à data de entrega D

Modelo matemático:

$$\min \sum_{k=1}^m c_k a_k + C_T T \quad (3.1)$$

sujeito a:

$$F_i \geq d_i, \quad i = 1, \dots, n \quad (3.2)$$

$$y_{ik} \geq r_{ik}, \quad i = 1, \dots, n; k = 1, \dots, m \quad (3.3)$$

$$y_{ik} \leq a_k, \quad i = 1, \dots, n; k = 1, \dots, m \quad (3.4)$$

$$F_j - F_i - d_j + (1 - z_{1ij})Mx \geq 0, \quad i < j = 1, \dots, n \quad (3.5)$$

$$F_i - F_j - d_i + (1 - z_{2ij})Mx \geq 0, \quad i < j = 1, \dots, n \quad (3.6)$$

$$y_{jk} - y_{ik} - r_{jk} + (1 - z_{3ij})NmMax_k \geq 0, \quad i \neq j = 1, \dots, n; k = 1, \dots, m \quad (3.7)$$

$$y_{ik} - y_{jk} - r_{ik} + (1 - z_{4ij})NmMax_k \geq 0, \quad i \neq j = 1, \dots, n; k = 1, \dots, m \quad (3.8)$$

$$z_{1ij} + z_{2ij} + z_{3ij} + z_{4ij} = 1, \quad i < j = 1, \dots, n \quad (3.9)$$

$$z_{1ij} = 1, \quad \forall (i, j) \in H \quad (3.10)$$

$$PCT \geq F_i, \quad i = 1, \dots, n \quad (3.11)$$

$$T \geq PCT - D \quad (3.12)$$

$$F_i, PCT, a_k, y_{ik}, T \in \mathbb{R}^+ \quad i = 1, \dots, n \quad (3.13)$$

$$z_{1ij}, z_{2ij}, z_{3ij}, z_{4ij} \in \{0,1\}, \quad i \neq j = 1, \dots, n \quad (3.14)$$

A função objetivo (3.1) visa minimizar o custo total de disponibilidade de todos os recursos mais a penalidade por atraso.

As restrições (3.2) garantem que a data de término da atividade seja sempre maior ou igual a sua duração, assegurando que todas as atividades sejam programadas.

Os conjuntos de restrições (3.3) e (3.4) garantem que a coordenada vertical de cada atividade i em cada recurso k seja sempre maior ou igual a sua necessidade e sempre menor ou igual a capacidade disponível do recurso.

As restrições (3.5), (3.6), (3.7) e (3.8) funcionam em conjunto: se $z_{1ij} = 1$, então a atividade i precede a atividade j e somente a restrição (3.5) é ativada, garantindo que a data de término da atividade j seja pelo menos igual a data de término da atividade i mais a duração da atividade j . Caso contrário, quando $z_{2ij} = 1$, a atividade j precede a atividade i e somente a restrição (3.6) é ativada. Se $z_{3ij} = 1$, então a atividade i está abaixo da atividade j e apenas a restrição (3.7) é ativada, garantindo que a coordenada vertical da atividade j seja pelo menos igual a coordenada vertical da atividade i mais a necessidade de recursos da atividade j , para todos os m recursos. Por fim, quando $z_{4ij} = 1$, a atividade i está acima da atividade j e a única restrição ativada é a (3.8).

As restrições (3.9) garantem que não ocorra sobreposição na programação para cada par de atividades i e j . As restrições (3.10) levam em consideração as relações de

precedência entre os pares de atividades (i, j) , onde j é uma atividade imediatamente sucessora de i .

Já as restrições (3.11) asseguram que a data de término do projeto seja maior ou igual a data de término de todas as atividades.

A restrição (3.12) representa o cálculo do atraso do projeto $T = \max\{0, PCT - D\}$. Se o projeto for concluído antes da data de entrega D , o atraso T é igual a zero, caso contrário, o atraso T é igual a diferença entre a data de término do projeto PCT e a data de entrega D .

Por fim, os conjuntos de restrições (3.13) e (3.14) definem o domínio das variáveis.

3.2 MÉTODOS HEURÍSTICOS

3.2.1 Heurística construtiva determinística

O primeiro método proposto é uma heurística construtiva determinística. A ideia é utilizar uma regra de prioridade combinando a flexibilidade das atividades e a flexibilidade dos recursos para selecionar a próxima atividade a ser programada.

A flexibilidade da atividade i ($FA_i, i = 1, \dots, n$) é determinada com base em regras de prioridade. As seguintes regras, mencionadas em Zhu et al. (2017) como as de melhor desempenho para o RCPSP, serão testadas para determinar qual o melhor cálculo de FA_i .

- Maior necessidade de recurso acumulada (GCRR – *Greatest Cumulative Resource Requirement*): maior prioridade para a atividade com a maior necessidade de recursos acumulada para completar a atividade e todas as suas atividades sucessoras (YANG, 1998).
- Rank posicional ponderado (RPW - *Rank Positional Weight*): maior prioridade para a atividade com a maior soma das durações da atividade e de todas as suas atividades sucessoras (YANG, 1998).
- Mínimo instante mais tarde de início (MLFT – *Minimum Latest Finish Time*): maior prioridade para a atividade com a menor data fim mais tarde, calculada pelo CPM (ZHU et al., 2017).

- Menor folga total (MAS – *Minimum Activity Slack*): maior prioridade para a atividade com a menor folga total, calculada pelo CPM (YANG, 1998).

As regras GCRR e RPW são regras estáticas, ou seja, são calculadas uma única vez. Já as regras MLFT e MAS são dinâmicas, sendo recalculadas a cada instante de decisão. Quanto maior a prioridade da atividade conforme a regra escolhida, menor o valor de FA_i e menos flexível é a atividade i .

A flexibilidade do recurso k ($FR_k, k = 1, \dots, m$) mede a intensidade de utilização do recurso em relação à sua necessidade média e ao seu custo unitário. Quanto maior o valor de FR_k , menos flexível é o recurso k . FR_k é mensurada pela equação:

$$FR_k = c_k \frac{a_simul_k}{média_k}, k = 1, \dots, m \quad (4.1)$$

O valor c_k é o custo unitário do recurso k . A variável a_simul_k ($k = 1, \dots, m$) é a quantidade disponível simulada (ou o pico de utilização simulado) dos recursos. Para calculá-la, considera-se as atividades já programadas na sua data de início S_i e as atividades não programadas na sua data mais cedo ES_i calculada pelo CPM, e simula-se a matriz de utilização dos recursos, obtendo assim o valor de a_simul_k . O valor de $média_k$ representa a necessidade média do recurso k e é calculado da seguinte forma:

$$média_k = \max \left(\frac{\sum_{i=1}^n r_{ik} d_i}{\max(D, PCT)}, l_k \right), k = 1, \dots, m \quad (4.2)$$

Onde r_{ik} é a quantidade necessária de recurso k para executar a atividade i , d_i é a duração da atividade i e D é a data de entrega do projeto. A variável PCT é a data mais cedo de término do projeto (igual a data mais cedo de término da atividade fictícia $n + 1$ calculada pelo CPM), sendo recalculada a cada atividade programada. O parâmetro l_k é o limitante inferior (*lower bound*) de a_k ($k = 1, \dots, m$), igual a maior necessidade de uma única atividade ($l_k = \max_{i=1, \dots, n} \{r_{ik}\}, k = 1, \dots, m$).

O pseudocódigo da heurística construtiva determinística e de seus procedimentos secundários são apresentados nas Figuras 7 a 10. O procedimento da heurística construtiva determinística consiste em: seja A o conjunto de todas as atividades do projeto, L o conjunto das atividades não programadas, C o conjunto das atividades candidatas (ainda não programadas e cujas atividades precedentes já foram programadas ou que não possuem precedentes) e R o conjunto dos recursos com

custo significativo (custo unitário c_k maior do que 25% do custo unitário de atraso C_T). A heurística começa com a inicialização dos dados da instância de teste (Figura 7, linha 1), o cálculo de GCRR e RPW para todas as atividades (Figura 7, linha 2) e a aplicação do método CPM (*Critical Path Method*), calculando as datas mais cedo (ES_i e EF_i , $i = 0, \dots, n + 1$) e mais tarde (LS_i e LF_i , $i = 0, \dots, n + 1$) de início e de término das n atividades, o *makespan* (duração total) do projeto, a folga total das n atividades e a matriz de utilização de recursos (Figura 7, linha 3). Em seguida, há o cálculo de l_k e $média_k$ para todos os recursos (Figura 7, linha 4).

A matriz de utilização dos recursos é uma matriz dos m recursos pelos Mx períodos de tempo do projeto e onde é possível calcular a disponibilidade de cada recurso a_k ($k = 1, \dots, m$) e a quantidade de recurso necessária em cada instante de tempo $r_k(S, t)$ ($k = 1, \dots, m$).

As disponibilidades de recursos $a_{k_{ES}}$ ($k = 1, \dots, m$) do programa considerando a data mais cedo de início do CPM são limitantes superiores u_k (*upper bound*) das disponibilidades de recursos a_k ($k = 1, \dots, m$), gerando uma solução viável que possui o maior custo de recursos e o menor custo de atraso. O cálculo do u_k usa o programa na data mais cedo do CPM porque a heurística proposta sempre busca programar a atividade selecionada na data mais cedo possível considerando as restrições. Já o limitante inferior l_k (*lower bound*) gera uma solução viável com o menor custo de recursos e o maior custo de atraso. Assim, a quantidade disponível de recursos a_k varia entre l_k e u_k .

No próximo passo da heurística, constrói-se uma solução para o problema (Figura 7, linhas 5 a 32). Calcula-se as datas mais cedo (ES_i e EF_i , $i = 0, \dots, n + 1$) e mais tarde (LS_i e LF_i , $i = 0, \dots, n + 1$) de início e de término das n atividades e a folga total das n atividades pelo CPM (Figura 7, linha 6). Atualiza-se a lista de atividades candidatas C , calcula-se FA_i ($i = 1, \dots, n$) e ordena-se as atividades de C por ordem crescente de FA_i (Figura 7, linha 7). O cálculo da FA_i depende da regra de prioridade a ser testada ou selecionada. Calcula-se FR_k ($k = 1, \dots, m$) e os recursos são ordenados por ordem decrescente de FR_k (Figura 7, linha 8). Então a lista de atividades candidatas C é varrida da atividade menos flexível para a mais flexível. A primeira atividade menos flexível que utilizar o recurso menos flexível é selecionada (Figura 7, linha 9). Sua data de início é a data mais cedo em que sua execução não gera o maior pico de utilização

para todos recursos utilizados pela atividade ($r_{ik} > 0$) com custo significativo (recursos do conjunto R) (Figura 7, linhas 10 a 30, e Figura 8). O critério de restringir somente os recursos de R é aplicado porque a restrição do pico de recursos baratos pode evitar a obtenção de boas soluções.

Caso a data mais cedo fora do pico seja maior do que a data mais tarde LS_i , então será necessário atrasar o término do projeto para que a atividade selecionada não gere pico. Então compara-se o custo simulado com atraso (atividade fora do pico) e sem atraso (atividade dentro do pico) e seleciona-se a data de início que gera o menor custo total (Figura 7, linhas 20 a 26, e Figura 9). Após a determinação da melhor data de início para a atividade selecionada, programa-se a atividade e atualiza-se a matriz de utilização de recursos (Figura 7, linha 31). Repete-se o passo acima, até que todas as atividades do projeto sejam programadas.

Se a solução gerada possuir atraso, executa-se o procedimento *backward-forward* de Tormos e Lova (2003) para o RCPSP (Figura 10), que possui um bom desempenho computacional e possibilita reduzir a data de término do projeto considerando as restrições de recursos e as relações de precedência. O procedimento *backward-forward* é executado enquanto houver atraso e enquanto gerar uma solução melhor (Figura 7, linhas 33 a 35).

Por fim, calcula-se o valor da função objetivo (Figura 7, linha 36) e os resultados são gravados em uma planilha do Microsoft Excel (Figura 7, linha 37). Segue abaixo o pseudocódigo da heurística construtiva determinística e de seus procedimentos secundários (Figuras 7 a 10).

Figura 7 – Pseudocódigo da heurística construtiva determinística.

```

Procedimento Construtivo Determinístico ()
1 Inicializa dados
2 Calcula GCRR e RPW
3 Calcula solução CPM
4 Calcula  $I_k$  e média $_k$ 
5 enquanto lista  $L$  não é vazia faça
6     Calcula  $ES_i$ ,  $EF_i$ ,  $LS_i$ ,  $LF_i$  e Folga Total (CPM)
7     Atualiza lista  $C$ , calcula  $FA_i$  e ordena as atividades de  $C$  por  $FA_i$  crescente
8     Calcula  $FR_k$  e ordena os recursos por  $FR_k$  decrescente
9     AtivSelec = Atividade menos flexível que utiliza o recurso menos flexível
10     $S_{AtivSelec} = ES_{AtivSelec}$ 
11    para cada recurso  $k = 1$  até  $m$  faça
12        se  $r_{AtivSelec,k} > 0$  e  $k \in R$ 
13            se execução da AtivSelec estiver fora do pico de utilização do recurso  $k$ 
14                mantém  $S_{AtivSelec}$ 
15            caso contrário
16                Melhor_S = ACHA_MELHOR_S (AtivSelec, recurso  $k$ )
17                se  $S_{AtivSelec} < Melhor\_S$ 
18                    se  $Melhor\_S \leq LS_{AtivSelec}$ 
19                         $S_{AtivSelec} = Melhor\_S$ 
20                    caso contrário
21                        CustoAtual = SIMULA_FO (AtivSelec,  $S_{AtivSelec}$ )
22                        CustoComAtraso = SIMULA_FO (AtivSelec, Melhor_S)
23                        se CustoComAtraso < CustoAtual
24                             $S_{AtivSelec} = Melhor\_S$ 
25                        fim
26                    fim
27                fim
28            fim
29        fim
30    fim
31    Atualiza matriz de utilização de recursos
32 fim
33 enquanto for possível reduzir a data de término do projeto e houver atraso faça
34     BACKWARD_FORWARD ()
35 fim
36 Calcula valor FO
37 Grava dados da solução
fim Construtivo Determinístico.

```

Figura 8 – Pseudocódigo do procedimento ACHA_MELHOR_S.

```

ACHA_MELHOR_S (atividade selec, recurso k)
1 proximo_t = Data_Fim_Picok + 1
2 enquanto execução da atividade selec estiver dentro do pico de utilização do recurso k faça
3     para cada atividade i = 0 até n faça
4         se i = selec
5             ES_Simuli = proximo_t
6         caso contrário
7             se i já foi programada
8                 ES_Simuli = Si
9             caso contrário
10                Calcula ES_Simuli (CPM)
11            fim
12        fim
13    fim
14    Simula matriz de utilização do recurso k considerando ES_Simuli e acha Picok
15    se execução da atividade selec estiver fora do pico de utilização do recurso k
16        Melhor_S = proximo_t
17    caso contrário
18        proximo_t = Data_Fim_Picok + 1
19    fim
20 fim
21 retorna Melhor_S
fim

```

Figura 9 – Pseudocódigo do procedimento SIMULA_FO.

```

SIMULA_FO (atividade selec, tempo início)
1 para cada atividade i = 0 até n faça
2     se i = selec
3         ES_Simuli = início
4     caso contrário
5         se i já foi programada
6             ES_Simuli = Si
7         caso contrário
8             Calcula ES_Simuli (CPM)
9         fim
10    fim
11 fim
12 Simula matriz de utilização de recursos considerando ES_Simuli
13 Calcula valor FO
14 retorna valor FO simulado
fim

```

Figura 10 – Pseudocódigo do procedimento BACKWARD_FORWARD.

```

BACKWARD_FORWARD
1 'Backward
2 Ordena a lista A em ordem decrescente de  $F_i + i$ 
3 para cada atividade  $i = 0$  até  $n$  da lista A faça
4     Programa  $i$  na data mais tarde considerando  $F_n$  original,  $a_k$  e as relações de precedência
5     Atualiza matriz de utilização de recursos
6 fim
7 'Forward
8 Ordena a lista A em ordem crescente de  $S_i + i$ 
9 para cada atividade  $i = 0$  até  $n$  da lista A faça
10    Programa  $i$  na data mais cedo considerando  $a_k$  e as relações de precedência
11    Atualiza matriz de utilização de recursos
12 fim
fim

```

3.2.2 Heurística construtiva probabilística

O segundo método proposto é uma heurística construtiva probabilística. A ideia é selecionar a próxima atividade a ser programada aleatoriamente, mas com viés baseado na flexibilidade da atividade FA_i e na flexibilidade do recurso FR_k . A melhor solução encontrada após as iterações é a solução do problema. O conceito de seleção aleatória com viés é baseado em Arcus (1965). O pseudocódigo da heurística construtiva probabilística é mostrado na Figura 13.

A probabilidade $prob_i$ de cada atividade é calculada da seguinte maneira (Figura 13, linha 22): após a ordenação da lista de atividades candidatas C pela flexibilidade FA_i e após a ordenação dos recursos pela flexibilidade FR_k , ordena-se novamente a lista C considerando como atividade menos flexível aquela que utiliza o recurso menos flexível. Assim, a lista de atividades candidatas C é varrida da atividade menos flexível para a mais flexível e considerando o recurso menos flexível. A primeira atividade menos flexível que utilizar o recurso menos flexível passa a ser a atividade menos flexível, a segunda atividade que utilizar o recurso menos flexível passa a ser a segunda menos flexível e assim por diante. Após todas as atividades de C serem avaliadas, passa-se para o segundo recurso menos flexível. A lista de atividades candidatas C é varrida novamente, e assim por diante, até que todas as atividades sejam reordenadas. A Figura 11 mostra um exemplo dessa nova ordenação da lista C .

Figura 11 – Exemplo de ordenação da lista de atividades candidatas C combinando FA_i e FR_k

I	Nº Ativ. (ordenada por FA_i crescente)	Nº Recurso (ordenado por FR_k decrescente)				Nova ordem das atividades	
		3	1	4	2	Novo I	Nº Ativ.
1	2		x		x	1	1
2	1	x	x			2	3
3	3	x		x	x	3	4
4	5			x	x	4	2
5	4	x		x		5	5

A probabilidade $prob_i$, diminui exponencialmente com l , que é a nova posição da atividade i na lista C . Assim, quanto menor a flexibilidade combinada da atividade, menor sua posição e maior a probabilidade de a atividade ser selecionada. A probabilidade é calculada através da equação abaixo:

$$prob_i = 0,8^{(2l+1)} \quad (4.3)$$

Após o cálculo, os valores são normalizados. Também foram testadas as equações $prob_i = 0,8^{(l+1)}$ e $prob_i = 0,8^{(3l+1)}$, que geraram resultados piores. A Figura 12 mostra um exemplo de como a probabilidade decresce exponencialmente com a posição da atividade. Caso a seleção fosse aleatória sem viés, cada uma das 6 atividades candidatas teria a mesma probabilidade de 16,7%.

Figura 12 – Exemplo de probabilidades com viés para uma lista com 6 atividades candidatas.

Na heurística construtiva probabilística, a construção da solução pode ser realizada de cinco formas diferentes, que serão chamadas de passos 1 a 5. O passo 1 (Figura 13, linhas 24 e 25) corresponde ao mesmo procedimento de construção da solução da heurística construtiva determinística, no qual a data de início da atividade selecionada é calculada de forma que a execução da atividade não gere o maior pico de utilização dos recursos. A diferença consiste na seleção aleatória da próxima atividade com base em $prob_i$. Quando o projeto é de “baixo recurso”, ou seja, quando $média_k = l_k$ para todos os recursos, a instância de teste pode ser considerada fácil e o passo 1 tem duração de 5.000 iterações. Para as instâncias mais difíceis, o passo 1 dura 10.000 iterações. Esses valores foram determinados após testes nos quais foram verificados que após um certo número de iterações não havia melhoria nas soluções geradas.

Nos passos 2 a 5, constrói-se uma solução na qual a data de início da atividade selecionada é a data mais cedo possível respeitando as relações de precedência e a limitação de recursos (Figura 13, linhas 26 a 35).

No passo 2, a quantidade disponível (ou pico de utilização) a_k dos recursos utilizados pela atividade ($r_{ik} > 0$) pertencentes ao conjunto R dos recursos com custo significativo deve respeitar o valor $limite_k$ (Figura 13, linhas 26 a 35). O valor de $limite_k$ é sorteado aleatoriamente dentro do intervalo $a_{melhor_sol_k} \pm 2$ para os recursos caros ($c_k > 0.75C_T$), sendo que $a_{melhor_sol_k}$ é a quantidade disponível da melhor solução encontrada até o momento. Para os demais recursos, o intervalo é $a_{melhor_sol_k} \pm 3$ (Figura 13, linhas 4 a 12). A variação para mais e para menos é interessante, pois uma solução melhor pode ser gerada com a redução da quantidade de um recurso e o aumento de outro. O passo 2 tem duração de 15.000 iterações. Para os projetos de “baixo recurso”, a primeira execução do passo 2 tem duração de 20.000 iterações.

No passo 4, que dura 2.500 iterações, a ideia é explorar a vizinhança da melhor solução, de modo que a quantidade disponível a_k dos recursos utilizados pela atividade ($r_{ik} > 0$) pertencentes ao conjunto R deve respeitar o valor $limite_k = a_{melhor_sol_k}$ (Figura 13, linhas 13 a 17 e linhas 26 a 35).

Os passos 3 (duração de 5.000 iterações) e 5 (2.500 iterações) diferem dos passos 2 e 4, respectivamente, apenas no fato de que a quantidade disponível a_k dos recursos

utilizados pela atividade e com custo baixo ($c_k \leq 0.25C_T$) também deve respeitar o limite $limite_k$ (Figura 13, linhas 4 a 17 e linhas 26 a 35).

Os passos são executados na sequência de 1 a 5. Após a primeira execução do passo 5, executa-se novamente do passo 2 até o 5, e assim sucessivamente até o término da execução da instância. O critério de parada selecionado foi o tempo limite de execução, por ser o mesmo critério adotado para o CPLEX nos experimentos computacionais (Figura 13, linha 2). Além disso, há outros critérios para acelerar a execução das instâncias:

- Em um programa parcial, caso o valor do custo dos recursos seja maior ou igual ao custo total da melhor solução encontrada até o momento, então encerra-se a iteração e inicia-se a próxima, pois nesse caso não será possível superar a melhor solução (Figura 13, linhas 37 a 39).
- Em um programa completo, se o valor total da função objetivo for igual ao valor do custo dos recursos com disponibilidade igual a l_k , então encerra-se a instância de teste e inicia-se a próxima, já que foi encontrada a solução de menor custo possível (Figura 13, linhas 48 e 49).
- Caso a iteração atual seja maior do que 85.000 (ou seja, os passos de 2 a 5 foram executados completamente pelo menos três vezes), e a heurística está há mais de 75.000 iterações sem melhorar a solução e o projeto é de “baixo recurso”, então provavelmente a melhor solução já foi encontrada, a instância é encerrada e a próxima é iniciada (Figura 13, linhas 48 e 49).
- Em um programa completo, se o valor do custo dos recursos for igual ao valor do custo dos recursos com disponibilidade igual a l_k e a heurística estiver nos passos 1, 2 ou 3, então inicia-se o passo 4 (vizinhança da melhor solução) imediatamente, já que foi encontrada uma solução com menor custo de recursos possível, faltando apenas melhorar o custo de atraso (Figura 13, linhas 50 a 54).

O pseudocódigo da heurística construtiva probabilística é mostrado na Figura 13. As linhas em preto são os mesmos procedimentos da heurística construtiva determinística, enquanto que as linhas em azul trazem as mudanças da probabilística.

Figura 13 – Pseudocódigo da heurística construtiva probabilística.

```

Procedimento Construtivo Probabilístico ()
1 Inicializa dados (idem linhas 1 a 4 do Procedimento Construtivo Determinístico)
2 enquanto tempo execução < tempo limite faça
3   Inicializa dados
4   se ciclo = 2 ou ciclo = 3
5     para cada recurso k = 1 até m faça
6       se  $c_k > C_T * 0,75$ 
7         Sorteia  $limite_k$  entre  $a\_melhor\_sol_k \pm 2$  (entre  $l_k$  e  $u_k$ )
8       caso contrário
9         Sorteia  $limite_k$  entre  $a\_melhor\_sol_k \pm 3$  (entre  $l_k$  e  $u_k$ )
10      fim
11    fim
12  fim
13  se ciclo = 4 ou ciclo = 5
14    para cada recurso k = 1 até m faça
15       $limite_k = a\_melhor\_sol_k$ 
16    fim
17  fim
18  enquanto lista L não é vazia faça
19    Calcula  $ES_i$ ,  $EF_i$ ,  $LS_i$ ,  $LF_i$  e Folga Total (CPM)
20    Atualiza lista C, calcula  $FA_i$  e ordena as atividades de C pela  $FA_i$ 
21    Calcula  $FR_k$  e ordena os recursos pela  $FR_k$ 
22    Ordena novamente lista C pela  $FA_i$  e pela  $FR_k$  combinados e calcula  $prob_i$  com viés
23    sorteia AtivSelec
24    se ciclo = 1
25      Define  $S_{AtivSelec}$  (idem linhas 10 a 30 do Procedimento Construtivo Determinístico)
26    caso contrário
27       $S_{AtivSelec} = ES_{AtivSelec}$ 
28      enquanto  $S_{AtivSelec}$  não for viável faça
29        para cada recurso k = 1 até m faça
30          se (ciclo = 2 ou 4 e  $r_{AtivSelec,k} > 0$  e  $k \in R$ ) ou (ciclo = 3 ou 5 e  $r_{AtivSelec,k} > 0$ )
31            Acha  $S_{AtivSelec}$  mais cedo considerando  $limite_k$  e precedência
32          fim
33        fim
34      fim
35    fim
36    Atualiza matriz de utilização de recursos e calcula valor FO Recursos
37    se valor FO Recursos  $\geq$  Melhor FO Total
38      Encerra iteração e inicia a próxima
39    fim
40  fim
41  enquanto for possível reduzir a data de término do projeto e houver atraso faça
42    BACKWARD_FORWARD ()
43  fim
44  Calcula valor FO Total
45  se valor FO Total < Melhor FO Total
46    Atualiza dados da melhor solução
47  fim
48  se valor FO Total =  $FO_{l_k}$  ou (iter > 85.000 e iter > 75.000 + IterMelhorSol e Baixo Recurso)
49    Encerra instância e inicia a próxima
50  caso contrário
51    se valor FO Recursos =  $FO_{l_k}$  e ciclo < 4
52      inicia ciclo = 4
53    fim
54  fim
55 fim
56 Grava dados da solução
fim Construtivo Probabilístico.

```

4 EXPERIMENTOS COMPUTACIONAIS

4.1 INSTÂNCIAS DE TESTES

Os testes computacionais serão realizados nas mesmas instâncias de teste do primeiro grupo de instâncias de Yamashita; Armentano e Laguna (2006) para o RACP. Os autores geraram uma instância para cada combinação dos seguintes parâmetros, totalizando 144 instâncias:

- Número de atividades (n): 30, 60, 90 e 120
- Número de recursos (m): 4, 6 e 8
- Fator de recursos (RF): 0,25; 0,50; 0,75 e 1,00
- Complexidade da rede (NC): 1,5; 1,8 e 2,1
- Fator de data de entrega (DF): 1,2

O fator de recursos (RF) varia entre 0 e 1 e mede a utilização dos diferentes tipos de recursos necessários para executar uma atividade. Se RF for igual a 1, todas as atividades precisam de todos os tipos de recursos e se RF for igual a 0, as atividades não precisam de nenhum tipo de recurso. A complexidade da rede (NC) reflete o número médio de atividades imediatamente sucessoras. Os valores utilizados de RF e NC seguem os parâmetros utilizados na biblioteca PSPLIB de Kolisch e Sprecher (1997) e em muitos trabalhos desenvolvidos sobre programação de projetos.

Já o fator de data de entrega (DF) é utilizado para calcular a data de entrega do projeto em função do caminho crítico, conforme Drexel e Kimms (2001):

$$D = DF * EF_{n+1} \quad (5.1)$$

Sendo que EF_{n+1} é o instante de tempo mais cedo possível para terminar o projeto, calculado através do método CPM (*Critical Path Method*). Yamashita; Armentano e Laguna (2006) escolheram o valor de DF igual a 1,2 para evitar a geração de instâncias com tempos muito justos ou muito folgados com relação à data de entrega.

Por fim, Yamashita; Armentano e Laguna (2006) escolheram os custos unitários c_k de cada recurso aleatoriamente a partir de uma distribuição uniforme $U[1, 10]$. Todos os valores são inteiros, com exceção dos parâmetros de custo unitário. Os autores

criaram as instâncias no programa ProGen de Kolisch; Sprecher e Drexl (1995) e adaptaram-nas para o RACP conforme a metodologia de Drexl e Kimms (2001).

As instâncias de teste com 4 recursos foram obtidas diretamente da biblioteca PSPLIB para o RCPSP. As instâncias com 30 atividades e 4 recursos também estão no segundo grupo de instâncias de Yamashita; Armentano e Laguna (2006), composto pelas mesmas utilizadas por Drexl e Kimms (2001).

Neste trabalho, também será testado o valor de DF igual a 1,1 e igual a 1,3 nas instâncias com 30 a 90 atividades, para melhor avaliação do impacto do custo do atraso no custo total e na data de término do projeto. Dessa forma, o número total de instâncias de testes é 360.

Para testar o RACPT, será considerado o valor de custo por atraso C_T igual a 0,25 da soma dos custos unitários dos recursos, seguindo Van Peteghem e Vanhoucke (2015) e Shadrokh e Kianfar (2007). Se este valor for muito alto, a solução do RACPT é muito próxima da solução do RACP e se for muito baixo, as disponibilidades dos recursos são muito próximas dos seus limitantes inferiores e o atraso do projeto é muito alto, conforme pode ser verificado nos resultados de Van Peteghem e Vanhoucke (2015). Pode-se considerar que valores muito baixos de custo por atraso não acontecem na prática.

4.2 PROCEDIMENTO EXPERIMENTAL

As heurísticas construtivas determinística e probabilística foram implementadas na linguagem *Visual Basic for Applications* (VBA) com interface no programa Microsoft Excel 2013. Seu código fonte encontra-se no apêndice A. Todos os testes foram realizados em um computador com processador Intel i7 com 2,93 GHz e com memória RAM de 16 GB.

Na primeira etapa dos testes, será escolhida a melhor combinação de regra de prioridade e regra de desempate para as heurísticas construtivas determinística e probabilística. Assim, caso duas atividades possuam o mesmo valor de regra de prioridade, a atividade com o melhor valor da regra de desempate será priorizada. Se o empate persistir, a maior prioridade será da atividade com o menor índice (menor ordem).

Para a heurística determinística serão avaliados os resultados obtidos nas 360 instâncias de teste. Já para a heurística probabilística, a melhor regra será determinada pelos resultados nas 36 instâncias com 30 atividades e 8 recursos. Cada instância será executada três vezes por até 900 segundos e será considerada a média dos três resultados obtidos.

A Tabela 2 apresenta as combinações de regra de prioridade e de regra de desempate que serão testadas. A regra combinada e a seleção aleatória serão testadas somente para a heurística construtiva probabilística. A regra combinada consiste em executar a regra GCRR na primeira execução dos passos 1 a 5, a regra RPW na segunda execução dos passos 2 a 5 e a regra MLFT na terceira execução dos passos 2 a 5 e assim sucessivamente. A regra aleatória será testada para verificar se há ganho no desempenho da heurística ao usar uma regra de prioridade mais sofisticada.

Tabela 2 – Combinações de regra de prioridade e regra de desempate selecionadas.

Regra de prioridade	Regra de desempate
Maior necessidade de recurso acumulada (GCRR)	Mínimo instante mais tarde de início (MLFT)
Rank posicional ponderado (RPW)	Mínimo instante mais tarde de início (MLFT)
Mínimo instante mais tarde de início (MLFT)	Menor folga total (MAS)
Menor folga total (MAS)	Maior número de sucessores imediatos (GNIS)
Combinado (GCRR, RPW e MLFT)	
Seleção aleatória (RAN)	

Na segunda etapa dos testes, a melhor combinação de regra de prioridade e regra de desempate para as heurísticas construtivas determinística e probabilística será comparada aos resultados obtidos pela implementação no CPLEX 12.6.0.0 do modelo matemático apresentado no capítulo 3, utilizando como interface o programa IBM ILOG CPLEX Optimization Studio. O código fonte desta implementação encontra-se no apêndice B.

O critério de parada escolhido foi o tempo limite de execução. Kolisch e Hartmann (2006) recomendaram o número de programas gerados como o melhor critério, porém este não é aplicável ao CPLEX. Como o RACPT é um problema NP-difícil, não é possível encontrar a solução ótima em tempo razoável nem mesmo para alguns problemas pequenos de 30 atividades e 4 recursos. Assim, foram realizados alguns testes preliminares para definir os tempos limites de execução para a heurística probabilística e para o CPLEX, de forma a equilibrar tempo e ganho nos resultados

(vide Tabela 3). Para a heurística construtiva probabilística, o resultado considerado será a média dos três resultados obtidos.

Tabela 3 – Tempo limite de execução da heurística probabilística e do CPLEX (em segundos).

Qtd. Atividades	DF	Heurística construtiva probabilística	CPLEX
30	Todos	3 vezes de 900	1.800
60	Todos	3 vezes de 1.800	3.600
90	Todos	3 vezes de 1.800	3.600
90	1,2	3 vezes de 1.800	7.200
120	1,2	3 vezes de 1.800	3.600
120	1,2	1 vez de 3.600	7.200

Os resultados obtidos pela heurística determinística e a média dos resultados obtidos pela heurística probabilística serão comparados aos limitantes inferiores e superiores obtidos pelo CPLEX através das seguintes medidas (LEMOS, 2013):

$$Gap = \frac{UB - LB_{CPLEX}}{LB_{CPLEX}} \quad (5.2)$$

$$Diferença = \frac{UB_{Método proposto} - UB_{CPLEX}}{UB_{Método proposto}} \quad (5.3)$$

4.3 RESULTADOS

As Tabelas 4 a 14 mostram os resultados da primeira etapa dos testes para as heurísticas construtivas determinística (HCD) e probabilística (HCP). Os valores em azul correspondem ao melhor resultado obtido para aquela instância.

Tabela 4 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 4 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	212,6	206,2	212,6	212,6
0,25	1,5	1,2	203,9	188,9	203,9	203,9
0,25	1,5	1,3	203,9	184,6	203,9	203,9
0,50	1,5	1,1	556,1	556,1	556,1	544,8
0,50	1,5	1,2	542,8	542,8	542,8	544,8
0,50	1,5	1,3	506,8	491,1	527,1	503,4
0,75	1,5	1,1	376,1	376,1	340,7	375,3
0,75	1,5	1,2	314,3	377,5	340,7	375,3
0,75	1,5	1,3	314,3	355,1	324,7	375,3
1,00	1,5	1,1	324,7	326,1	296,2	310,5
1,00	1,5	1,2	322,2	326,1	288,7	302,9
1,00	1,5	1,3	296,4	292,4	288,7	309,9
0,25	1,8	1,1	227,7	227,7	227,7	178,8
0,25	1,8	1,2	213,4	213,4	164,5	213,4
0,25	1,8	1,3	165,6	165,6	165,6	165,6
0,50	1,8	1,1	278,7	278,0	278,0	278,0
0,50	1,8	1,2	255,5	259,5	259,5	259,5
0,50	1,8	1,3	262,5	257,5	257,5	257,5
0,75	1,8	1,1	408,7	440,0	406,8	398,4
0,75	1,8	1,2	405,6	416,4	413,3	394,7
0,75	1,8	1,3	419,0	438,3	419,0	419,0
1,00	1,8	1,1	420,4	403,3	455,8	384,8
1,00	1,8	1,2	420,4	408,7	455,8	384,9
1,00	1,8	1,3	420,4	385,6	467,0	384,9
0,25	2,1	1,1	125,6	125,6	125,6	125,6
0,25	2,1	1,2	101,8	101,8	101,8	101,8
0,25	2,1	1,3	99,0	99,0	99,0	99,0
0,50	2,1	1,1	519,6	519,6	486,4	499,3
0,50	2,1	1,2	431,3	460,8	431,3	468,9
0,50	2,1	1,3	481,2	440,0	475,4	468,9
0,75	2,1	1,1	705,9	705,9	705,9	709,7
0,75	2,1	1,2	668,7	668,7	668,7	646,4
0,75	2,1	1,3	646,4	646,4	646,4	646,4
1,00	2,1	1,1	549,3	549,3	548,8	580,6
1,00	2,1	1,2	513,3	513,3	512,8	580,6
1,00	2,1	1,3	513,3	547,0	567,4	648,9
Total			13.427,2	13.494,3	13.465,7	13.557,8

Tabela 5 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 6 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	237,0	237,0	237,0	237,0
0,25	1,5	1,2	237,0	237,0	237,0	237,0
0,25	1,5	1,3	214,2	214,2	214,2	214,2
0,50	1,5	1,1	502,6	479,6	568,2	518,2
0,50	1,5	1,2	461,3	460,7	568,2	518,2
0,50	1,5	1,3	431,1	436,6	619,0	551,6
0,75	1,5	1,1	841,4	872,1	756,1	827,7
0,75	1,5	1,2	796,7	822,3	800,4	783,1
0,75	1,5	1,3	761,0	786,6	702,0	744,7
1,00	1,5	1,1	835,2	828,2	883,5	940,2
1,00	1,5	1,2	801,8	794,8	850,1	940,2
1,00	1,5	1,3	760,1	769,7	816,8	923,0
0,25	1,8	1,1	419,0	419,0	419,0	419,0
0,25	1,8	1,2	394,1	394,1	394,1	394,1
0,25	1,8	1,3	442,5	442,5	442,5	415,5
0,50	1,8	1,1	673,0	673,0	673,0	679,6
0,50	1,8	1,2	634,0	634,0	623,3	639,7
0,50	1,8	1,3	580,4	580,4	580,4	580,4
0,75	1,8	1,1	638,4	638,4	638,4	624,8
0,75	1,8	1,2	620,2	620,2	620,2	612,3
0,75	1,8	1,3	575,8	660,2	602,7	602,7
1,00	1,8	1,1	678,0	677,8	728,8	656,1
1,00	1,8	1,2	664,4	644,0	741,2	664,3
1,00	1,8	1,3	641,7	644,0	741,2	688,4
0,25	2,1	1,1	410,8	410,8	410,8	426,4
0,25	2,1	1,2	356,2	356,2	356,2	371,8
0,25	2,1	1,3	309,4	309,4	309,4	309,4
0,50	2,1	1,1	476,9	476,9	476,9	476,9
0,50	2,1	1,2	476,9	476,9	476,9	476,9
0,50	2,1	1,3	476,9	476,9	476,9	476,9
0,75	2,1	1,1	437,8	437,8	466,8	461,7
0,75	2,1	1,2	410,6	431,7	439,6	398,2
0,75	2,1	1,3	395,5	446,2	396,0	398,2
1,00	2,1	1,1	723,0	723,0	733,0	846,8
1,00	2,1	1,2	674,8	674,8	674,8	682,9
1,00	2,1	1,3	665,8	665,8	665,8	664,9
Total			19.655,9	19.852,9	20.340,8	20.403,3

Tabela 6 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 30 atividades e 8 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	421,5	421,5	421,5	421,5
0,25	1,5	1,2	411,5	411,5	411,5	411,5
0,25	1,5	1,3	472,3	472,3	472,3	472,3
0,50	1,5	1,1	584,8	584,8	543,6	580,3
0,50	1,5	1,2	521,1	636,3	521,7	521,7
0,50	1,5	1,3	522,1	596,7	497,9	497,9
0,75	1,5	1,1	805,6	903,5	840,7	821,1
0,75	1,5	1,2	757,8	903,5	792,9	773,4
0,75	1,5	1,3	738,7	738,7	724,8	805,9
1,00	1,5	1,1	947,0	971,8	955,9	968,7
1,00	1,5	1,2	945,1	945,1	951,8	932,1
1,00	1,5	1,3	879,5	879,5	960,7	877,5
0,25	1,8	1,1	465,2	465,2	465,2	465,2
0,25	1,8	1,2	462,9	462,9	462,9	462,9
0,25	1,8	1,3	452,2	452,2	445,5	445,5
0,50	1,8	1,1	788,6	788,6	788,6	788,6
0,50	1,8	1,2	788,6	788,6	788,6	788,6
0,50	1,8	1,3	764,4	764,4	764,4	764,4
0,75	1,8	1,1	894,8	894,8	889,1	894,8
0,75	1,8	1,2	830,5	830,5	830,1	830,5
0,75	1,8	1,3	932,4	932,4	793,6	932,4
1,00	1,8	1,1	958,8	958,8	1.025,4	1.076,4
1,00	1,8	1,2	924,9	924,9	1.058,4	1.079,3
1,00	1,8	1,3	924,9	924,9	1.058,4	1.070,6
0,25	2,1	1,1	545,7	545,7	545,7	545,7
0,25	2,1	1,2	477,3	477,3	477,3	477,3
0,25	2,1	1,3	435,8	435,8	435,8	435,8
0,50	2,1	1,1	646,2	645,4	646,2	657,6
0,50	2,1	1,2	609,2	609,2	609,2	600,0
0,50	2,1	1,3	600,0	600,0	600,0	606,7
0,75	2,1	1,1	454,5	454,5	508,3	482,2
0,75	2,1	1,2	506,4	506,4	455,8	469,4
0,75	2,1	1,3	527,4	527,4	441,1	460,5
1,00	2,1	1,1	1.303,4	1.303,4	1.309,0	1.372,1
1,00	2,1	1,2	1.278,4	1.278,4	1.246,4	1.372,1
1,00	2,1	1,3	1.278,4	1.278,4	1.233,9	1.372,1
Total			25.857,8	26.315,2	25.974,0	26.534,4

Tabela 7 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 4 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	182,8	182,8	182,8	167,7
0,25	1,5	1,2	182,8	182,8	182,8	167,7
0,25	1,5	1,3	182,8	182,8	182,8	167,7
0,50	1,5	1,1	456,5	453,6	477,8	477,8
0,50	1,5	1,2	457,0	422,5	457,0	457,0
0,50	1,5	1,3	457,0	423,5	499,0	499,0
0,75	1,5	1,1	1.055,6	958,8	1.154,6	1.112,4
0,75	1,5	1,2	1.258,3	1.131,3	1.022,4	1.069,9
0,75	1,5	1,3	1.258,3	1.131,3	1.024,8	942,2
1,00	1,5	1,1	1.413,6	1.387,5	1.351,0	1.228,6
1,00	1,5	1,2	1.434,3	1.387,5	1.351,0	1.242,1
1,00	1,5	1,3	1.434,3	1.387,5	1.351,0	1.242,1
0,25	1,8	1,1	284,5	269,8	266,5	284,5
0,25	1,8	1,2	278,3	272,2	272,2	272,2
0,25	1,8	1,3	278,3	278,3	278,3	272,2
0,50	1,8	1,1	583,8	583,8	583,8	576,5
0,50	1,8	1,2	580,7	580,7	580,7	580,7
0,50	1,8	1,3	569,1	569,1	569,1	582,8
0,75	1,8	1,1	871,3	871,3	904,2	913,9
0,75	1,8	1,2	871,3	871,3	904,2	913,9
0,75	1,8	1,3	895,1	895,1	904,2	913,9
1,00	1,8	1,1	949,3	916,4	949,3	1.043,6
1,00	1,8	1,2	907,7	874,8	942,6	1.043,6
1,00	1,8	1,3	900,8	1.037,5	1.016,2	1.053,5
0,25	2,1	1,1	195,3	173,7	173,6	170,3
0,25	2,1	1,2	176,3	176,3	164,1	167,8
0,25	2,1	1,3	167,8	167,8	167,8	161,6
0,50	2,1	1,1	488,3	488,3	539,7	532,3
0,50	2,1	1,2	539,0	539,0	539,0	539,0
0,50	2,1	1,3	598,2	598,2	598,2	598,2
0,75	2,1	1,1	406,1	406,1	382,5	382,5
0,75	2,1	1,2	406,1	406,1	382,5	382,5
0,75	2,1	1,3	406,1	406,1	382,5	382,5
1,00	2,1	1,1	526,0	526,0	503,7	533,8
1,00	2,1	1,2	526,0	526,0	503,7	550,8
1,00	2,1	1,3	526,0	526,0	503,7	550,8
Total			22.704,7	22.191,6	22.249,1	22.177,7

Tabela 8 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 6 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	332,1	332,1	332,1	326,3
0,25	1,5	1,2	332,1	332,1	332,1	326,3
0,25	1,5	1,3	329,2	329,2	329,2	293,8
0,50	1,5	1,1	550,5	574,5	559,0	593,8
0,50	1,5	1,2	568,2	593,8	524,2	529,7
0,50	1,5	1,3	547,7	547,7	560,3	586,7
0,75	1,5	1,1	1.050,2	1.080,9	1.144,8	1.208,1
0,75	1,5	1,2	1.049,5	1.069,3	1.146,9	979,0
0,75	1,5	1,3	1.012,2	1.069,3	977,5	964,3
1,00	1,5	1,1	1.294,9	1.309,0	1.293,7	1.266,0
1,00	1,5	1,2	1.306,3	1.268,1	1.320,6	1.320,0
1,00	1,5	1,3	1.306,3	1.300,3	1.320,6	1.320,0
0,25	1,8	1,1	490,7	490,7	490,7	508,9
0,25	1,8	1,2	457,3	457,3	450,3	457,3
0,25	1,8	1,3	457,3	457,3	471,3	450,3
0,50	1,8	1,1	962,8	974,0	962,8	940,3
0,50	1,8	1,2	925,5	974,0	925,5	940,3
0,50	1,8	1,3	925,5	949,4	925,5	913,2
0,75	1,8	1,1	925,5	925,5	923,5	916,1
0,75	1,8	1,2	895,1	912,3	912,3	944,5
0,75	1,8	1,3	894,3	894,3	912,3	956,1
1,00	1,8	1,1	897,5	905,6	904,6	895,6
1,00	1,8	1,2	856,9	871,7	864,0	877,7
1,00	1,8	1,3	873,9	866,2	857,2	775,0
0,25	2,1	1,1	431,8	431,8	452,3	436,9
0,25	2,1	1,2	429,9	429,9	441,0	451,5
0,25	2,1	1,3	411,2	411,2	418,2	402,3
0,50	2,1	1,1	615,9	615,9	615,9	615,9
0,50	2,1	1,2	671,2	671,2	671,2	671,2
0,50	2,1	1,3	592,4	592,4	554,1	568,2
0,75	2,1	1,1	596,6	598,5	596,6	574,8
0,75	2,1	1,2	550,1	570,4	550,1	571,9
0,75	2,1	1,3	550,1	570,4	547,3	571,9
1,00	2,1	1,1	1.245,1	1.247,5	1.298,5	1.270,8
1,00	2,1	1,2	1.139,9	1.240,6	1.306,1	1.287,2
1,00	2,1	1,3	1.158,8	1.196,8	1.308,8	1.209,7
Total			27.633,9	28.060,8	28.200,4	27.921,3

Tabela 9 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 60 atividades e 8 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	712,4	712,4	712,4	712,4
0,25	1,5	1,2	632,9	632,9	632,9	632,9
0,25	1,5	1,3	563,3	563,3	563,3	563,3
0,50	1,5	1,1	796,9	836,0	785,5	790,6
0,50	1,5	1,2	807,6	814,4	788,9	773,5
0,50	1,5	1,3	807,6	807,6	791,8	773,5
0,75	1,5	1,1	1.312,9	1.354,8	1.369,6	1.244,2
0,75	1,5	1,2	1.346,7	1.362,7	1.248,5	1.244,2
0,75	1,5	1,3	1.248,3	1.254,8	1.248,5	1.251,8
1,00	1,5	1,1	1.757,9	1.849,2	1.703,6	1.644,0
1,00	1,5	1,2	1.757,9	1.849,2	1.721,6	1.633,1
1,00	1,5	1,3	1.799,7	1.856,1	1.721,6	1.633,1
0,25	1,8	1,1	700,8	717,8	715,5	676,6
0,25	1,8	1,2	686,9	686,9	686,9	686,9
0,25	1,8	1,3	675,5	675,5	675,5	675,5
0,50	1,8	1,1	1.146,9	1.159,0	1.125,2	1.040,1
0,50	1,8	1,2	1.029,2	1.029,2	995,2	1.026,5
0,50	1,8	1,3	1.000,7	996,0	992,8	1.193,5
0,75	1,8	1,1	1.102,8	1.102,8	1.192,0	1.192,0
0,75	1,8	1,2	1.150,0	1.150,0	1.127,7	1.080,0
0,75	1,8	1,3	1.124,2	1.124,2	1.213,0	1.213,0
1,00	1,8	1,1	1.319,1	1.287,0	1.326,7	1.452,2
1,00	1,8	1,2	1.279,3	1.278,8	1.328,7	1.427,6
1,00	1,8	1,3	1.288,0	1.340,9	1.326,1	1.446,6
0,25	2,1	1,1	808,1	808,1	808,1	865,8
0,25	2,1	1,2	694,6	694,6	694,6	682,1
0,25	2,1	1,3	633,3	633,3	642,8	669,3
0,50	2,1	1,1	902,0	902,0	902,0	902,0
0,50	2,1	1,2	911,3	911,3	911,3	902,0
0,50	2,1	1,3	857,6	857,6	826,2	822,8
0,75	2,1	1,1	699,8	691,0	671,0	696,8
0,75	2,1	1,2	699,8	691,0	671,0	696,8
0,75	2,1	1,3	748,5	748,5	719,7	652,6
1,00	2,1	1,1	1.886,7	1.886,7	1.855,0	1.807,3
1,00	2,1	1,2	1.886,7	1.886,7	1.855,0	1.807,3
1,00	2,1	1,3	1.886,7	1.886,7	1.855,0	1.807,3
Total			38.662,2	39.038,8	38.405,1	38.318,8

Tabela 10 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 4 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	303,9	303,9	320,2	315,2
0,25	1,5	1,2	303,9	303,9	320,2	315,2
0,25	1,5	1,3	303,9	303,9	320,2	315,2
0,50	1,5	1,1	734,1	760,1	794,3	827,2
0,50	1,5	1,2	733,3	770,2	756,8	759,8
0,50	1,5	1,3	755,8	770,2	756,8	759,8
0,75	1,5	1,1	1.452,7	1.502,8	1.349,9	1.361,0
0,75	1,5	1,2	1.344,1	1.502,8	1.354,1	1.460,5
0,75	1,5	1,3	1.353,1	1.367,7	1.383,8	1.426,7
1,00	1,5	1,1	1.606,4	1.638,1	1.529,5	1.548,5
1,00	1,5	1,2	1.604,3	1.637,5	1.556,7	1.548,5
1,00	1,5	1,3	1.604,3	1.637,5	1.556,7	1.548,5
0,25	1,8	1,1	437,4	437,4	450,0	448,2
0,25	1,8	1,2	417,5	435,9	450,0	448,2
0,25	1,8	1,3	392,6	435,9	450,0	448,2
0,50	1,8	1,1	897,5	878,6	935,2	965,2
0,50	1,8	1,2	897,5	876,5	935,2	954,4
0,50	1,8	1,3	851,2	854,4	935,2	992,1
0,75	1,8	1,1	1.196,9	1.196,9	1.204,5	1.271,2
0,75	1,8	1,2	1.188,9	1.268,1	1.211,2	1.271,2
0,75	1,8	1,3	1.188,9	1.268,1	1.211,2	1.271,2
1,00	1,8	1,1	1.518,7	1.519,4	1.332,6	1.265,0
1,00	1,8	1,2	1.542,7	1.526,6	1.332,6	1.265,0
1,00	1,8	1,3	1.542,7	1.526,6	1.332,6	1.265,0
0,25	2,1	1,1	241,4	241,4	230,3	201,5
0,25	2,1	1,2	242,3	242,3	230,3	216,7
0,25	2,1	1,3	220,1	220,1	227,8	216,7
0,50	2,1	1,1	671,2	677,5	737,9	849,0
0,50	2,1	1,2	773,8	773,8	760,7	791,8
0,50	2,1	1,3	773,8	773,8	760,7	791,8
0,75	2,1	1,1	550,8	559,9	559,9	559,9
0,75	2,1	1,2	547,7	570,8	558,6	559,9
0,75	2,1	1,3	564,7	559,9	558,6	559,9
1,00	2,1	1,1	484,7	484,7	563,2	542,2
1,00	2,1	1,2	484,7	484,7	561,8	564,5
1,00	2,1	1,3	500,5	500,5	561,8	564,5
Total			30.227,8	30.812,2	30.091,1	30.469,5

Tabela 11 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 6 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	430,7	430,7	430,7	409,1
0,25	1,5	1,2	416,7	416,7	416,7	416,7
0,25	1,5	1,3	416,7	416,7	416,7	416,7
0,50	1,5	1,1	851,8	881,2	861,4	906,3
0,50	1,5	1,2	846,6	899,5	834,1	883,7
0,50	1,5	1,3	872,2	907,8	892,1	889,7
0,75	1,5	1,1	1.664,7	1.664,7	1.669,5	1.794,7
0,75	1,5	1,2	1.584,1	1.611,5	1.669,5	1.834,2
0,75	1,5	1,3	1.728,9	1.728,9	1.679,0	1.834,2
1,00	1,5	1,1	1.528,8	1.581,7	1.590,3	1.620,6
1,00	1,5	1,2	1.530,8	1.583,0	1.597,2	1.620,6
1,00	1,5	1,3	1.530,8	1.583,0	1.597,2	1.620,6
0,25	1,8	1,1	542,6	542,6	517,7	542,6
0,25	1,8	1,2	466,4	466,4	461,0	473,4
0,25	1,8	1,3	473,4	473,4	461,0	473,4
0,50	1,8	1,1	1.168,2	1.168,2	1.236,8	1.234,0
0,50	1,8	1,2	1.359,0	1.359,0	1.323,1	1.296,8
0,50	1,8	1,3	1.199,5	1.185,8	1.317,2	1.296,8
0,75	1,8	1,1	1.050,5	1.050,5	1.049,7	1.037,0
0,75	1,8	1,2	1.023,8	1.023,8	970,9	1.071,7
0,75	1,8	1,3	1.038,5	1.038,5	968,9	1.084,2
1,00	1,8	1,1	1.174,4	1.100,5	1.208,4	1.148,3
1,00	1,8	1,2	1.183,6	1.094,1	1.166,2	1.156,0
1,00	1,8	1,3	1.183,6	1.100,5	1.166,2	1.156,0
0,25	2,1	1,1	607,0	607,0	599,1	587,7
0,25	2,1	1,2	624,5	624,5	620,7	597,9
0,25	2,1	1,3	597,9	597,9	620,7	597,9
0,50	2,1	1,1	848,9	923,6	910,8	892,4
0,50	2,1	1,2	888,4	888,4	901,1	898,4
0,50	2,1	1,3	839,8	839,8	828,5	821,5
0,75	2,1	1,1	809,8	809,8	830,4	826,0
0,75	2,1	1,2	750,8	752,2	740,6	826,0
0,75	2,1	1,3	750,8	752,2	740,6	826,0
1,00	2,1	1,1	1.487,7	1.578,3	1.512,5	1.488,7
1,00	2,1	1,2	1.483,0	1.522,3	1.512,5	1.499,1
1,00	2,1	1,3	1.483,0	1.522,3	1.512,5	1.499,1
Total			36.437,4	36.726,7	36.831,2	37.577,8

Tabela 12 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 90 atividades e 8 recursos.

RF	NC	DF	GCRR	RPW	MLFT	MAS
0,25	1,5	1,1	778,2	778,2	753,9	753,9
0,25	1,5	1,2	783,7	783,7	783,7	902,9
0,25	1,5	1,3	748,2	690,4	746,9	748,2
0,50	1,5	1,1	1.180,2	1.081,7	1.122,4	1.146,0
0,50	1,5	1,2	1.201,5	1.081,7	1.132,6	1.146,0
0,50	1,5	1,3	1.138,4	1.114,1	1.132,6	1.161,5
0,75	1,5	1,1	1.318,2	1.318,2	1.420,3	1.400,8
0,75	1,5	1,2	1.318,2	1.318,2	1.434,3	1.430,3
0,75	1,5	1,3	1.342,0	1.342,0	1.397,7	1.430,3
1,00	1,5	1,1	2.515,4	2.538,0	2.564,4	2.473,3
1,00	1,5	1,2	2.560,9	2.695,5	2.593,8	2.518,5
1,00	1,5	1,3	2.460,4	2.695,5	2.593,8	2.518,5
0,25	1,8	1,1	828,8	828,8	827,4	846,6
0,25	1,8	1,2	839,9	864,7	853,3	839,9
0,25	1,8	1,3	851,8	851,8	872,0	857,6
0,50	1,8	1,1	1.494,3	1.402,7	1.478,6	1.469,0
0,50	1,8	1,2	1.436,2	1.442,1	1.494,8	1.529,8
0,50	1,8	1,3	1.436,2	1.438,6	1.520,0	1.520,0
0,75	1,8	1,1	1.532,4	1.526,8	1.462,5	1.610,6
0,75	1,8	1,2	1.484,7	1.484,7	1.421,5	1.483,8
0,75	1,8	1,3	1.500,3	1.500,3	1.427,5	1.496,2
1,00	1,8	1,1	2.154,2	2.176,0	2.110,3	2.338,5
1,00	1,8	1,2	2.205,7	2.153,8	2.257,6	2.338,5
1,00	1,8	1,3	2.205,7	2.144,0	2.257,6	2.338,5
0,25	2,1	1,1	969,3	969,3	887,6	887,6
0,25	2,1	1,2	969,3	969,3	857,7	857,7
0,25	2,1	1,3	969,3	969,3	857,7	857,7
0,50	2,1	1,1	1.071,1	1.071,1	1.142,2	1.062,1
0,50	2,1	1,2	1.087,7	1.087,7	1.047,2	954,1
0,50	2,1	1,3	1.013,8	1.013,8	1.054,3	954,1
0,75	2,1	1,1	819,9	883,0	883,0	813,1
0,75	2,1	1,2	819,9	883,0	883,0	813,1
0,75	2,1	1,3	828,9	820,2	806,0	823,4
1,00	2,1	1,1	2.484,2	2.482,9	2.513,5	2.545,0
1,00	2,1	1,2	2.484,2	2.482,9	2.530,4	2.338,3
1,00	2,1	1,3	2.528,3	2.530,9	2.528,4	2.334,2
Total			51.361,1	51.414,5	51.650,1	51.539,5

Tabela 13 – Resultados da comparação entre as regras de prioridade para a heurística construtiva determinística – instâncias de teste com 120 atividades e DF=1,2.

Nº Recursos	RF	NC	GCRR	RPW	MLFT	MAS
4	0,25	1,5	266,6	306,1	371,7	340,1
4	0,50	1,5	422,2	422,2	413,9	401,5
4	0,75	1,5	1.055,0	1.078,0	894,7	1.008,6
4	1,00	1,5	1.413,2	1.385,0	1.382,2	1.393,4
4	0,25	1,8	1.296,4	1.205,2	1.317,8	1.275,9
4	0,50	1,8	578,5	578,5	623,4	592,1
4	0,75	1,8	737,2	737,2	727,5	646,9
4	1,00	1,8	1.140,2	1.140,2	1.100,9	1.147,2
4	0,25	2,1	471,8	457,5	490,6	499,2
4	0,50	2,1	1.657,2	1.699,8	1.649,6	1.593,5
4	0,75	2,1	288,4	288,4	279,0	279,0
4	1,00	2,1	338,3	319,0	328,2	310,9
6	0,25	1,5	464,1	461,9	475,3	492,6
6	0,50	1,5	949,8	949,8	997,1	947,6
6	0,75	1,5	2.015,8	2.248,3	2.045,0	2.147,1
6	1,00	1,5	2.504,3	2.444,5	2.499,1	2.686,9
6	0,25	1,8	845,6	845,6	813,2	784,7
6	0,50	1,8	1.288,4	1.288,4	1.313,0	1.299,2
6	0,75	1,8	1.327,3	1.376,7	1.397,7	1.398,8
6	1,00	1,8	1.416,3	1.485,8	1.361,3	1.499,6
6	0,25	2,1	775,7	775,7	736,9	747,2
6	0,50	2,1	911,8	946,8	963,4	983,3
6	0,75	2,1	1.008,2	1.008,2	1.007,8	960,7
6	1,00	2,1	1.873,4	1.929,2	1.842,7	2.015,7
8	0,25	1,5	901,8	901,8	870,8	902,6
8	0,50	1,5	1.178,5	1.230,4	1.382,2	1.308,6
8	0,75	1,5	2.405,9	2.358,6	2.227,4	2.579,1
8	1,00	1,5	2.955,3	3.008,1	3.141,6	2.833,2
8	0,25	1,8	860,6	868,1	845,0	845,0
8	0,50	1,8	1.969,4	1.881,6	1.918,1	1.900,7
8	0,75	1,8	1.885,8	1.859,2	1.756,3	1.992,8
8	1,00	1,8	2.143,1	2.172,4	2.156,9	2.340,2
8	0,25	2,1	1.114,3	1.114,3	1.199,8	1.139,0
8	0,50	2,1	1.160,0	1.160,0	1.161,8	1.120,8
8	0,75	2,1	895,8	892,1	923,0	913,3
8	1,00	2,1	2.730,5	2.523,6	2.472,5	2.380,6
Total			45.246,6	45.347,9	45.086,9	45.707,7

Tabela 14 – Resultados da comparação entre as regras de prioridade para a heurística construtiva probabilística – instâncias de teste com 30 atividades e 8 recursos (900 segundos).

RF	NC	DF	GCRR	RPW	MLFT	MAS	Combina- do	RAN
0,25	1,5	1,1	384,3	384,3	382,2	382,2	384,3	388,4
0,25	1,5	1,2	348,6	348,6	348,6	348,6	348,6	348,6
0,25	1,5	1,3	348,6	348,6	348,6	348,6	348,6	348,6
0,50	1,5	1,1	500,4	503,4	502,1	506,3	502,7	507,1
0,50	1,5	1,2	470,3	464,9	467,3	466,0	461,0	465,4
0,50	1,5	1,3	431,8	432,8	431,6	432,3	432,0	428,3
0,75	1,5	1,1	744,4	739,5	743,9	743,5	744,7	746,7
0,75	1,5	1,2	692,2	700,2	696,2	695,1	693,8	707,1
0,75	1,5	1,3	654,9	664,6	661,2	660,3	662,6	668,4
1,00	1,5	1,1	866,0	872,4	873,2	877,7	868,8	877,5
1,00	1,5	1,2	818,6	818,3	819,3	818,7	818,7	817,7
1,00	1,5	1,3	749,6	750,8	758,7	755,8	755,5	765,7
0,25	1,8	1,1	379,8	379,8	379,8	379,8	379,8	379,8
0,25	1,8	1,2	374,6	370,2	369,7	370,8	369,5	367,6
0,25	1,8	1,3	351,2	351,2	351,2	351,2	351,2	351,2
0,50	1,8	1,1	604,3	609,0	618,6	605,6	614,7	608,5
0,50	1,8	1,2	557,2	557,8	561,4	560,5	560,2	548,1
0,50	1,8	1,3	502,7	501,1	504,2	501,1	501,1	508,5
0,75	1,8	1,1	757,3	761,2	769,0	762,0	767,5	757,3
0,75	1,8	1,2	713,3	716,5	713,3	717,1	713,0	713,0
0,75	1,8	1,3	679,5	679,9	677,0	682,9	678,3	673,5
1,00	1,8	1,1	847,1	846,7	849,5	854,0	845,1	845,8
1,00	1,8	1,2	798,6	793,3	794,4	791,2	802,1	788,5
1,00	1,8	1,3	746,9	746,0	743,8	751,0	747,1	744,4
0,25	2,1	1,1	421,7	421,7	421,7	421,7	421,7	421,7
0,25	2,1	1,2	404,2	404,2	404,2	404,2	404,2	404,2
0,25	2,1	1,3	389,9	388,0	389,9	389,9	389,0	388,0
0,50	2,1	1,1	588,9	588,9	595,6	588,9	588,9	603,5
0,50	2,1	1,2	548,2	548,9	547,7	547,0	548,2	547,7
0,50	2,1	1,3	506,8	508,1	512,1	508,7	508,5	520,8
0,75	2,1	1,1	440,6	439,9	441,3	441,4	440,6	442,7
0,75	2,1	1,2	410,2	416,1	411,4	411,2	410,8	410,8
0,75	2,1	1,3	384,7	381,3	383,5	389,3	383,6	380,8
1,00	2,1	1,1	1.128,4	1.124,8	1.123,5	1.143,1	1.126,0	1.130,7
1,00	2,1	1,2	1.069,5	1.063,5	1.067,1	1.069,4	1.063,5	1.074,0
1,00	2,1	1,3	1.001,0	1.013,1	997,9	1.011,2	1.008,7	1.014,0
Total			21.616,1	21.639,5	21.660,6	21.688,1	21.644,6	21.694,4

Tabela 16 – Resultados da comparação entre as heurísticas e o CPLEX (30 atividades e 6 recursos).

RF	NC	DF	HCD-GCRR			HCP-GCRR (900 s)			CPLEX (1.800 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	237,0	9,7%	10,8%	213,9	0,0%	0,0%	213,9	213,9	0,0%
0,25	1,5	1,2	237,0	9,7%	10,8%	213,9	0,0%	0,0%	213,9	213,9	0,0%
0,25	1,5	1,3	214,2	0,1%	0,1%	213,9	0,0%	0,0%	213,9	213,9	0,0%
0,50	1,5	1,1	502,6	30,5%	23,7%	408,1	14,3%	0,4%	349,5	406,3	14,0%
0,50	1,5	1,2	461,3	32,2%	19,9%	382,0	18,1%	-0,7%	312,7	384,7	18,7%
0,50	1,5	1,3	431,1	31,0%	19,8%	355,5	16,3%	-1,2%	297,5	359,9	17,3%
0,75	1,5	1,1	841,4	33,3%	16,0%	700,2	19,9%	-3,4%	561,2	725,1	22,6%
0,75	1,5	1,2	796,7	38,9%	17,8%	656,8	25,8%	-2,9%	487,1	676,5	28,0%
0,75	1,5	1,3	761,0	40,0%	14,1%	632,3	27,8%	-5,2%	456,4	666,7	31,6%
1,00	1,5	1,1	835,2	43,3%	9,9%	726,1	34,8%	-4,4%	473,5	759,7	37,7%
1,00	1,5	1,2	801,8	45,2%	13,2%	693,8	36,6%	-2,1%	439,6	708,3	37,9%
1,00	1,5	1,3	760,1	47,6%	8,0%	647,1	38,5%	-8,1%	398,2	703,8	43,4%
0,25	1,8	1,1	419,0	12,8%	14,7%	365,2	0,0%	0,0%	365,2	365,2	0,0%
0,25	1,8	1,2	394,1	12,0%	13,6%	346,8	0,0%	0,0%	346,8	346,8	0,0%
0,25	1,8	1,3	442,5	25,0%	33,3%	331,8	0,0%	0,0%	331,8	331,8	0,0%
0,50	1,8	1,1	673,0	16,4%	19,7%	562,4	0,0%	0,0%	562,4	562,4	0,0%
0,50	1,8	1,2	634,0	19,1%	23,6%	513,0	0,0%	0,0%	512,9	513,0	0,0%
0,50	1,8	1,3	580,4	15,4%	18,3%	495,0	0,9%	0,9%	490,7	490,8	0,0%
0,75	1,8	1,1	638,4	38,6%	10,0%	553,7	29,2%	-4,6%	392,0	580,2	32,4%
0,75	1,8	1,2	620,2	42,1%	17,8%	517,0	30,6%	-1,8%	358,8	526,5	31,9%
0,75	1,8	1,3	575,8	44,1%	10,9%	480,4	33,0%	-7,5%	321,7	519,3	38,1%
1,00	1,8	1,1	678,0	42,8%	18,0%	555,5	30,1%	-3,3%	388,1	574,6	32,5%
1,00	1,8	1,2	664,4	43,1%	25,2%	522,7	27,7%	-1,5%	377,9	530,8	28,8%
1,00	1,8	1,3	641,7	46,2%	26,5%	494,5	30,1%	-2,5%	345,5	507,3	31,9%
0,25	2,1	1,1	410,8	32,8%	48,8%	276,2	0,0%	0,0%	276,2	276,2	0,0%
0,25	2,1	1,2	356,2	26,1%	35,3%	263,4	0,0%	0,0%	263,4	263,4	0,0%
0,25	2,1	1,3	309,4	14,9%	17,5%	263,4	0,0%	0,0%	263,4	263,4	0,0%
0,50	2,1	1,1	476,9	11,7%	13,3%	421,1	0,0%	0,0%	421,0	421,0	0,0%
0,50	2,1	1,2	476,9	19,4%	24,1%	384,3	0,0%	0,0%	384,2	384,2	0,0%
0,50	2,1	1,3	476,9	24,3%	32,0%	362,5	0,3%	0,3%	361,2	361,2	0,0%
0,75	2,1	1,1	437,8	25,6%	15,0%	372,9	12,7%	-2,0%	325,5	380,6	14,5%
0,75	2,1	1,2	410,6	28,2%	17,4%	349,6	15,7%	0,0%	294,7	349,6	15,7%
0,75	2,1	1,3	395,5	36,9%	23,3%	319,5	21,9%	-0,4%	249,7	320,8	22,2%
1,00	2,1	1,1	723,0	28,4%	9,9%	649,6	20,3%	-1,3%	517,6	658,0	21,3%
1,00	2,1	1,2	674,8	29,2%	11,1%	593,1	19,5%	-2,3%	477,4	607,3	21,4%
1,00	2,1	1,3	665,8	29,0%	22,1%	551,4	14,2%	1,1%	473,1	545,3	13,2%
Total			19.655,9	31,2%	17,6%	16.388,5	17,5%	-1,9%	13.518,3	16.712,5	19,1%
Tempo médio execução (s)			0,106			765			1.228		
Número médio iterações						81.587					

Tabela 17 – Resultados da comparação entre as heurísticas e o CPLEX (30 atividades e 8 recursos).

RF	NC	DF	HCD-GCRR			HCP-GCRR (900 s)			CPLEX (1.800 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	421,5	9,3%	10,3%	384,3	0,5%	0,5%	382,2	382,2	0,0%
0,25	1,5	1,2	411,5	15,3%	18,1%	348,6	0,0%	0,0%	348,6	348,6	0,0%
0,25	1,5	1,3	472,3	26,2%	35,5%	348,6	0,0%	0,0%	348,6	348,6	0,0%
0,50	1,5	1,1	584,8	35,1%	14,3%	500,4	24,1%	-2,2%	379,6	511,8	25,8%
0,50	1,5	1,2	521,1	35,1%	12,6%	470,3	28,1%	1,6%	338,0	463,0	27,0%
0,50	1,5	1,3	522,1	40,0%	21,9%	431,8	27,4%	0,8%	313,3	428,2	26,8%
0,75	1,5	1,1	805,6	31,4%	6,8%	744,4	25,7%	-1,3%	552,9	754,3	26,7%
0,75	1,5	1,2	757,8	32,9%	6,9%	692,2	26,5%	-2,4%	508,8	709,0	28,2%
0,75	1,5	1,3	738,7	40,0%	10,8%	654,9	32,3%	-1,8%	443,5	666,6	33,5%
1,00	1,5	1,1	947,0	32,1%	4,5%	866,0	25,8%	-4,5%	642,9	906,7	29,1%
1,00	1,5	1,2	945,1	37,9%	13,7%	818,6	28,3%	-1,5%	586,9	830,9	29,4%
1,00	1,5	1,3	879,5	46,3%	10,5%	749,6	37,0%	-5,8%	472,3	796,0	40,7%
0,25	1,8	1,1	465,2	18,4%	22,5%	379,8	0,0%	0,0%	379,8	379,8	0,0%
0,25	1,8	1,2	462,9	20,6%	25,9%	374,6	1,9%	1,9%	367,6	367,6	0,0%
0,25	1,8	1,3	452,2	23,2%	30,3%	351,2	1,2%	1,2%	347,1	347,1	0,0%
0,50	1,8	1,1	788,6	30,6%	31,2%	604,3	9,5%	0,5%	547,1	601,0	9,0%
0,50	1,8	1,2	788,6	36,5%	45,8%	557,2	10,2%	3,0%	500,6	540,8	7,4%
0,50	1,8	1,3	764,4	34,5%	52,6%	502,7	0,3%	0,3%	501,0	501,1	0,0%
0,75	1,8	1,1	894,8	28,9%	14,9%	757,3	15,9%	-2,8%	636,5	779,1	18,3%
0,75	1,8	1,2	830,5	29,8%	11,6%	713,3	18,3%	-4,2%	582,8	744,4	21,7%
0,75	1,8	1,3	932,4	46,7%	36,6%	679,5	26,9%	-0,5%	497,0	682,5	27,2%
1,00	1,8	1,1	958,8	37,7%	13,4%	847,1	29,5%	0,2%	597,0	845,8	29,4%
1,00	1,8	1,2	924,9	39,8%	13,4%	798,6	30,2%	-2,1%	557,2	815,6	31,7%
1,00	1,8	1,3	924,9	43,1%	21,9%	746,9	29,6%	-1,6%	526,0	758,9	30,7%
0,25	2,1	1,1	545,7	22,7%	29,4%	421,7	0,0%	0,0%	421,7	421,7	0,0%
0,25	2,1	1,2	477,3	15,3%	18,1%	404,2	0,0%	0,0%	404,2	404,2	0,0%
0,25	2,1	1,3	435,8	11,0%	12,3%	389,9	0,5%	0,5%	388,0	388,0	0,0%
0,50	2,1	1,1	646,2	13,1%	9,9%	588,9	4,6%	0,1%	561,8	588,3	4,5%
0,50	2,1	1,2	609,2	17,2%	11,7%	548,2	8,0%	0,5%	504,7	545,6	7,5%
0,50	2,1	1,3	600,0	26,6%	18,4%	506,8	13,1%	0,0%	440,7	506,8	13,1%
0,75	2,1	1,1	454,5	24,2%	0,1%	440,6	21,8%	-2,9%	344,4	453,8	24,1%
0,75	2,1	1,2	506,4	35,7%	21,9%	410,2	20,6%	-1,3%	325,6	415,5	21,6%
0,75	2,1	1,3	527,4	45,9%	34,0%	384,7	25,8%	-2,2%	285,3	393,5	27,5%
1,00	2,1	1,1	1.303,4	34,6%	14,7%	1.128,4	24,4%	-0,7%	852,7	1.136,1	24,9%
1,00	2,1	1,2	1.278,4	37,6%	19,2%	1.069,5	25,4%	-0,3%	797,6	1.072,2	25,6%
1,00	2,1	1,3	1.278,4	46,0%	27,9%	1.001,0	31,0%	0,1%	690,4	999,6	30,9%
Total			25.857,8	32,8%	18,4%	21.616,1	19,6%	-1,0%	17.374,1	21.835,0	20,4%
Tempo médio execução (s)			0,106			853			1.317		
Número médio iterações						72.876					

Tabela 18 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 4 recursos).

RF	NC	DF	HCD-MAS			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	167,7	21%	27%	131,7	0%	0%	131,6	131,6	0%
0,25	1,5	1,2	167,7	21%	27%	131,6	0%	0%	131,6	131,6	0%
0,25	1,5	1,3	167,7	21%	27%	131,6	0%	0%	131,6	131,6	0%
0,50	1,5	1,1	477,8	44%	15%	399,1	34%	-4%	265,2	416,7	36%
0,50	1,5	1,2	457,0	53%	16%	369,8	42%	-6%	213,9	393,4	46%
0,50	1,5	1,3	499,0	57%	35%	343,7	38%	-7%	213,0	370,6	43%
0,75	1,5	1,1	1.112,4	74%	41%	658,0	55%	-17%	294,6	790,3	63%
0,75	1,5	1,2	1.069,9	76%	58%	622,8	58%	-8%	259,5	678,0	62%
0,75	1,5	1,3	942,2	73%	49%	582,1	56%	-8%	257,1	632,7	59%
1,00	1,5	1,1	1.228,6	68%	17%	874,4	55%	-17%	393,2	1.047,6	62%
1,00	1,5	1,2	1.242,1	75%	33%	825,1	63%	-12%	304,5	934,9	67%
1,00	1,5	1,3	1.242,1	76%	39%	777,0	61%	-13%	299,5	892,0	66%
0,25	1,8	1,1	284,5	31%	46%	199,3	2%	2%	195,4	195,4	0%
0,25	1,8	1,2	272,2	33%	48%	183,4	0%	0%	183,4	183,4	0%
0,25	1,8	1,3	272,2	33%	48%	183,4	0%	0%	183,4	183,4	0%
0,50	1,8	1,1	576,5	48%	30%	435,4	31%	-2%	300,0	443,6	32%
0,50	1,8	1,2	580,7	55%	40%	397,8	34%	-4%	262,1	415,2	37%
0,50	1,8	1,3	582,8	56%	58%	363,6	30%	-1%	253,6	368,4	31%
0,75	1,8	1,1	913,9	55%	12%	736,9	44%	-10%	413,0	818,3	50%
0,75	1,8	1,2	913,9	60%	23%	677,6	46%	-9%	364,6	745,2	51%
0,75	1,8	1,3	913,9	65%	33%	627,7	49%	-9%	320,5	689,5	54%
1,00	1,8	1,1	1.043,6	63%	11%	812,8	52%	-14%	391,1	941,8	58%
1,00	1,8	1,2	1.043,6	69%	19%	767,7	58%	-12%	319,3	874,5	63%
1,00	1,8	1,3	1.053,5	73%	28%	728,5	61%	-11%	283,2	821,7	66%
0,25	2,1	1,1	170,3	16%	20%	142,4	0%	0%	142,4	142,4	0%
0,25	2,1	1,2	167,8	25%	33%	126,3	0%	0%	126,3	126,3	0%
0,25	2,1	1,3	161,6	22%	29%	125,5	0%	0%	125,5	125,5	0%
0,50	2,1	1,1	532,3	40%	30%	395,4	19%	-3%	319,4	409,7	22%
0,50	2,1	1,2	539,0	48%	54%	356,7	21%	2%	282,0	351,1	20%
0,50	2,1	1,3	598,2	56%	75%	331,1	21%	-3%	261,5	341,8	24%
0,75	2,1	1,1	382,5	61%	30%	255,7	41%	-13%	151,0	293,8	49%
0,75	2,1	1,2	382,5	64%	44%	237,7	42%	-11%	138,9	265,8	48%
0,75	2,1	1,3	382,5	64%	62%	218,2	37%	-8%	138,1	235,9	41%
1,00	2,1	1,1	533,8	70%	35%	317,2	49%	-19%	161,0	394,1	59%
1,00	2,1	1,2	550,8	75%	66%	295,8	54%	-11%	135,5	332,5	59%
1,00	2,1	1,3	550,8	75%	78%	275,4	51%	-11%	135,2	308,6	56%
Total			22.177,7	62%	34%	15.038,3	44%	-9%	8.481,8	16.558,9	49%
Tempo médio execução (s)			0,103			1.522			2.702		
Número médio iterações						76.464					

Tabela 19 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 6 recursos).

RF	NC	DF	HCD-GCRR			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	332,1	29%	40%	237,2	0%	0%	237,2	237,2	0%
0,25	1,5	1,2	332,1	31%	45%	228,3	0%	0%	228,3	228,3	0%
0,25	1,5	1,3	329,2	31%	44%	228,3	0%	0%	228,3	228,3	0%
0,50	1,5	1,1	550,5	53%	12%	430,0	40%	-12%	258,3	490,5	47%
0,50	1,5	1,2	568,2	59%	32%	396,5	41%	-8%	233,1	431,2	46%
0,50	1,5	1,3	547,7	57%	38%	367,8	37%	-7%	233,1	397,2	41%
0,75	1,5	1,1	1.050,2	57%	-10%	913,5	51%	-21%	449,1	1.162,2	61%
0,75	1,5	1,2	1.049,5	61%	0%	852,1	52%	-19%	413,0	1.052,2	61%
0,75	1,5	1,3	1.012,2	64%	10%	813,8	56%	-12%	361,6	923,0	61%
1,00	1,5	1,1	1.294,9	68%	18%	961,5	57%	-12%	410,4	1.094,2	62%
1,00	1,5	1,2	1.306,3	74%	23%	902,5	63%	-15%	338,4	1.062,6	68%
1,00	1,5	1,3	1.306,3	74%	37%	832,8	60%	-13%	333,8	953,8	65%
0,25	1,8	1,1	490,7	32%	47%	339,0	1%	1%	334,3	334,3	0%
0,25	1,8	1,2	457,3	32%	48%	309,7	0%	0%	309,7	309,7	0%
0,25	1,8	1,3	457,3	34%	51%	302,7	0%	0%	302,7	302,7	0%
0,50	1,8	1,1	962,8	53%	19%	746,2	39%	-8%	451,8	808,6	44%
0,50	1,8	1,2	925,5	58%	20%	694,5	44%	-10%	389,0	769,1	49%
0,50	1,8	1,3	925,5	58%	41%	636,0	39%	-3%	386,3	657,3	41%
0,75	1,8	1,1	925,5	63%	5%	697,4	51%	-21%	342,6	878,9	61%
0,75	1,8	1,2	895,1	67%	22%	652,1	55%	-11%	294,3	733,5	60%
0,75	1,8	1,3	894,3	68%	32%	613,2	53%	-9%	288,7	675,2	57%
1,00	1,8	1,1	897,5	60%	-3%	795,9	55%	-14%	360,5	920,8	61%
1,00	1,8	1,2	856,9	63%	-9%	753,1	57%	-20%	320,2	945,7	66%
1,00	1,8	1,3	873,9	69%	10%	714,8	62%	-10%	273,9	794,0	66%
0,25	2,1	1,1	431,8	17%	20%	376,6	5%	5%	359,2	359,2	0%
0,25	2,1	1,2	429,9	27%	38%	326,3	4%	4%	312,4	312,4	0%
0,25	2,1	1,3	411,2	32%	46%	281,9	0%	0%	281,4	281,4	0%
0,50	2,1	1,1	615,9	42%	13%	515,3	30%	-6%	359,8	545,8	34%
0,50	2,1	1,2	671,2	56%	37%	471,7	37%	-4%	295,0	491,4	40%
0,50	2,1	1,3	592,4	52%	30%	444,9	36%	-2%	284,5	455,4	38%
0,75	2,1	1,1	596,6	55%	17%	460,3	41%	-9%	270,1	508,4	47%
0,75	2,1	1,2	550,1	58%	24%	418,6	45%	-6%	232,1	443,7	48%
0,75	2,1	1,3	550,1	60%	30%	385,2	44%	-9%	217,5	424,3	49%
1,00	2,1	1,1	1.245,1	66%	10%	977,0	56%	-14%	426,2	1.134,8	62%
1,00	2,1	1,2	1.139,9	67%	15%	912,7	59%	-8%	374,7	991,9	62%
1,00	2,1	1,3	1.158,8	69%	18%	847,6	58%	-14%	358,6	985,8	64%
Total			27.633,9	58%	18%	20.837,1	45%	-11%	11.550,2	23.325,0	50%
Tempo médio execução (s)			0,106			1.658			2.726		
Número médio iterações						61.707					

Tabela 20 – Resultados da comparação entre as heurísticas e o CPLEX (60 atividades e 8 recursos).

RF	NC	DF	HCD-MAS			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	712,4	32%	41%	509,0	5%	1%	484,3	504,7	4%
0,25	1,5	1,2	632,9	32%	48%	446,3	4%	4%	428,2	428,2	0%
0,25	1,5	1,3	563,3	29%	40%	410,8	2%	2%	402,4	402,4	0%
0,50	1,5	1,1	790,6	54%	20%	544,2	34%	-17%	360,4	656,9	45%
0,50	1,5	1,2	773,5	61%	40%	505,1	41%	-8%	298,6	551,0	46%
0,50	1,5	1,3	773,5	62%	56%	480,0	38%	-3%	296,2	496,2	40%
0,75	1,5	1,1	1.244,2	65%	-3%	994,3	56%	-22%	438,1	1.282,4	66%
0,75	1,5	1,2	1.244,2	70%	13%	942,8	60%	-14%	374,1	1.102,2	66%
0,75	1,5	1,3	1.251,8	70%	8%	872,4	57%	-25%	374,1	1.161,0	68%
1,00	1,5	1,1	1.644,0	67%	-14%	1.362,4	60%	-29%	549,6	1.914,4	71%
1,00	1,5	1,2	1.633,1	69%	-4%	1.317,4	62%	-22%	500,5	1.696,4	70%
1,00	1,5	1,3	1.633,1	73%	2%	1.242,2	65%	-23%	433,1	1.602,9	73%
0,25	1,8	1,1	676,6	35%	29%	556,1	21%	6%	437,9	526,4	17%
0,25	1,8	1,2	686,9	43%	53%	492,2	21%	10%	391,1	448,7	13%
0,25	1,8	1,3	675,5	45%	61%	445,7	16%	6%	373,4	418,9	11%
0,50	1,8	1,1	1.040,1	47%	2%	973,4	43%	-5%	554,6	1.019,5	46%
0,50	1,8	1,2	1.026,5	53%	8%	880,1	45%	-7%	484,9	948,5	49%
0,50	1,8	1,3	1.193,5	60%	33%	816,8	41%	-9%	482,6	898,6	46%
0,75	1,8	1,1	1.192,0	60%	11%	917,8	49%	-14%	471,2	1.073,0	56%
0,75	1,8	1,2	1.080,0	65%	9%	847,6	56%	-15%	375,0	992,4	62%
0,75	1,8	1,3	1.213,0	69%	36%	791,7	53%	-11%	373,1	892,6	58%
1,00	1,8	1,1	1.452,2	68%	1%	1.136,8	59%	-21%	462,1	1.442,5	68%
1,00	1,8	1,2	1.427,6	72%	13%	1.071,3	62%	-15%	403,3	1.267,5	68%
1,00	1,8	1,3	1.446,6	73%	14%	1.005,4	61%	-21%	393,1	1.271,2	69%
0,25	2,1	1,1	865,8	33%	31%	648,8	10%	-2%	584,3	660,1	11%
0,25	2,1	1,2	682,1	28%	17%	577,3	15%	-1%	490,3	584,4	16%
0,25	2,1	1,3	669,3	30%	22%	553,3	15%	1%	470,9	548,9	14%
0,50	2,1	1,1	902,0	54%	16%	712,3	42%	-8%	413,9	775,2	47%
0,50	2,1	1,2	902,0	59%	32%	663,7	45%	-3%	367,7	685,8	46%
0,50	2,1	1,3	822,8	56%	34%	618,5	41%	1%	362,4	611,8	41%
0,75	2,1	1,1	696,8	58%	18%	530,2	45%	-10%	294,2	590,5	50%
0,75	2,1	1,2	696,8	66%	26%	478,9	51%	-13%	236,1	551,9	57%
0,75	2,1	1,3	652,6	64%	36%	450,6	48%	-6%	234,5	481,2	51%
1,00	2,1	1,1	1.807,3	67%	17%	1.323,6	55%	-14%	598,4	1.538,3	61%
1,00	2,1	1,2	1.807,3	73%	22%	1.220,7	59%	-17%	495,3	1.476,2	66%
1,00	2,1	1,3	1.807,3	73%	38%	1.118,4	56%	-15%	494,5	1.313,9	62%
Total			38.318,8	60%	17%	28.458,1	47%	-13%	15.184,2	32.816,5	54%
Tempo médio execução (s)			0,103			1.801			3.465		
Número médio iterações						49.112					

Tabela 21 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 4 recursos).

RF	NC	DF	HCD-MLFT			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	320,2	54%	52%	216,1	32%	2%	146,6	210,9	30%
0,25	1,5	1,2	320,2	54%	67%	202,7	28%	6%	146,2	191,3	24%
0,25	1,5	1,3	320,2	54%	75%	185,5	21%	1%	146,2	183,2	20%
0,50	1,5	1,1	794,3	72%	29%	451,8	51%	-26%	222,1	613,4	64%
0,50	1,5	1,2	756,8	73%	52%	424,5	51%	-15%	207,6	498,4	58%
0,50	1,5	1,3	756,8	73%	53%	398,0	48%	-20%	207,6	494,8	58%
0,75	1,5	1,1	1.349,9	75%	18%	772,3	57%	-32%	334,5	1.142,3	71%
0,75	1,5	1,2	1.354,1	81%	28%	716,6	64%	-32%	257,7	1.059,9	76%
0,75	1,5	1,3	1.383,8	81%	48%	662,7	61%	-29%	257,1	932,3	72%
1,00	1,5	1,1	1.529,5	79%	-27%	1.010,2	67%	-52%	328,4	2.105,6	84%
1,00	1,5	1,2	1.556,7	81%	10%	953,6	69%	-33%	299,5	1.415,5	79%
1,00	1,5	1,3	1.556,7	81%	16%	889,9	66%	-34%	299,5	1.346,4	78%
0,25	1,8	1,1	450,0	46%	68%	277,8	12%	3%	245,1	268,5	9%
0,25	1,8	1,2	450,0	52%	86%	252,6	14%	4%	217,3	242,2	10%
0,25	1,8	1,3	450,0	57%	98%	230,9	15%	1%	195,7	227,6	14%
0,50	1,8	1,1	935,2	68%	26%	612,6	52%	-17%	295,3	740,7	60%
0,50	1,8	1,2	935,2	72%	50%	568,0	54%	-9%	261,2	624,9	58%
0,50	1,8	1,3	935,2	72%	49%	523,2	50%	-17%	261,2	627,3	58%
0,75	1,8	1,1	1.204,5	73%	16%	810,2	60%	-22%	326,3	1.036,0	69%
0,75	1,8	1,2	1.211,2	74%	25%	746,3	57%	-23%	319,8	968,0	67%
0,75	1,8	1,3	1.211,2	74%	42%	697,6	54%	-18%	319,8	853,2	63%
1,00	1,8	1,1	1.332,6	78%	16%	783,6	62%	-32%	296,3	1.149,3	74%
1,00	1,8	1,2	1.332,6	79%	18%	718,6	61%	-36%	277,3	1.130,5	75%
1,00	1,8	1,3	1.332,6	79%	36%	660,5	58%	-32%	277,3	977,7	72%
0,25	2,1	1,1	230,3	31%	44%	160,0	0%	0%	160,0	160,0	0%
0,25	2,1	1,2	230,3	36%	54%	149,1	1%	0%	148,1	149,1	1%
0,25	2,1	1,3	227,8	39%	65%	140,0	1%	1%	138,3	138,3	0%
0,50	2,1	1,1	737,9	61%	11%	534,5	46%	-20%	286,6	665,4	57%
0,50	2,1	1,2	760,7	66%	36%	482,4	46%	-14%	260,4	559,0	53%
0,50	2,1	1,3	760,7	66%	51%	444,1	41%	-12%	260,4	504,5	48%
0,75	2,1	1,1	559,9	70%	23%	352,9	53%	-23%	166,2	456,9	64%
0,75	2,1	1,2	558,6	75%	41%	328,3	58%	-17%	138,1	394,9	65%
0,75	2,1	1,3	558,6	75%	52%	297,6	54%	-19%	138,1	367,9	62%
1,00	2,1	1,1	563,2	68%	-10%	396,8	55%	-37%	177,9	626,1	72%
1,00	2,1	1,2	561,8	73%	-12%	362,9	58%	-43%	153,2	638,9	76%
1,00	2,1	1,3	561,8	75%	9%	335,3	59%	-35%	138,1	516,0	73%
Total			30.091,1	72%	24%	17.749,7	53%	-27%	8.310,7	24.216,8	66%
Tempo médio execução (s)			0,100			1.801			3.506		
Número médio iterações						47.312					

Tabela 22 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 6 recursos).

RF	NC	DF	HCD-GCRR			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	430,7	45%	70%	265,7	10%	5%	238,9	253,5	6%
0,25	1,5	1,2	416,7	48%	85%	230,3	5%	2%	217,9	225,7	3%
0,25	1,5	1,3	416,7	48%	90%	221,2	2%	1%	216,3	219,6	2%
0,50	1,5	1,1	851,8	71%	11%	619,1	60%	-19%	246,0	765,7	68%
0,50	1,5	1,2	846,6	71%	27%	577,1	58%	-13%	241,3	664,8	64%
0,50	1,5	1,3	872,2	72%	38%	539,2	55%	-15%	241,3	633,3	62%
0,75	1,5	1,1	1.664,7	77%	-36%	1.195,0	68%	-54%	381,0	2.596,3	85%
0,75	1,5	1,2	1.584,1	77%	-16%	1.131,7	68%	-40%	357,4	1.896,3	81%
0,75	1,5	1,3	1.728,9	79%	-7%	1.066,3	67%	-43%	357,1	1.855,8	81%
1,00	1,5	1,1	1.528,8	78%	-4%	1.119,2	70%	-30%	334,6	1.596,9	79%
1,00	1,5	1,2	1.530,8	78%	-4%	1.048,8	68%	-34%	333,9	1.586,5	79%
1,00	1,5	1,3	1.530,8	78%	-9%	975,4	66%	-42%	333,9	1.682,0	80%
0,25	1,8	1,1	542,6	39%	65%	334,8	2%	2%	329,5	329,5	0%
0,25	1,8	1,2	466,4	30%	43%	325,3	0%	0%	325,3	325,3	0%
0,25	1,8	1,3	473,4	31%	46%	325,3	0%	0%	325,3	325,3	0%
0,50	1,8	1,1	1.168,2	65%	9%	889,9	55%	-17%	403,4	1.067,3	62%
0,50	1,8	1,2	1.359,0	71%	34%	817,0	52%	-20%	395,5	1.015,7	61%
0,50	1,8	1,3	1.199,5	67%	19%	754,0	48%	-25%	395,5	1.005,7	61%
0,75	1,8	1,1	1.050,5	70%	-21%	793,3	60%	-40%	318,9	1.323,0	76%
0,75	1,8	1,2	1.023,8	72%	-14%	727,7	60%	-39%	290,0	1.191,7	76%
0,75	1,8	1,3	1.038,5	72%	6%	674,4	57%	-31%	288,7	979,9	71%
1,00	1,8	1,1	1.174,4	74%	-27%	917,7	67%	-43%	306,1	1.599,3	81%
1,00	1,8	1,2	1.183,6	77%	-7%	860,3	69%	-32%	270,9	1.274,3	79%
1,00	1,8	1,3	1.183,6	77%	11%	803,8	66%	-24%	270,9	1.064,3	75%
0,25	2,1	1,1	607,0	42%	47%	412,1	15%	0%	351,3	413,5	15%
0,25	2,1	1,2	624,5	50%	69%	381,9	19%	3%	310,8	369,2	16%
0,25	2,1	1,3	597,9	48%	76%	362,3	15%	6%	309,1	340,4	9%
0,50	2,1	1,1	848,9	62%	6%	666,8	52%	-17%	321,7	799,6	60%
0,50	2,1	1,2	888,4	67%	24%	612,6	52%	-14%	294,2	716,2	59%
0,50	2,1	1,3	839,8	65%	28%	562,4	48%	-14%	294,2	654,8	55%
0,75	2,1	1,1	809,8	68%	-17%	676,7	62%	-31%	256,3	975,5	74%
0,75	2,1	1,2	750,8	69%	-21%	626,1	63%	-34%	234,1	948,7	75%
0,75	2,1	1,3	750,8	71%	-10%	579,5	62%	-31%	217,5	837,7	74%
1,00	2,1	1,1	1.487,7	70%	-28%	1.220,9	63%	-41%	446,6	2.057,6	78%
1,00	2,1	1,2	1.483,0	74%	-9%	1.126,0	66%	-31%	379,3	1.632,7	77%
1,00	2,1	1,3	1.483,0	76%	-5%	1.059,2	66%	-32%	358,6	1.561,0	77%
Total			36.437,4	69%	-1%	25.499,0	56%	-31%	11.193,4	36.784,2	70%
Tempo médio execução (s)			0,106			1.708			3.320		
Número médio iterações						33.865					

Tabela 23 – Resultados da comparação entre as heurísticas e o CPLEX (90 atividades e 8 recursos).

RF	NC	DF	HCD-GCRR			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
0,25	1,5	1,1	778,2	48%	19%	596,1	32%	-9%	407,9	652,7	38%
0,25	1,5	1,2	783,7	52%	36%	545,2	30%	-5%	379,7	576,6	34%
0,25	1,5	1,3	748,2	49%	48%	504,8	25%	0%	379,7	504,5	25%
0,50	1,5	1,1	1.180,2	74%	18%	752,9	59%	-25%	308,7	998,8	69%
0,50	1,5	1,2	1.201,5	74%	22%	703,5	56%	-29%	308,7	985,8	69%
0,50	1,5	1,3	1.138,4	73%	34%	655,2	53%	-23%	308,7	850,4	64%
0,75	1,5	1,1	1.318,2	71%	-23%	1.000,0	62%	-42%	376,9	1.710,7	78%
0,75	1,5	1,2	1.318,2	71%	-9%	922,3	59%	-36%	376,9	1.448,0	74%
0,75	1,5	1,3	1.342,0	72%	-9%	862,2	56%	-42%	376,9	1.476,2	74%
1,00	1,5	1,1	2.515,4	80%	-31%	1.725,7	70%	-53%	514,0	3.634,8	86%
1,00	1,5	1,2	2.560,9	81%	-31%	1.648,5	71%	-56%	481,0	3.707,9	87%
1,00	1,5	1,3	2.460,4	82%	10%	1.562,0	72%	-30%	431,6	2.228,6	81%
0,25	1,8	1,1	828,8	47%	21%	696,1	37%	2%	436,0	684,3	36%
0,25	1,8	1,2	839,9	55%	37%	629,7	39%	3%	381,3	612,5	38%
0,25	1,8	1,3	851,8	55%	53%	581,3	34%	4%	381,3	558,1	32%
0,50	1,8	1,1	1.494,3	67%	9%	1.051,4	53%	-23%	494,6	1.366,4	64%
0,50	1,8	1,2	1.436,2	66%	22%	967,7	50%	-18%	484,4	1.180,0	59%
0,50	1,8	1,3	1.436,2	66%	36%	910,5	47%	-14%	484,4	1.054,1	54%
0,75	1,8	1,1	1.532,4	73%	-14%	1.143,4	64%	-36%	415,2	1.789,4	77%
0,75	1,8	1,2	1.484,7	74%	-8%	1.058,4	64%	-34%	383,1	1.614,5	76%
0,75	1,8	1,3	1.500,3	74%	8%	969,7	60%	-30%	383,1	1.391,8	72%
1,00	1,8	1,1	2.154,2	78%	-36%	1.564,9	70%	-53%	476,2	3.342,9	86%
1,00	1,8	1,2	2.205,7	81%	8%	1.484,1	72%	-27%	410,9	2.040,0	80%
1,00	1,8	1,3	2.205,7	82%	17%	1.415,1	72%	-25%	398,2	1.878,8	79%
0,25	2,1	1,1	969,3	50%	57%	657,5	26%	6%	488,0	619,0	21%
0,25	2,1	1,2	969,3	54%	65%	590,8	24%	0%	446,3	588,2	24%
0,25	2,1	1,3	969,3	54%	85%	553,1	20%	6%	444,9	523,6	15%
0,50	2,1	1,1	1.071,1	62%	-8%	862,6	53%	-26%	407,1	1.169,9	65%
0,50	2,1	1,2	1.087,7	66%	4%	788,3	53%	-25%	370,9	1.049,6	65%
0,50	2,1	1,3	1.013,8	63%	22%	719,0	48%	-14%	370,9	832,2	55%
0,75	2,1	1,1	819,9	68%	-35%	707,4	63%	-44%	259,4	1.264,1	79%
0,75	2,1	1,2	819,9	70%	-16%	649,5	62%	-33%	243,6	970,9	75%
0,75	2,1	1,3	828,9	71%	-15%	605,2	60%	-38%	243,6	974,3	75%
1,00	2,1	1,1	2.484,2	76%	-33%	1.921,8	69%	-49%	593,5	3.734,3	84%
1,00	2,1	1,2	2.484,2	79%	3%	1.825,6	72%	-24%	513,2	2.409,9	79%
1,00	2,1	1,3	2.528,3	80%	11%	1.726,8	71%	-24%	501,1	2.273,0	78%
Total			51.361,1	71%	-3%	35.558,6	59%	-33%	14.662,0	52.696,7	72%
Tempo médio execução (s)			0,106			1.801			3.607		
Número médio iterações						27.282					

Tabela 24 – Resultados da comparação entre as heurísticas e o CPLEX (7.200 segundos) – instâncias de teste com 90 atividades e DF=1,2.

Nº Rec.	RF	NC	HCD-MLFT/GCRR			HCP-GCRR (1.800 s)			CPLEX (7.200 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
4	0,25	1,5	320,2	54%	64%	202,7	28%	4%	146,2	195,0	25%
4	0,50	1,5	756,8	73%	49%	424,5	51%	-17%	207,6	509,4	59%
4	0,75	1,5	1.354,1	81%	39%	716,6	64%	-27%	257,8	976,3	74%
4	1,00	1,5	1.556,7	81%	25%	953,6	69%	-23%	299,5	1.244,8	76%
4	0,25	1,8	450,0	47%	86%	252,6	5%	4%	239,9	242,2	1%
4	0,50	1,8	935,2	72%	50%	568,0	54%	-9%	261,2	623,7	58%
4	0,75	1,8	1.211,2	74%	36%	746,3	57%	-17%	319,8	893,9	64%
4	1,00	1,8	1.332,6	79%	38%	718,6	61%	-26%	277,3	965,8	71%
4	0,25	2,1	230,3	35%	54%	149,1	0%	0%	149,1	149,1	0%
4	0,50	2,1	760,7	66%	42%	482,4	46%	-10%	260,4	535,5	51%
4	0,75	2,1	558,6	75%	46%	328,3	58%	-14%	138,1	381,9	64%
4	1,00	2,1	561,8	73%	1%	362,9	58%	-35%	154,1	555,0	72%
6	0,25	1,5	416,7	48%	85%	230,3	5%	2%	218,1	225,7	3%
6	0,50	1,5	846,6	71%	26%	577,1	58%	-14%	241,3	670,1	64%
6	0,75	1,5	1.584,1	77%	-10%	1.131,7	68%	-35%	358,1	1.750,9	80%
6	1,00	1,5	1.530,8	78%	4%	1.048,8	68%	-29%	333,9	1.471,5	77%
6	0,25	1,8	466,4	30%	43%	325,3	0%	0%	325,3	325,3	0%
6	0,50	1,8	1.359,0	71%	37%	817,0	52%	-18%	395,5	991,0	60%
6	0,75	1,8	1.023,8	72%	-13%	727,7	60%	-38%	290,0	1.179,4	75%
6	1,00	1,8	1.183,6	77%	-2%	860,3	69%	-29%	270,9	1.212,2	78%
6	0,25	2,1	624,5	49%	72%	381,9	17%	5%	317,5	362,6	12%
6	0,50	2,1	888,4	67%	21%	612,6	52%	-17%	294,2	736,8	60%
6	0,75	2,1	750,8	69%	-15%	626,1	63%	-29%	234,4	879,8	73%
6	1,00	2,1	1.483,0	74%	-2%	1.126,0	66%	-26%	379,3	1.518,1	75%
8	0,25	1,5	783,7	51%	36%	545,2	30%	-5%	380,4	576,5	34%
8	0,50	1,5	1.201,5	74%	27%	703,5	56%	-26%	308,7	949,1	67%
8	0,75	1,5	1.318,2	71%	-6%	922,3	59%	-34%	376,9	1.395,4	73%
8	1,00	1,5	2.560,9	81%	-15%	1.648,5	70%	-46%	488,2	3.027,0	84%
8	0,25	1,8	839,9	55%	39%	629,7	39%	5%	381,3	602,2	37%
8	0,50	1,8	1.436,2	66%	18%	967,7	50%	-20%	484,4	1.214,7	60%
8	0,75	1,8	1.484,7	74%	12%	1.058,4	64%	-20%	383,1	1.320,5	71%
8	1,00	1,8	2.205,7	81%	17%	1.484,1	72%	-22%	414,6	1.893,2	78%
8	0,25	2,1	969,3	54%	65%	590,8	24%	1%	447,1	587,1	24%
8	0,50	2,1	1.087,7	66%	12%	788,3	53%	-19%	370,9	968,5	62%
8	0,75	2,1	819,9	70%	-11%	649,5	62%	-30%	243,6	923,5	74%
8	1,00	2,1	2.484,2	79%	5%	1.825,6	72%	-23%	516,6	2.369,5	78%
Total			39.377,5	72%	14%	26.183,9	57%	-24%	11.165,0	34.423,0	68%
Tempo médio execução (s)			0,106			1.755			6.936		
Número médio iterações						35.863					

Tabela 25 – Resultados da comparação entre as heurísticas e o CPLEX (3.600 segundos) – instâncias de teste com 120 atividades e DF=1,2.

Nº Rec.	RF	NC	HCD-MLFT			HCP-GCRR (1.800 s)			CPLEX (3.600 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
4	0,25	1,5	371,7	63%	170%	137,6	0%	0%	137,5	137,6	0%
4	0,50	1,5	413,9	50%	73%	243,1	15%	2%	207,6	238,7	13%
4	0,75	1,5	894,7	71%	5%	611,2	58%	-28%	257,1	854,3	70%
4	1,00	1,5	1.382,2	78%	-21%	829,3	64%	-53%	299,5	1.756,4	83%
4	0,25	1,8	1.317,8	85%	-17%	779,2	75%	-51%	197,2	1.591,3	88%
4	0,50	1,8	623,4	58%	83%	343,1	24%	1%	261,2	340,3	23%
4	0,75	1,8	727,5	56%	57%	450,2	29%	-3%	319,8	464,0	31%
4	1,00	1,8	1.100,9	75%	39%	553,8	50%	-30%	277,3	793,3	65%
4	0,25	2,1	490,6	74%	-31%	330,4	62%	-53%	126,3	709,4	82%
4	0,50	2,1	1.649,6	84%	-7%	919,8	72%	-48%	260,4	1.779,8	85%
4	0,75	2,1	279,0	51%	81%	167,6	18%	9%	137,7	153,8	10%
4	1,00	2,1	328,2	56%	116%	161,1	11%	6%	143,7	152,2	6%
6	0,25	1,5	475,3	53%	46%	319,4	30%	-2%	224,2	324,9	31%
6	0,50	1,5	997,1	76%	16%	542,4	56%	-37%	241,3	857,9	72%
6	0,75	1,5	2.045,0	83%	-1%	1.296,2	72%	-38%	357,1	2.074,5	83%
6	1,00	1,5	2.499,1	86%	-23%	1.423,8	76%	-56%	338,2	3.240,2	90%
6	0,25	1,8	813,2	59%	56%	490,7	32%	-6%	332,3	521,6	36%
6	0,50	1,8	1.313,0	70%	-18%	839,6	53%	-47%	395,5	1.592,8	75%
6	0,75	1,8	1.397,7	79%	-26%	866,7	67%	-54%	288,7	1.882,2	85%
6	1,00	1,8	1.361,3	80%	-46%	1.021,7	73%	-59%	275,5	2.522,6	89%
6	0,25	2,1	736,9	58%	97%	395,5	21%	6%	311,9	374,6	17%
6	0,50	2,1	963,4	70%	-10%	677,2	57%	-37%	291,3	1.072,0	73%
6	0,75	2,1	1.007,8	79%	-31%	706,3	69%	-52%	215,6	1.467,1	85%
6	1,00	2,1	1.842,7	81%	-22%	1.436,6	75%	-40%	358,6	2.376,8	85%
8	0,25	1,5	870,8	54%	17%	651,2	39%	-13%	397,7	745,8	47%
8	0,50	1,5	1.382,2	77%	-6%	793,4	60%	-46%	315,2	1.464,3	78%
8	0,75	1,5	2.227,4	83%	-40%	1.418,4	73%	-62%	382,2	3.717,0	90%
8	1,00	1,5	3.141,6	86%	-22%	1.722,8	75%	-57%	436,9	4.031,0	89%
8	0,25	1,8	845,0	55%	48%	501,7	24%	-12%	378,9	571,8	34%
8	0,50	1,8	1.918,1	75%	9%	1.091,3	55%	-38%	487,2	1.763,5	72%
8	0,75	1,8	1.756,3	78%	-28%	1.110,9	65%	-55%	393,7	2.451,6	84%
8	1,00	1,8	2.156,9	82%	-33%	1.370,8	71%	-58%	398,2	3.238,1	88%
8	0,25	2,1	1.199,8	63%	69%	703,6	37%	-1%	443,8	709,3	37%
8	0,50	2,1	1.161,8	68%	-49%	789,6	53%	-66%	368,0	2.293,8	84%
8	0,75	2,1	923,0	74%	-35%	718,8	66%	-50%	243,6	1.430,9	83%
8	1,00	2,1	2.472,5	80%	-10%	1.745,9	71%	-36%	501,1	2.739,1	82%
Total			45.086,9	76%	-14%	28.160,9	61%	-46%	11.001,9	52.434,1	79%
Tempo médio execução (s)			0,100			1.801			3.608		
Número médio iterações						24.951					

Tabela 26 – Resultados da comparação entre as heurísticas e o CPLEX (7.200 segundos) – instâncias de teste com 120 atividades e DF=1,2.

Nº Rec.	RF	NC	HCD-MLFT			HCP-GCRR (3.600 s)			CPLEX (7.200 s)		
			FO	Gap LB	Dif. UB	FO	Gap LB	Dif. UB	LB	UB	Gap LB
4	0,25	1,5	371,7	63%	170%	137,6	0%	0%	137,5	137,6	0%
4	0,50	1,5	413,9	50%	80%	244,2	15%	6%	207,6	229,8	10%
4	0,75	1,5	894,7	71%	9%	612,1	58%	-25%	257,1	820,9	69%
4	1,00	1,5	1.382,2	78%	-24%	829,3	64%	-54%	299,5	1.810,0	83%
4	0,25	1,8	1.317,8	85%	-17%	781,2	75%	-51%	198,0	1.591,3	88%
4	0,50	1,8	623,4	58%	83%	340,3	23%	0%	261,2	340,3	23%
4	0,75	1,8	727,5	55%	55%	455,2	29%	-3%	323,9	470,7	31%
4	1,00	1,8	1.100,9	75%	63%	553,7	50%	-18%	277,3	676,3	59%
4	0,25	2,1	490,6	74%	-14%	324,0	61%	-43%	126,3	571,0	78%
4	0,50	2,1	1.649,6	84%	-7%	918,6	72%	-48%	260,4	1.779,8	85%
4	0,75	2,1	279,0	50%	77%	167,0	17%	6%	138,1	157,6	12%
4	1,00	2,1	328,2	56%	116%	161,0	11%	6%	143,7	152,2	6%
6	0,25	1,5	475,3	53%	46%	318,6	30%	-2%	224,2	325,2	31%
6	0,50	1,5	997,1	76%	24%	548,4	56%	-32%	241,3	800,9	70%
6	0,75	1,5	2.045,0	83%	-1%	1.292,6	72%	-37%	357,2	2.056,6	83%
6	1,00	1,5	2.499,1	86%	-23%	1.422,4	76%	-56%	343,3	3.240,2	89%
6	0,25	1,8	813,2	59%	63%	485,3	32%	-3%	332,3	499,9	34%
6	0,50	1,8	1.313,0	70%	6%	829,3	52%	-33%	395,5	1.235,8	68%
6	0,75	1,8	1.397,7	79%	-19%	867,0	67%	-50%	288,7	1.721,9	83%
6	1,00	1,8	1.361,3	80%	-14%	1.017,0	73%	-36%	276,3	1.585,0	83%
6	0,25	2,1	736,9	58%	94%	386,2	19%	2%	311,9	379,5	18%
6	0,50	2,1	963,4	70%	15%	674,6	57%	-20%	291,3	838,1	65%
6	0,75	2,1	1.007,8	79%	-31%	698,3	69%	-52%	215,6	1.461,6	85%
6	1,00	2,1	1.842,7	81%	-25%	1.428,4	75%	-41%	358,6	2.441,5	85%
8	0,25	1,5	870,8	54%	26%	647,4	39%	-6%	397,7	690,2	42%
8	0,50	1,5	1.382,2	77%	-4%	790,8	60%	-45%	315,2	1.433,1	78%
8	0,75	1,5	2.227,4	83%	-40%	1.416,4	73%	-62%	382,2	3.707,5	90%
8	1,00	1,5	3.141,6	86%	-5%	1.722,1	75%	-48%	436,9	3.299,5	87%
8	0,25	1,8	845,0	55%	74%	494,0	23%	1%	378,9	486,9	22%
8	0,50	1,8	1.918,1	75%	23%	1.065,9	54%	-32%	487,2	1.559,6	69%
8	0,75	1,8	1.756,3	77%	-27%	1.116,6	65%	-53%	396,0	2.394,1	83%
8	1,00	1,8	2.156,9	82%	-16%	1.363,9	71%	-47%	398,2	2.570,6	85%
8	0,25	2,1	1.199,8	63%	66%	687,6	35%	-5%	443,8	722,4	39%
8	0,50	2,1	1.161,8	68%	-10%	776,7	53%	-40%	368,0	1.292,3	72%
8	0,75	2,1	923,0	74%	-31%	714,3	66%	-47%	243,6	1.337,6	82%
8	1,00	2,1	2.472,5	80%	-8%	1.731,5	71%	-36%	501,1	2.701,6	81%
Total			45.086,9	76%	-5%	28.019,0	61%	-41%	11.015,4	47.518,9	77%
Tempo médio execução (s)			0,100			3.601			7.209		
Número médio iterações						42.895					

5 DISCUSSÃO

O objetivo da primeira etapa dos testes computacionais era encontrar a melhor combinação de regra de prioridade e de regra de desempate para as heurísticas construtivas determinística e probabilística. A Tabela 27 apresenta um resumo dos resultados obtidos pela heurística construtiva determinística, sumarizados pelo número de atividades e pelo número de recursos. Os valores em azul são os melhores obtidos para o respectivo conjunto de instâncias.

Tabela 27 – Resumo da comparação entre as regras de prioridade para a heurística construtiva determinística.

Nº Atividades	Nº Recursos	DF	GCRR	RPW	MLFT	MAS
30	4	Todos	13.427,2	13.494,3	13.465,7	13.557,8
30	6	Todos	19.655,9	19.852,9	20.340,8	20.403,3
30	8	Todos	25.857,8	26.315,2	25.974,0	26.534,4
60	4	Todos	22.704,7	22.191,6	22.249,1	22.177,7
60	6	Todos	27.633,9	28.060,8	28.200,4	27.921,3
60	8	Todos	38.662,2	39.038,8	38.405,1	38.318,8
90	4	Todos	30.227,8	30.812,2	30.091,1	30.469,5
90	6	Todos	36.437,4	36.726,7	36.831,2	37.577,8
90	8	Todos	51.361,1	51.414,5	51.650,1	51.539,5
120	Todos	1,2	45.246,6	45.347,9	45.086,9	45.707,7
Total			311.214,5	313.254,9	312.294,5	314.207,5
Tempo total execução (s)			38	38	36	37
Tempo médio execução (s)			0,106	0,106	0,100	0,103

A regra de prioridade GCRR (maior necessidade de recurso acumulada) combinada com a regra de desempate MLFT (mínimo instante mais tarde de início) gerou o melhor resultado em 6 dos 10 grupos e também o melhor resultado total (considerando as 360 instâncias de testes). A regra de prioridade MLFT combinada com a regra de desempate MAS (menor folga total) ficou em segundo lugar, gerando o melhor resultado em 2 dos 10 grupos e o segundo melhor resultado total (0,3% pior do que GCRR + MLFT). O tempo computacional foi muito baixo, variando entre 36 segundos (0,100 segundos por instância) e 38 segundos (0,106 segundos por instância) para as 360 instâncias de testes.

Para a heurística construtiva probabilística, as seis combinações de regra de prioridade e regra de desempate testadas geraram resultados muito próximos, com a pior regra (RAN – seleção aleatória) gerando resultado apenas 0,4% pior do que a melhor. A combinação GCRR + MLFT foi considerada a melhor, por gerar o menor resultado total, o maior número de melhores resultados (em 16 das 36 instâncias) e os melhores resultados na análise por fator (fator de recursos (RF), complexidade da rede (NC) e fator de data de entrega (DF)), conforme mostra a Tabela 28. Mesmo quando a regra GCRR + MLFT não apresenta o melhor resultado (por exemplo, quando RF=0,25), o valor gerado é muito próximo do melhor valor. Dessa forma, a combinação de regra de prioridade GCRR e regra de desempate MLFT foi escolhida e será utilizada na heurística construtiva probabilística na segunda etapa de testes.

Tabela 28 – Resumo da comparação entre as regras de prioridade para a heurística construtiva probabilística (instâncias com 30 atividades e 8 recursos).

		GCRR	RPW	MLFT	MAS	Combinado	RAN
RF	0,25	3.402,9	3.396,6	3.395,9	3.397,0	3.396,8	3.398,1
	0,50	4.710,7	4.714,8	4.740,6	4.716,3	4.717,3	4.737,9
	0,75	5.476,9	5.499,3	5.496,8	5.502,8	5.494,9	5.500,2
	1,00	8.025,7	8.028,8	8.027,3	8.072,0	8.035,6	8.058,2
NC	1,5	7.009,6	7.028,4	7.032,8	7.035,0	7.021,3	7.069,4
	1,8	7.312,5	7.312,7	7.332,0	7.327,2	7.329,6	7.286,2
	2,1	7.294,0	7.298,4	7.295,8	7.326,0	7.293,6	7.338,8
DF	1,1	7.663,2	7.671,5	7.700,4	7.706,2	7.684,8	7.709,5
	1,2	7.205,5	7.202,5	7.200,5	7.199,7	7.193,7	7.192,8
	1,3	6.747,4	6.765,4	6.759,8	6.782,3	6.766,1	6.792,1
Total		21.616,1	21.639,5	21.660,6	21.688,1	21.644,6	21.694,4
Tempo médio execução (s)		853	846	852	854	850	686
Número médio iterações		72.876	70.531	72.709	71.596	70.638	146.090

Com relação ao tempo computacional e ao número de iterações, as cinco combinações de regras de prioridade e regras de desempate apresentaram valores muito similares. O tempo médio de execução por instância variou entre 846 e 854 segundos, inferior a 900 segundos devido ao critério de parada para as instâncias fáceis (com baixa demanda por recursos). O número médio de iterações por instância de teste variou entre 70.531 e 72.876, de modo que a maior parte das execuções terminou na terceira rodada dos ciclos (entre 60.000 e 85.000 iterações). Já a sexta regra testada, a seleção aleatória (RAN), foi a exceção, com tempo médio de 686

segundos e número médio de 146.090 iterações (sexta rodada dos ciclos, entre 135.000 e 160.000 iterações), por não possuir as etapas de cálculo da flexibilidade da atividade e da flexibilidade do recurso, o que torna sua execução muito mais rápida.

Na segunda etapa dos testes, os resultados das melhores versões das heurísticas construtivas determinística (HCD) e probabilística (HCP) foram comparados aos resultados obtidos pelo CPLEX (limitantes inferior e superior). A Tabela 29 mostra um resumo dos resultados obtidos para as 360 instâncias, sumarizados pelo número de atividades. O “gap médio” corresponde ao desvio médio do conjunto de instâncias em relação aos limitantes inferiores (LB) obtidos pelo CPLEX, o “gap máximo” é igual ao maior desvio de uma instância do conjunto em relação ao LB do CPLEX, “% solução ótima” é o percentual de instâncias em que a solução ótima foi encontrada, “% melhor solução” é o percentual de instâncias em que o método encontrou a melhor solução dentre os três procedimentos e o “tempo médio de execução” é o tempo médio para executar uma instância de teste, em segundos.

Tabela 29 – Resumo da comparação entre as heurísticas e o CPLEX.

Nº Atividades	Método	Gap médio %	Gap máximo %	% Solução Ótima	% Melhor Solução	Tempo médio execução (s)
30	HCD	31,6%	57,6%	0,0%	0,0%	0,106
	HCP 900 s	16,5%	38,5%	31,5%	75,0%	786
	CPLEX 1.800 s	17,5%	43,4%	40,7%	57,4%	1150
60	HCD	60,0%	75,9%	0,0%	0,0%	0,104
	HCP 1.800 s	45,3%	65,1%	11,1%	86,1%	1660
	CPLEX 3.600 s	51,6%	73,0%	18,5%	25,0%	2964
90	HCD	71,0%	82,5%	0,0%	0,0%	0,104
	HCP 1.800 s	56,6%	72,4%	2,8%	81,5%	1770
	CPLEX 3.600 s	70,0%	87,0%	4,6%	22,2%	3477
90 (DF=1,2)	HCD	71,6%	81,2%	0,0%	0,0%	0,104
	HCP 1.800 s	57,4%	72,1%	5,6%	83,3%	1755
	CPLEX 7.200 s	67,6%	83,9%	5,6%	22,2%	6936
120(DF=1,2)	HCD	75,6%	86,5%	0,0%	0,0%	0,100
	HCP 1.800 s	60,9%	76,2%	0,0%	83,3%	1801
	CPLEX 3.600 s	79,0%	89,7%	0,0%	16,7%	3608
120(DF=1,2)	HCD	75,6%	86,3%	0,0%	0,0%	0,100
	HCP 3.600 s	60,9%	75,9%	0,0%	86,1%	3601
	CPLEX 7.200 s	76,8%	89,7%	0,0%	19,4%	7209

De forma geral, pode-se verificar que a heurística construtiva probabilística (HCP) com regra de prioridade GCRR obteve os melhores resultados, com os menores *gaps* médios, os menores *gaps* máximos e o maior percentual de melhores soluções em um tempo computacional menor do que do CPLEX. O CPLEX obteve o maior percentual de soluções ótimas, que foram encontradas com um tempo médio de execução muito pequeno na maioria das instâncias. Já a heurística construtiva determinística (HCD) foi o método de menor tempo de execução, mas que apresentou altos *gaps* médios, altos *gaps* máximos e não encontrou nenhuma solução ótima ou melhor solução. No entanto, a HCD gerou *gaps* máximos menores que o CPLEX para instâncias com 90 e 120 atividades e gerou *gap* médio menor que o CPLEX para instâncias com 120 atividades, para ambos os tempos de execução. O desempenho da HCP foi sempre melhor do que o da HCD, conforme esperado.

Pode-se observar que conforme o número de atividades aumenta e a dificuldade de resolução da instância aumenta, a diferença entre a HCP e os outros dois métodos também aumenta. Para instâncias com 30 atividades, a diferença total entre a HCP e o CPLEX é de -1,2%. Para instâncias com 60 atividades, essa diferença total passa a ser de -11,5%; chegando a -46,3% para instâncias com 120 atividades e tempo limite de execução de 3.600 segundos para o CPLEX.

A Tabela 30 mostra um resumo por fator (número de recursos, fator de recursos (RF), complexidade da rede (NC) e fator de data de entrega (DF)) dos resultados obtidos nas instâncias de testes com 30 atividades. O *gap* médio obtido pela HCP e pelo CPLEX foram muito próximos, com desempenho da HCP ligeiramente melhor. Porém, nos grupos de instâncias mais fáceis (RF=0,25, RF=0,50 ou NC=2,1), o desempenho do CPLEX foi melhor. A diferença total entre as soluções obtidas pela PCH e pelo CPLEX, considerando todas as 108 instâncias, foi de -1,2%, com a diferença por instância variando entre 3,6% e -8,1%. Em todos os subgrupos, o CPLEX obteve o maior número de soluções ótimas, encontrando todas as 27 soluções ótimas quando RF = 0,25; que são as instâncias de testes mais fáceis.

Tabela 30 – Resumo da comparação entre a heurística construtiva probabilística (900 s) e o CPLEX (1.800 s) – instâncias de teste com 30 atividades.

		HCP-GCRR				CPLEX			
		Gap médio %	Gap máximo %	% Solução Ótima	% Melhor Solução	Gap médio %	Gap máximo %	% Solução Ótima	% Melhor Solução
Nº Recursos	4	8,3%	35,3%	50,0%	77,8%	8,5%	37,5%	52,8%	72,2%
	6	17,5%	38,5%	30,6%	83,3%	19,1%	43,4%	41,7%	47,2%
	8	19,6%	37,0%	13,9%	63,9%	20,4%	40,7%	27,8%	52,8%
RF	0,25	0,2%	1,9%	85,2%	85,2%	0,0%	0,0%	100,0%	100,0%
	0,50	7,1%	28,1%	37,0%	51,9%	6,8%	27,0%	59,3%	88,9%
	0,75	20,3%	33,0%	3,7%	81,5%	22,0%	38,1%	3,7%	22,2%
	1,00	27,3%	38,5%	0,0%	81,5%	28,7%	43,4%	0,0%	18,5%
NC	1,5	21,8%	38,5%	30,6%	86,1%	23,5%	43,4%	33,3%	44,4%
	1,8	15,5%	33,0%	30,6%	72,2%	16,4%	38,1%	44,4%	58,3%
	2,1	12,3%	31,0%	33,3%	66,7%	12,5%	30,9%	44,4%	69,4%
DF	1,1	15,2%	34,8%	30,6%	75,0%	16,3%	37,7%	38,9%	55,6%
	1,2	16,5%	36,6%	33,3%	77,8%	17,2%	37,9%	38,9%	55,6%
	1,3	18,0%	38,5%	30,6%	72,2%	19,1%	43,4%	44,4%	61,1%

A Tabela 31 contém um resumo dos resultados obtidos nas instâncias de testes com 60 atividades. O desempenho da HCP foi claramente superior ao do CPLEX. Novamente, o grupo de instâncias mais fáceis foi a exceção (RF=0,25), no qual o desempenho do CPLEX foi melhor do que o da HCP. A diferença total entre as soluções geradas pela HCP e pelo CPLEX, considerando todas as 108 instâncias, foi de -11,5%; variando instância a instância de 9,7% a -28,8%.

A HCD teve desempenho pior do que o CPLEX considerando todas as 108 instâncias. No entanto, gerou soluções melhores do que o CPLEX em 11% das instâncias do grupo com 6 recursos e em 8% das instâncias do grupo com 8 recursos, com diferenças nessas soluções geradas variando entre -0,3% e -14,1%.

Tabela 31 – Resumo da comparação entre a heurística construtiva probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 60 atividades.

		HCP-GCRR				CPLEX			
		Gap médio %	Gap máximo %	% Solução Ótima	% Melhor Solução	Gap médio %	Gap máximo %	% Solução Ótima	% Melhor Solução
Nº Recursos	4	43,6%	63,1%	19,4%	91,7%	48,8%	67,4%	25,0%	27,8%
	6	44,6%	62,5%	13,9%	88,9%	50,5%	68,2%	25,0%	25,0%
	8	46,6%	65,1%	0,0%	77,8%	53,7%	73,0%	5,6%	22,2%
RF	0,25	7,2%	21,3%	44,4%	51,9%	5,4%	16,8%	74,1%	92,6%
	0,50	37,8%	44,9%	0,0%	92,6%	41,6%	49,4%	0,0%	7,4%
	0,75	51,4%	60,3%	0,0%	100,0%	58,3%	67,8%	0,0%	0,0%
	1,00	58,8%	65,1%	0,0%	100,0%	65,6%	73,0%	0,0%	0,0%
NC	1,5	48,9%	65,1%	13,9%	88,9%	56,6%	73,0%	22,2%	25,0%
	1,8	45,4%	62,4%	11,1%	86,1%	51,1%	69,1%	16,7%	25,0%
	2,1	40,6%	59,4%	8,3%	83,3%	45,6%	66,4%	16,7%	25,0%
DF	1,1	43,2%	59,7%	5,6%	83,3%	50,7%	71,3%	16,7%	22,2%
	1,2	47,1%	63,1%	13,9%	88,9%	52,8%	70,5%	19,4%	25,0%
	1,3	45,7%	65,1%	13,9%	86,1%	51,3%	73,0%	19,4%	27,8%

A Tabela 32 mostra um resumo dos resultados obtidos nas instâncias de testes com 90 atividades e tempo limite de execução para o CPLEX de 3.600 segundos. Novamente, o desempenho da HCP foi claramente superior ao do CPLEX, com exceção para o grupo fácil com RF=0,25. A diferença total entre as soluções geradas pela HCP e pelo CPLEX, considerando todas as 108 instâncias, foi de -30,7%; variando instância a instância de 6,7% a -55,6%.

A HCD gerou *gaps* máximos menores do que o CPLEX, exceto para as instâncias com RF=0,25; RF=0,50 e DF=1,3. A HCD também gerou *gaps* médios menores do que o CPLEX para os grupos com 6 recursos, 8 recursos; RF=0,75; RF=1,00; NC=1,5 e DF=1,1; que são as instâncias mais difíceis. A HCD gerou soluções melhores do que o CPLEX em 8% das instâncias do grupo com 4 recursos, em 44% das instâncias do grupo com 6 recursos e em 36% das instâncias do grupo com 8 recursos, com diferenças nessas soluções geradas variando entre -3,5% e -35,9%.

Tabela 32 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 90 atividades.

		HCD- MLFT/GCRR		HCP-GCRR		% Sol. Ótima		% Melhor Sol.		CPLEX		% Sol. Ótima	% Melhor Sol.
		Gap médio	Gap máx.	Gap médio	Gap máx.					Gap médio	Gap máx.		
Nº Rec.	4	72%	81%	53%	69%	3%	81%	66%	84%	6%	25%		
	6	69%	79%	56%	70%	6%	83%	70%	85%	8%	22%		
	8	71%	82%	59%	72%	0%	81%	72%	87%	0%	19%		
RF	0,25	48%	57%	21%	39%	11%	26%	20%	38%	19%	89%		
	0,50	68%	74%	52%	60%	0%	100%	61%	69%	0%	0%		
	0,75	74%	81%	61%	68%	0%	100%	75%	85%	0%	0%		
	1,00	78%	82%	68%	72%	0%	100%	80%	87%	0%	0%		
NC	1,5	74%	82%	60%	72%	0%	81%	74%	87%	0%	19%		
	1,8	70%	82%	55%	72%	6%	81%	67%	86%	8%	25%		
	2,1	68%	80%	55%	72%	3%	83%	67%	84%	6%	22%		
DF	1,1	69%	80%	57%	70%	3%	83%	72%	86%	6%	19%		
	1,2	72%	81%	58%	72%	3%	83%	70%	87%	3%	22%		
	1,3	72%	82%	55%	72%	3%	78%	67%	81%	6%	25%		

A Tabela 33 apresenta um resumo dos resultados obtidos nas instâncias de testes com 90 atividades e tempo limite de execução para o CPLEX de 7.200 segundos. Conforme esperado, o desempenho do CPLEX melhorou, com diminuição de 6,7% no valor total das soluções geradas, mas a HCP continua apresentando desempenho claramente superior ao do CPLEX, mesmo com tempo limite de execução de 1.800 segundos. Novamente, a exceção foi no grupo fácil com RF=0,25; no qual o desempenho do CPLEX é levemente superior. A diferença total entre as soluções geradas pela HCP e pelo CPLEX, considerando todas as 36 instâncias, reduziu para -23,9%; variando instância a instância de 5,3% a -45,5%.

A melhora de desempenho do CPLEX em relação a HCD foi mais significativa. A HCD gerou *gaps* máximos menores do que o CPLEX somente para as instâncias com 6 recursos, 8 recursos; RF=1,00 e NC=1,5; que são as instâncias mais difíceis, e não gerou nenhum *gap* médio menor. A HCD gerou soluções melhores do que o CPLEX em 42% das instâncias do grupo com 6 recursos e em 25% das instâncias do grupo com 8 recursos e as diferenças nessas soluções geradas variou de -2,3% e -15,4%.

Tabela 33 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (7.200 s) – instâncias de teste com 90 atividades e DF=1,2.

		HCD- MLFT/GCRR		HCP-GCRR		CPLEX					
		Gap médio	Gap máx.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.
Nº Rec.	4	73%	81%	54%	69%	8%	83%	63%	76%	8%	25%
	6	70%	78%	57%	69%	8%	83%	68%	80%	8%	25%
	8	72%	81%	59%	72%	0%	83%	70%	84%	0%	17%
RF	0,25	49%	55%	21%	39%	22%	33%	20%	37%	22%	89%
	0,50	70%	74%	52%	58%	0%	100%	61%	67%	0%	0%
	0,75	74%	81%	62%	68%	0%	100%	73%	80%	0%	0%
	1,00	79%	81%	69%	72%	0%	100%	78%	84%	0%	0%
NC	1,5	75%	81%	60%	70%	0%	83%	72%	84%	0%	17%
	1,8	71%	81%	56%	72%	8%	83%	65%	78%	8%	25%
	2,1	69%	79%	56%	72%	8%	83%	65%	78%	8%	25%

A Tabela 34 contém um resumo dos resultados obtidos nas instâncias de testes com 120 atividades e tempo limite de execução para o CPLEX de 3.600 segundos. O desempenho da HCP foi claramente superior ao do CPLEX em todos os grupos, mesmo no grupo mais fácil com RF=0,25 no qual o CPLEX havia sido superior nas instâncias com até 90 atividades. A diferença total entre as soluções geradas pela HCP e pelo CPLEX, considerando todas as 36 instâncias, foi de -46,3%; variando instância a instância de 8,9% a -65,6%.

Tabela 34 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (1.800 s) e o CPLEX (3.600 s) – instâncias de teste com 120 atividades e DF=1,2.

		HCD- MLFT/GCRR		HCP-GCRR		CPLEX					
		Gap médio	Gap máx.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.
Nº Rec.	4	73%	85%	52%	75%	0%	58%	71%	88%	0%	42%
	6	77%	86%	64%	76%	0%	92%	80%	90%	0%	8%
	8	76%	86%	62%	75%	0%	100%	81%	90%	0%	0%
RF	0,25	64%	85%	41%	75%	0%	78%	55%	88%	0%	22%
	0,50	73%	84%	55%	72%	0%	78%	75%	85%	0%	22%
	0,75	77%	83%	65%	73%	0%	89%	82%	90%	0%	11%
	1,00	81%	86%	70%	76%	0%	89%	85%	90%	0%	11%
NC	1,5	78%	86%	64%	76%	0%	83%	82%	90%	0%	17%
	1,8	74%	85%	57%	75%	0%	92%	77%	89%	0%	8%
	2,1	74%	84%	61%	75%	0%	75%	78%	85%	0%	25%

A HCD também obteve desempenho melhor do que o CPLEX, com uma diferença total entre as soluções geradas de -14,0%. A HCD gerou *gaps* médios menores, sendo que a única exceção foi no grupo com 4 recursos, onde o CPLEX apresentou *gap* médio menor. A HCD também gerou *gaps* máximos menores do que o CPLEX em todos os grupos de instâncias. A HCD gerou soluções melhores do que o CPLEX em 33% das instâncias do grupo com 4 recursos, em 67% das instâncias do grupo com 6 recursos e em 67% das instâncias do grupo com 8 recursos, com diferenças nessas soluções geradas variando entre -1,4% a -49,3%.

Por fim, a Tabela 35 mostra um resumo dos resultados obtidos nas instâncias de testes com 120 atividades e tempo limite de execução para a HCP de 3.600 segundos e para o CPLEX de 7.200 segundos. Conforme esperado, o desempenho do CPLEX melhorou bastante, com diminuição em 9,4% no valor total das soluções geradas, mas a HCP continua apresentando desempenho claramente superior ao do CPLEX. Já a HCP apresentou uma melhora de apenas 0,4% no valor total das soluções geradas, com ganho maior nas instâncias mais difíceis com 8 recursos. A diferença total entre as soluções geradas pela HCP e pelo CPLEX, considerando todas as 36 instâncias, reduziu para -41,0%; variando instância a instância de 6,3% a -61,8%.

Tabela 35 – Resumo da comparação entre as heurísticas construtivas determinística e probabilística (3.600 s) e o CPLEX (7.200 s) – instâncias de teste com 120 atividades e DF=1,2.

		HCD- MLFT/GCRR		HCP-GCRR				CPLEX			
		Gap médio	Gap máx.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.	Gap médio	Gap máx.	% Sol. Ótima	% Melhor Sol.
Nº Rec.	4	73%	85%	52%	75%	0%	75%	70%	88%	0%	42%
	6	76%	86%	64%	76%	0%	92%	78%	89%	0%	8%
	8	76%	86%	62%	75%	0%	92%	79%	90%	0%	8%
RF	0,25	64%	85%	40%	75%	0%	78%	53%	88%	0%	33%
	0,50	73%	84%	54%	72%	0%	89%	70%	85%	0%	22%
	0,75	77%	83%	65%	73%	0%	89%	82%	90%	0%	11%
	1,00	81%	86%	70%	76%	0%	89%	84%	89%	0%	11%
NC	1,5	78%	86%	64%	76%	0%	92%	81%	90%	0%	17%
	1,8	74%	85%	57%	75%	0%	92%	73%	88%	0%	17%
	2,1	74%	84%	61%	75%	0%	75%	75%	85%	0%	25%

A melhora de desempenho do CPLEX em relação a HCD foi mais significativa, porém a HCD continuou com desempenho melhor do que o CPLEX, com uma diferença total

entre as soluções geradas reduzindo para -5,1%. A HCD gerou *gaps* médios menores, sendo que novamente a única exceção foi no grupo com 4 recursos, onde o CPLEX apresentou *gap* médio menor. A HCD continuou gerando *gaps* máximos menores do que o CPLEX em todos os grupos de instâncias. A HCD gerou soluções melhores do que o CPLEX em 33% das instâncias do grupo com 4 recursos, em 50% das instâncias do grupo com 6 recursos e em 67% das instâncias do grupo com 8 recursos, com diferenças nessas soluções geradas variando entre -0,6% a -39,9%.

6 CONCLUSÕES

Este trabalho teve como objetivo estudar o problema de programação de projetos com custo de disponibilidade de recursos e custo de penalidade por atraso no término do projeto (RACPT – *Resource Availability Cost Problem with Tardiness*) e propor uma heurística construtiva determinística (HCD) e uma heurística construtiva probabilística (HCP) para resolvê-lo, dada a escassez de métodos de resolução publicados. A ideia dos algoritmos é combinar a flexibilidade da atividade com a flexibilidade do recurso para selecionar a próxima atividade a ser programada, sendo que na heurística probabilística essa seleção é aleatória com viés. As atividades são programadas na data mais cedo em que sua execução não gerar o maior pico de utilização dos recursos ou na data mais cedo em que o custo total do projeto for menor.

A melhor versão das heurísticas foi obtida após o teste de várias regras de prioridade, conforme a extensa revisão bibliográfica realizada. A regra de prioridade GCRR (maior necessidade de recurso acumulada) combinada com a regra de desempate MLFT (mínimo instante mais tarde de início) gerou o melhor resultado tanto para a heurística construtiva determinística quanto para a probabilística.

Em seguida, as heurísticas propostas foram testadas em 360 instâncias de testes e seus resultados foram comparados aos obtidos pela formulação matemática baseada em *strip packing* e restrições disjuntivas implementada no programa CPLEX. A heurística construtiva determinística gerou uma solução viável rapidamente, porém de baixa qualidade, principalmente ao compararmos com os resultados da heurística construtiva probabilística. Esse desempenho fraco era esperado devido ao critério guloso (*greedy*) de seleção da próxima atividade a ser programada. No entanto, a HCD gerou soluções melhores do que o CPLEX em 16,4% das instâncias e obteve desempenho claramente melhor do que o CPLEX nas instâncias com 120 atividades, mesmo quando o tempo limite de execução do CPLEX foi de 7.200 segundos.

Já a heurística construtiva probabilística gera soluções ótimas ou próximas da ótima para problemas pequenos ou para problemas fáceis e gera soluções muito melhores do que o CPLEX na metade do tempo computacional para os problemas médios e grandes ou para problemas difíceis. O aspecto guloso da HCD é atenuado pela seleção aleatória com viés, gerando melhorias significativas nas soluções obtidas. De forma geral, pode-se verificar que a HCP com regra de prioridade GCRR obteve os

menores *gaps* médios, os menores *gaps* máximos e o maior percentual de melhores soluções. O CPLEX obteve o maior percentual de soluções ótimas, que foram encontradas com um tempo médio de execução muito pequeno na maioria das instâncias. Pode-se observar que conforme o número de atividades aumenta e a dificuldade de resolução da instância aumenta, a diferença entre a HCP e o CPLEX também aumenta. Para instâncias com 30 atividades, a diferença total entre a HCP e o CPLEX é de -1,2%; chegando a -46,3% para instâncias com 120 atividades e tempo limite de execução de 3.600 segundos para o CPLEX. Por outro lado, para as instâncias com 120 atividades, ao aumentarmos o tempo limite de execução da HCP de 1.800 segundos para 3.600 segundos, a melhora no valor total das soluções geradas foi de apenas 0,4%, indicando potencial de melhoria na sua estrutura.

Dado os bons resultados obtidos e à implementação no programa VBA for Microsoft Excel, a heurística construtiva probabilística proposta é um método bom e prático para resolução do RACPT, podendo ser combinada com a heurística construtiva determinística para gerar uma solução viável rapidamente.

REFERÊNCIAS BIBLIOGRÁFICAS

AFSHAR-NADJAFI, B. Using GRASP for resource availability cost problem with time dependent resource cost. **Economic Computation and Economic Cybernetics Studies and Research**, v. 48, n. 1, 2014a.

AFSHAR-NADJAFI, B. Multi-mode resource availability cost problem with recruitment and release dates for resources. **Applied Mathematical Modelling**, v. 38, n. 21–22, p. 5347–5355, 2014b.

AFSHAR-NADJAFI, B.; ARANI, M. Multimode preemptive resource investment problem subject to due dates for activities: Formulation and solution procedure. **Advances in Operations Research**, v. 2014, 2014.

ARCUS, A. L. COMSOAL: a computer method of sequencing operations for assembly lines. **International Journal of Production Research**, v. 4, n. 4, p. 259–277, 1965.

BURGESS, A. R.; KILLEBREW, J. B. Variation in activity level on a cyclic arrow diagram. **International Journal of Industrial Engineering**, v. 13, n. 2, p. 76–83, 1962.

CHEN, L.; LI, X.; CAI, Z. Heuristic methods for minimizing resource availability costs in multi-mode project scheduling. **Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics**, p. 809–813, 2012.

COLAK, E.; AZIZOGLU, M. A resource investment problem with time/resource trade-offs. **Journal of the Operational Research Society**, v. 65, n. 5, p. 777–790, 2014.

COUGHLAN, E. T.; LÜBBECKE, M. E.; SCHULZ, J. A branch-and-price algorithm for multi-mode resource leveling. In: FESTA, P. (Ed.). **SEA 2010, Lecture Notes in Computer Science 6049**. Verlag Berlin Heidelberg: Springer, 2010. p. 226–238.

COUGHLAN, E. T.; LÜBBECKE, M. E.; SCHULZ, J. A branch-price-and-cut algorithm for multi-mode resource leveling. **European Journal of Operational Research**, v. 245, n. 1, p. 70–80, 2015.

DEMEULEMEESTER, E. Minimizing resource availability costs in time-limited project networks. **Management Science**, v. 41, n. 10, p. 1590–1598, 1995.

DEMEULEMEESTER, E.; HERROELEN, W. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. **Management Science**, v. 38, n. 12, p. 1803–1818, 1992.

DREXL, A. et al. ProGen/πχ - an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. **European Journal of Operational Research**, v. 125, n. 1, p. 59–72, 2000.

DREXL, A.; KIMMS, A. Optimization guided lower and upper bounds for the resource investment problem. **Journal of the Operational Research Society**, v. 52, n. 3, p. 340–351, 2001.

HARTMANN, S.; BRISKORN, D. A survey of variants and extensions of the resource-constrained project scheduling problem. **European Journal of Operational Research**, v. 207, n. 1, p. 1–14, 2010.

HSU, C.-C.; KIM, D. S. A new heuristic for the multi-mode resource investment problem. **Journal of the Operational Research Society**, v. 56, n.4, p. 406–413, 2005.

JAVANMARD, S. et al. Preemptive multi-skilled resource investment project scheduling problem: Mathematical modelling and solution approaches. **Computers and Chemical Engineering**, v. 96, p. 55–68, 2017.

JĘDRZEJOWICZ, P.; RATAJCZAK-ROPEL, E. A-Team for solving the resource availability cost problem. In: NGUYEN, N. T.; HOANG, K.; JĘDRZEJOWICZ, P. (Eds.). **Computational Collective Intelligence. Technologies and Applications. ICCCI 2012. Lecture Notes in Computer Science**. Berlin Heidelberg: Springer, 2012. v. 7654. p. 443–452.

KASTOR, A.; SIRAKOULIS, K. The effectiveness of resource levelling tools for Resource Constraint Project Scheduling Problem. **International Journal of Project Management**, v. 27, n. 5, p. 493–500, 2009.

KELLEY JR., J. E. The Critical-Path Method: Resource planning and scheduling. In: MUTH, J. F.; THOMPSON, G. L. (Eds.). **Industrial Scheduling**. New Jersey: Prentice-Hall, 1963. p. 347–365.

KLEIN, R. **Scheduling of resource-constrained projects**. New York: Springer Science+Business Media, 2000. 378 p.

KOLISCH, R.; HARTMANN, S. Experimental investigation of heuristics for resource-constrained project scheduling: An update. **European Journal of Operational Research**, v. 174, n. 1, p. 23–37, 2006.

KOLISCH, R.; SPRECHER, A. PSPLIB - A project scheduling problem library. **European Journal of Operational Research**, v. 96, n. 1, p. 205–216, 1997.

KOLISCH, R.; SPRECHER, A.; DREXL, A. Characterization and generation of a general class of resource-constrained project scheduling problems. **Management Science**, v. 41, n. 10, p. 1693–1703, 1995.

KOULINAS, G. K.; ANAGNOSTOPOULOS, K. P. A new tabu search-based hyper-heuristic algorithm for solving construction leveling problems with limited resource availabilities. **Automation in Construction**, v. 31, p. 169–175, 2013.

LEMOES, F. K. **Problema de programação de uma operação de empacotamento não-guilhotinado em ambiente de máquina única, minimizando custos de matéria-prima e desvio de datas**: formulação e solução heurística. 2013. 174 p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2013.

LIBERATORE, M. J.; POLLACK-JOHNSON, B.; SMITH, C. A. Project management in

construction: Software use and research directions. **Journal of Construction Engineering and Management**, v. 127, n. 2, p. 101–107, 2001.

MENG, H. et al. A scatter search hybrid algorithm for resource availability cost problem. In: KIM, J.; GEEM, Z. (Eds.). **Harmony Search Algorithm. Advances in Intelligent Systems and Computing**, vol. 382. Berlin, Heidelberg: Springer, 2016. p. 39–51.

MÖHRING, R. H. Minimizing costs of resource requirements in project networks subject to a fixed completion time. **Operations Research**, v. 32, n. 1, p. 89–120, 1984.

MORABITO, R.; PUREZA, V. Modelagem e simulação. In: MIGUEL, P. A. C. (Ed.). **Metodologia de pesquisa em engenharia de produção e gestão de operações**. 2. ed. Rio de Janeiro: Elsevier, 2012. p. 169–197.

NEUMANN, K.; SCHWINDT, C.; ZIMMERMANN, J. Recent results on resource-constrained project scheduling with time windows: Models, solution methods, and applications. **Central European Journal of Operations Research**, v. 10, n. 2, p. 113–148, 2002.

NEUMANN, K.; ZIMMERMANN, J. Resource levelling for projects with schedule-dependent time windows. **European Journal of Operational Research**, v. 117, n. 3, p. 591–605, 1999.

NEUMANN, K.; ZIMMERMANN, J. Procedures for resource levelling and net present value problems in project scheduling with general temporal and resource constraints. **European Journal of Operational Research**, v. 127, n. 2, p. 425–443, 2000.

PATTERSON, J. H. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. **Management Science**, v. 30, n. 7, p. 854–867, 1984.

PINEDO, M. L. **Scheduling: theory, algorithms, and systems**. 4. ed. New York: Springer, 2010. 673 p.

QI, J.-J. et al. Solving the multi-mode resource availability cost problem in project scheduling based on modified particle swarm optimization. **Arabian Journal for Science and Engineering**, v. 39, n. 6, p. 5279–5288, 2014a.

QI, J.-J. et al. Solving resource availability cost problem in project scheduling by pseudo particle swarm optimization. **Journal of Systems Engineering and Electronics**, v. 25, n. 1, p. 69–76, 2014b.

QI, J.-J. et al. Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization. **Journal of Scheduling**, v. 18, n. 3, p. 285–298, 2015.

RANJBAR, M.; KIANFAR, F.; SHADROKH, S. Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. **Applied Mathematics and Computation**, v. 196, n. 2, p. 879–888, 2008.

RODRIGUES, S. B.; YAMASHITA, D. S. An exact algorithm for minimizing resource availability costs in project scheduling. **European Journal of Operational Research**, v. 206, n. 3, p. 562–568, 2010.

SABZEHPARVAR, M.; SEYED-HOSSEINI, S. M.; NOURI, S. A mathematical model for the multi-mode resource investment problem. **Journal of Industrial Engineering International**, v. 4, n. 7, p. 25–32, 2008.

SANTORO, M. C. **Formulação disjuntiva para problemas de scheduling**. São Paulo: Departamento de Engenharia de Produção da Escola Politécnica da Universidade de São Paulo, 2016. (Publicação interna).

SHADROKH, S.; KIANFAR, F. A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. **European Journal of Operational Research**, v. 181, n. 1, p. 86–101, 2007.

SPRECHER, A.; DREXL, A. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. **European Journal of Operational Research**, v. 107, n. 2, p. 431–450, 1998.

TORMOS, P.; LOVA, A. An efficient multi-pass heuristic for project scheduling with constrained resources. **International Journal of Production Research**, v. 41, n. 5, p. 1071–1086, 2003.

VAN PETEGHEM, V.; VANHOUCKE, M. An artificial immune system algorithm for the resource availability cost problem. **Flexible Services and Manufacturing Journal**, v. 25, n. 1–2, p. 122–144, 2013.

VAN PETEGHEM, V.; VANHOUCKE, M. Heuristic methods for the resource availability cost problem. In: SCHWINDT, C.; ZIMMERMANN, J. (Eds.). **Handbook on Project Management and Scheduling Vol.1**. Cham: Springer International Publishing, 2015. p. 339–359.

VANHOUCKE, M. et al. An evaluation of the adequacy of project network generators with systematically sampled networks. **European Journal of Operational Research**, v. 187, n. 2, p. 511–524, 2008.

YAMASHITA, D. S.; ARMENTANO, V. A.; LAGUNA, M. Scatter search for project scheduling with resource availability cost. **European Journal of Operational Research**, v. 169, n. 2, p. 623–637, 2006.

YAMASHITA, D. S.; MORABITO, R. A note on time/cost tradeoff curve generation for project scheduling with multi-mode resource availability costs. **International Journal of Operational Research**, v. 5, n. 4, p. 429–444, 2009.

YANG, K.-K. A comparison of dispatching rules for executing a resource-constrained project with estimated activity durations. **Omega**, v. 26, n. 6, p. 729–738, 1998.

ZHU, X. et al. An effective heuristic for project scheduling with resource availability cost. **European Journal of Operational Research**, v. 257, n. 3, p. 746–762, 2017.

APÊNDICE A – Código fonte para a heurística construtiva em VBA

Option Explicit

'dados das instancias de teste

Type rProj

IdProj As String

QtdAtiv As Long

QtdRec As Long

CustoRec() As Double

DF As Double

End Type

Public mProj() As rProj

'dados das atividades

Type rAtiv

IdProj As String

IdAtiv As Long

QtdPreced As Long

Preced() As Long

Sucess(3) As Long

Dura As Integer

NecesRec() As Long

QtdSucess As Long

RPW As Long

CRR As Long

RR As Long

End Type

Public mAtiv() As rAtiv

Public supAtiv() As rAtiv

'dados da solução CPM

Type rCPM

IdAtiv As Long

ES As Long

EF As Long

LS As Long

LF As Long

folga_total As Long

folga_livre As Long

folga_seguranca As Long

folga_independente As Long

End Type

Public mCPM() As rCPM

Public mSimulCPM() As rCPM

'dados de uma solução - atividades

Type rSolAtiv

IdAtiv As Long

TInicio As Long

TFim As Long

End Type

Public mSolAtivAtual() As rSolAtiv

```

Public mMelhorSolAtiv() As rSolAtiv

'dados de uma solução - Matriz de Recursos
Type rMatrizRec
    IdRec As Long
    ConsumoRecursos() As Long
    Dispon As Long
    MediaNecess As Double
    PercPicoMedia As Double
    InicioPico As Long
    FimPico As Long
    SomaRecursos As Long
End Type
Public mMatrRecSolAtual() As rMatrizRec
Public mMatrRecMelhorSol() As rMatrizRec
Public mMatrRecLB() As rMatrizRec
Public mMatrRecSimul() As rMatrizRec
Public mMatrizRecCPM_ES() As rMatrizRec

'dados de uma solução - valor da função objetivo
Type rValorFO
    CustoRecursos As Double
    CustoAtraso As Double
    CustoTotal As Double
End Type
Public mValorFO_SolAtual As rValorFO
Public mValorFO_MelhorSol As rValorFO
Public mValorFO_Simul As rValorFO
Public mValorFO_MelhorSimul As rValorFO

'dados de uma solução - Lista de Atividades Candidatas
Type rListaAtivCand
    IdAtiv As Long
    Chave As String
    Probab As Double
    Chave_OK As Long
End Type
Public mListaAtivCand() As rListaAtivCand

Public QtdInst As Long, QtdInstAtiv As Long
Public Inst_Atual As Long
Public CustoUnitAtraso As Double
Public MakespanCPM As Long, DataEntrega As Long, DataLimite As Long
Public ListaAtivSGS() As Long, ListaAtivNaoProgr() As Long
Public ListaRecursos() As String
Public cont_ativ As Long
Public cont_cand As Long
Public HoraInicio As Date, HoraFim As Date
Public m_anterior As Long
Public FO_LB As Double
Public ProjetoBaixoRecurso As Long

```

```

Sub CarregaDados()

Dim i As Long
Dim Suporte As Variant, num As Long

'limpa dados abas de resultado e preenche cabeçalho
Sheets("Result_F0").Select
Cells.Select
Selection.ClearContents
Selection.Delete Shift:=xlUp
Cells(1, 1) = "Nome Instância"
Cells(1, 2) = "Data de Término"
Cells(1, 3) = "Valor da Função Objetivo"
Cells(1, 4) = "Data Entrega"
Cells(1, 5) = "Custo Recursos"
Cells(1, 6) = "Custo Atraso"
Cells(1, 7) = "Iteração Melhor Sol"
Cells(1, 8) = "Ultima Iteração"
Cells(1, 9) = "Tempo Melhor Solução"
Cells(1, 10) = "Tempo Execução"
Cells.Select
Cells.EntireColumn.AutoFit
Columns("E:E").Select
Selection.Style = "Percent"

'grava horario inicio execução
Sheets("Result_F0").Select
Cells(1, 15) = "Horário Início"
Cells(2, 15) = Now

'limpa dados abas de resultado e preenche cabeçalho
Sheets("Result_Rec").Select
Cells.Select
Selection.ClearContents
Selection.Delete Shift:=xlUp
Cells(1, 1) = "Nome Instância"
Cells(1, 2) = "Cód Recurso"
Cells(1, 3) = "Disponibilidade"
Cells.Select
Cells.EntireColumn.AutoFit

'limpa dados abas de resultado e preenche cabeçalho
Sheets("Result_Ativ").Select
Cells.Select
Selection.ClearContents
Selection.Delete Shift:=xlUp
Cells(1, 1) = "Nome Instância"
Cells(1, 2) = "Cód Atividade"
Cells(1, 3) = "Data Início"
Cells(1, 4) = "Duração"
Cells(1, 5) = "Data Fim"
Cells.Select
Cells.EntireColumn.AutoFit

```

```

'carrega mProj com dados de todas as instancias
Sheets("Dados Instâncias").Select
i = 0
Do While Cells(i + 2, 1) <> ""
    i = i + 1
Loop

QtdInst = i
QtdInstAtiv = 0
ReDim mProj(QtdInst)

For i = 1 To QtdInst
    mProj(i).IdProj = Cells(1 + i, 1)
    mProj(i).QtdAtiv = Cells(1 + i, 2)
    mProj(i).QtdRec = Cells(1 + i, 3)
    ReDim mProj(i).CustoRec(Cells(1 + i, 3))
    Suporte = Split(Cells(1 + i, 4), " ")
    For num = LBound(Suporte) To UBound(Suporte) - 1
        mProj(i).CustoRec(num) = Suporte(num)
    Next
    mProj(i).DF = Cells(1 + i, 7)
    QtdInstAtiv = QtdInstAtiv + mProj(i).QtdAtiv
Next

'ajusta QtdInstAtiv - divide por 3 porque foram utilizados 3 DF
QtdInstAtiv = QtdInstAtiv / 3

'carrega mAtiv com os dados de todas as atividades de todas as instancias
Sheets("Dados Atividades").Select

ReDim mAtiv(QtdInstAtiv)
For i = 1 To QtdInstAtiv
    mAtiv(i).IdProj = Cells(1 + i, 1)
    mAtiv(i).IdAtiv = Cells(1 + i, 2)
    mAtiv(i).QtdPreced = Cells(1 + i, 3)
    ReDim mAtiv(i).Preced(Cells(1 + i, 3))
    If (Cells(1 + i, 3) > 1) Then
        Suporte = Split(Cells(1 + i, 4), " ")
        For num = LBound(Suporte) To UBound(Suporte) - 1
            mAtiv(i).Preced(num) = Suporte(num)
        Next
    End If
    If (Cells(1 + i, 3) = 1) Then
        mAtiv(i).Preced(0) = Cells(1 + i, 4)
    End If
    mAtiv(i).Dura = Cells(1 + i, 5)
    Suporte = Split(Cells(1 + i, 6), " ")
    ReDim mAtiv(i).NecesRec(UBound(Suporte) - 1)
    For num = LBound(Suporte) To UBound(Suporte) - 1
        mAtiv(i).NecesRec(num) = Suporte(num)
    Next
Next

End Sub

```



```

Sub InicializaDados_Instance()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
i = Inst_Atual

HoraInicio = Now
sup_tempo = 0

'cria supAtiv somente com os dados das atividades da instancia atual
ReDim supAtiv(mProj(i).QtdAtiv)
For j = 1 To QtdInstAtiv
    If mProj(i).IdProj = mAtiv(j).IdProj Then
        supAtiv(mAtiv(j).IdAtiv).IdProj = mAtiv(j).IdProj
        supAtiv(mAtiv(j).IdAtiv).IdAtiv = mAtiv(j).IdAtiv
        supAtiv(mAtiv(j).IdAtiv).QtdPreced = mAtiv(j).QtdPreced
        ReDim supAtiv(mAtiv(j).IdAtiv).Preced(mAtiv(j).QtdPreced)
        For k = 0 To mAtiv(j).QtdPreced - 1
            supAtiv(mAtiv(j).IdAtiv).Preced(k) = mAtiv(j).Preced(k)
        Next
        supAtiv(mAtiv(j).IdAtiv).Sucess(1) = -1
        supAtiv(mAtiv(j).IdAtiv).Sucess(2) = -1
        supAtiv(mAtiv(j).IdAtiv).Sucess(3) = -1
        supAtiv(mAtiv(j).IdAtiv).Dura = mAtiv(j).Dura
        supAtiv(mAtiv(j).IdAtiv).RR = 0
        ReDim supAtiv(mAtiv(j).IdAtiv).NecesRec(mProj(i).QtdRec)
        For k = 0 To mProj(i).QtdRec - 1
            supAtiv(mAtiv(j).IdAtiv).NecesRec(k) = mAtiv(j).NecesRec(k)
            supAtiv(mAtiv(j).IdAtiv).RR = supAtiv(mAtiv(j).IdAtiv).Dura *
mAtiv(j).NecesRec(k) + supAtiv(mAtiv(j).IdAtiv).RR
        Next
        supAtiv(mAtiv(j).IdAtiv).QtdSucess = 0
        'se for a ultima atividade do projeto, sai do loop
        If mAtiv(j).IdAtiv = mProj(i).QtdAtiv - 1 Then
            j = QtdInstAtiv + 1
        End If
    End If
Next

'preenche supAtiv com dados das atividades sucessoras
For k = 0 To mProj(i).QtdAtiv - 1
    For l = 0 To mProj(i).QtdAtiv - 1
        For m = 0 To supAtiv(l).QtdPreced - 1
            If supAtiv(l).Preced(m) = k Then
                If supAtiv(k).Sucess(1) = -1 Then
                    supAtiv(k).Sucess(1) = 1
                    supAtiv(k).QtdSucess = 1
                Else
                    If supAtiv(k).Sucess(2) = -1 Then
                        supAtiv(k).Sucess(2) = 1
                        supAtiv(k).QtdSucess = 2
                    Else
                        If supAtiv(k).Sucess(3) = -1 Then
                            supAtiv(k).Sucess(3) = 1
                        End If
                    End If
                End If
            End If
        Next m
    Next l
Next k

```

```

        supAtiv(k).QtdSucess = 3
        m = supAtiv(1).QtdPreced
        l = mProj(i).QtdAtiv
    End If
End If
End If
End If
Next
Next
Next

'preenche RPW e CRR
k = mProj(i).QtdAtiv - 2
Do While k > 0
    supAtiv(k).RPW = supAtiv(k).Dura
    supAtiv(k).CRR = supAtiv(k).RR
    For l = 1 To supAtiv(k).QtdSucess
        supAtiv(k).RPW = supAtiv(k).RPW +
supAtiv(supAtiv(k).Sucess(l)).RPW
        supAtiv(k).CRR = supAtiv(k).CRR +
supAtiv(supAtiv(k).Sucess(l)).CRR
    Next
    k = k - 1
Loop

'inicializa dados melhor solução, ES_CPM e LB recursos (1x por instancia)
e calcula Data Limite como soma das durações das atividades
DataLimite = 0
ReDim mMelhorSolAtiv(mProj(i).QtdAtiv)
For l = 0 To mProj(i).QtdAtiv - 1
    DataLimite = DataLimite + supAtiv(l).Dura
    mMelhorSolAtiv(l).IdAtiv = l
    mMelhorSolAtiv(l).TInicio = -1
    mMelhorSolAtiv(l).TFim = -1
Next
ReDim mMatrRecMelhorSol(mProj(i).QtdRec)
ReDim Limite_Perc_PicoMedia(mProj(i).QtdRec)
ReDim mMatrRecLB(mProj(i).QtdRec)
ReDim mMatrizRecCPM_ES(mProj(i).QtdRec)
For k = 0 To mProj(i).QtdRec - 1
    mMatrRecMelhorSol(k).IdRec = k
    mMatrRecMelhorSol(k).Dispon = 0
    mMatrRecMelhorSol(k).PercPicoMedia = 0
    mMatrRecMelhorSol(k).MediaNecess = 0
    mMatrRecMelhorSol(k).SomaRecursos = 0
    mMatrRecLB(k).IdRec = k
    mMatrRecLB(k).Dispon = 0
    mMatrizRecCPM_ES(k).IdRec = k
    mMatrizRecCPM_ES(k).Dispon = 0
    ReDim mMatrRecMelhorSol(k).ConsumoRecursos(DataLimite)
    ReDim mMatrizRecCPM_ES(k).ConsumoRecursos(DataLimite)
    For j = 0 To DataLimite
        mMatrRecMelhorSol(k).ConsumoRecursos(j) = 0
        mMatrizRecCPM_ES(k).ConsumoRecursos(j) = 0
    Next
Next

```

```

    Next
Next
mValorFO_MelhorSol.CustoRecursos = 0
mValorFO_MelhorSol.CustoAtraso = 0
mValorFO_MelhorSol.CustoTotal = 999999999

'prepara matrizes utilizadas no calculo da solucao
ReDim mCPM(mProj(i).QtdAtiv)
ReDim mSimulCPM(mProj(i).QtdAtiv)
ReDim mSolAtivAtual(mProj(i).QtdAtiv)
ReDim mListaAtivCand(mProj(i).QtdAtiv)
ReDim ListaAtivNaoProgr(mProj(i).QtdAtiv)
ReDim mMatrRecSolAtual(mProj(i).QtdRec)
ReDim mMatrRecSimul(mProj(i).QtdRec)
ReDim ListaRecursos(mProj(i).QtdRec)

'calcula custo por tempo de atraso
CustoUnitAtraso = 0
For k = 0 To mProj(i).QtdRec - 1
    CustoUnitAtraso = CustoUnitAtraso + mProj(i).CustoRec(k)
Next
CustoUnitAtraso = CustoUnitAtraso * 0.25

End Sub

Sub InicializaDados_Solucao()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
i = Inst_Atual

'inicializa dados do resultado CPM, dados das atividades - solucao atual,
inicializa Listas de Atividades
For l = 0 To mProj(i).QtdAtiv - 1
    mCPM(l).IdAtiv = l
    mSolAtivAtual(l).IdAtiv = l
    mSolAtivAtual(l).TInicio = -1
    mSolAtivAtual(l).TFim = -1
    ListaAtivNaoProgr(l) = 1
Next
mCPM(0).ES = 0
mCPM(0).EF = 0
mCPM(0).LS = 0
mCPM(0).LF = 0

'inicializa dados da matriz de recursos - Solucao Atual
For k = 0 To mProj(i).QtdRec - 1
    mMatrRecSolAtual(k).IdRec = k
    mMatrRecSolAtual(k).Dispon = 0
    mMatrRecSolAtual(k).PercPicoMedia = 0
    mMatrRecSimul(k).IdRec = k
    mMatrRecSimul(k).Dispon = 0
    mMatrRecSimul(k).PercPicoMedia = 0
    mMatrRecSimul(k).InicioPico = 0
    mMatrRecSimul(k).FimPico = 0

```

```

        ListaRecursos(k) = 999999999
        ReDim mMatrRecSolAtual(k).ConsumoRecursos(DataLimite)
        ReDim mMatrRecSimul(k).ConsumoRecursos(DataLimite)
        For j = 0 To DataLimite
            mMatrRecSolAtual(k).ConsumoRecursos(j) = 0
            mMatrRecSimul(k).ConsumoRecursos(j) = 0
        Next
    Next

'Inicializa valor da FO
mValorFO_SolAtual.CustoRecursos = 0
mValorFO_SolAtual.CustoAtraso = 0
mValorFO_SolAtual.CustoTotal = 0
mValorFO_Simul.CustoRecursos = 0
mValorFO_Simul.CustoAtraso = 0
mValorFO_Simul.CustoTotal = 0

End Sub

Sub CalculaCPM_Parte1()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
Dim min_ES_Suc As Long, max_LF_Prec As Long
i = Inst_Atual

'calcula ES e EF
For k = 1 To mProj(i).QtdAtiv - 1
    'limpa dados ES, EF, LS, LF
    mCPM(k).ES = 0
    mCPM(k).EF = 0
    mCPM(k).LS = 0
    mCPM(k).LF = 0
    'se atividade ainda não foi programada, calcula ES com base nas
    atividades precedentes
    If mSolAtivAtual(k).TFim = -1 Then
        For l = 0 To supAtiv(k).QtdPreced - 1
            If mCPM(k).ES < mCPM(supAtiv(k).Preced(l)).EF Then
                mCPM(k).ES = mCPM(supAtiv(k).Preced(l)).EF
            End If
        Next
    Else
        'se atividade já foi programada, considera ES = Data Inicio
        mCPM(k).ES = mSolAtivAtual(k).TInicio
    End If
    mCPM(k).EF = mCPM(k).ES + supAtiv(k).Dura
Next

'calcula LS e LF
If mCPM(mProj(i).QtdAtiv - 1).EF < DataEntrega Then
    mCPM(mProj(i).QtdAtiv - 1).LF = DataEntrega
Else
    mCPM(mProj(i).QtdAtiv - 1).LF = mCPM(mProj(i).QtdAtiv - 1).EF
End If

```

```

mCPM(mProj(i).QtdAtiv - 1).LS = mCPM(mProj(i).QtdAtiv - 1).LF -
supAtiv(mProj(i).QtdAtiv - 1).Dura
k = mProj(i).QtdAtiv - 2
Do While k > 0
    For l = 1 To 3
        If supAtiv(k).Sucess(l) <> -1 Then
            If mCPM(k).LF = 0 Then
                mCPM(k).LF = mCPM(supAtiv(k).Sucess(1)).LS
            Else
                If mCPM(k).LF > mCPM(supAtiv(k).Sucess(1)).LS Then
                    mCPM(k).LF = mCPM(supAtiv(k).Sucess(1)).LS
                End If
            End If
        End If
    Next
    mCPM(k).LS = mCPM(k).LF - supAtiv(k).Dura
    k = k - 1
Loop

'calcula folgas
For k = 1 To mProj(i).QtdAtiv - 2
    'só calcula para atividades nao programadas - economiza tempo
    If mSolAtivAtual(k).TFim = -1 Then
        mCPM(k).folga_total = mCPM(k).LS - mCPM(k).ES
        min_ES_Suc = mCPM(supAtiv(k).Sucess(1)).ES
        For m = 2 To 3
            If supAtiv(k).Sucess(m) <> -1 Then
                If mCPM(supAtiv(k).Sucess(m)).ES < min_ES_Suc Then
                    min_ES_Suc = mCPM(supAtiv(k).Sucess(m)).ES
                End If
            End If
        Next
        mCPM(k).folga_livre = min_ES_Suc - mCPM(k).EF
        max_LF_Prec = mCPM(supAtiv(k).Preced(0)).LF
        For m = 1 To supAtiv(k).QtdPreced - 1
            If mCPM(supAtiv(k).Preced(m)).LF > max_LF_Prec Then
                max_LF_Prec = mCPM(supAtiv(k).Preced(m)).LF
            End If
        Next
        mCPM(k).folga_seguranca = mCPM(k).LS - max_LF_Prec
        mCPM(k).folga_independente = 500 + min_ES_Suc - max_LF_Prec -
supAtiv(k).Dura
    End If
Next

End Sub

Sub CalculaCPM_Parte2()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
i = Inst_Atual

'calcula makespan cpm
MakespanCPM = mCPM(mProj(i).QtdAtiv - 1).EF

```

```

'calcula data entrega do projeto
DataEntrega = MakespanCPM * mProj(i).DF

'preenche matriz de recursos CPM
For l = 0 To mProj(i).QtdAtiv - 1
    For k = 0 To mProj(i).QtdRec - 1
        If supAtiv(l).NecesRec(k) > 0 Then
            For j = mCPM(l).ES To mCPM(l).ES + supAtiv(l).Dura - 1
                mMatrizRecCPM_ES(k).ConsumoRecursos(j) =
mMatrizRecCPM_ES(k).ConsumoRecursos(j) + supAtiv(l).NecesRec(k)
                If mMatrizRecCPM_ES(k).Dispon <
mMatrizRecCPM_ES(k).ConsumoRecursos(j) Then
                    mMatrizRecCPM_ES(k).Dispon =
mMatrizRecCPM_ES(k).ConsumoRecursos(j)
                End If
            Next
        End If
    Next
Next

End Sub

Sub CalculaMakespan_LBRec()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
Dim viab_rec As Long
i = Inst_Atual

'acha LB dos recursos - maior necessidade de uma unica atividade e calcula
média da necessidade de recursos
For k = 0 To mProj(i).QtdRec - 1
    For l = 0 To mProj(i).QtdAtiv - 1
        'acha LB dos recursos
        If mMatrRecLB(k).Dispon < supAtiv(l).NecesRec(k) Then
            mMatrRecLB(k).Dispon = supAtiv(l).NecesRec(k)
        End If
        'calcula média da necessidade de recursos e custo de recursos
        If supAtiv(l).NecesRec(k) > 0 Then
            mMatrRecMelhorSol(k).SomaRecursos =
mMatrRecMelhorSol(k).SomaRecursos + supAtiv(l).NecesRec(k) *
supAtiv(l).Dura
        End If
    Next
    mMatrRecMelhorSol(k).MediaNecess =
WorksheetFunction.RoundUp(mMatrRecMelhorSol(k).SomaRecursos / DataEntrega,
0)
    'se a media de necessidade de recursos for menor que o LB de recursos,
atualiza o valor para LB
    If mMatrRecMelhorSol(k).MediaNecess < mMatrRecLB(k).Dispon Then
        mMatrRecMelhorSol(k).MediaNecess = mMatrRecLB(k).Dispon
    End If
Next
'calcula FO_LB e Projeto com baixa utilização de recursos
FO_LB = 0

```

```

ProjetoBaixoRecurso = 0
For k = 0 To mProj(i).QtdRec - 1
    FO_LB = FO_LB + mMatrRecLB(k).Dispon * mProj(i).CustoRec(k)
    If mMatrRecMelhorSol(k).MediaNecess = mMatrRecLB(k).Dispon Then
        ProjetoBaixoRecurso = ProjetoBaixoRecurso + 1
    End If
Next

End Sub

Sub QS_Recurso_Desc(iMin As Long, iMax As Long)
'ordena a matriz ListaRecurso em ordem decrescente pela key
string(Pico/Media / Cod Recurso)
Dim i As Long, j As Long
Dim vMeio As String
Dim tReg As String

i = iMin: j = iMax
vMeio = ListaRecurso((i + j) \ 2)
Do While i <= j
    Do While ListaRecurso(i) > vMeio And i <= j
        i = i + 1
    Loop
    Do While ListaRecurso(j) < vMeio And j >= i
        j = j - 1
    Loop
    If i <= j Then 'swap
        tReg = ListaRecurso(i)
        ListaRecurso(i) = ListaRecurso(j)
        ListaRecurso(j) = tReg
        i = i + 1
        j = j - 1
    End If
Loop
If iMin < j Then QS_Recurso_Desc iMin, j
If i < iMax Then QS_Recurso_Desc i, iMax

End Sub

Sub CalculaFlexRec()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
Dim pico_ok As Long
i = Inst_Atual

'limpa dados de Simulação de Disponibilidade de recursos - considera
somente dados das atividades já programadas
For k = 0 To mProj(i).QtdRec - 1
    pico_ok = -1
    mMatrRecSimul(k).Dispon = mMatrRecSolAtual(k).Dispon
    For j = 0 To DataLimite
        mMatrRecSimul(k).ConsumoRecurso(j) =
mMatrRecSolAtual(k).ConsumoRecurso(j)
        'guarda data inicio do pico
    
```

```

        If mMatrRecSimul(k).ConsumoRecursos(j) = mMatrRecSimul(k).Dispon
And pico_ok = -1 Then
            mMatrRecSimul(k).InicioPico = j
            mMatrRecSimul(k).FimPico = j
            pico_ok = 0
        End If
    Next
Next

'simula a disponibilidade de recursos com a data mais cedo das atividades
não programadas considerando o limite %pico/media
For l = 0 To mProj(i).QtdAtiv - 1
    If ListaAtivNaoProgr(l) = 1 Then
        For k = 0 To mProj(i).QtdRec - 1
            If supAtiv(l).NecesRec(k) > 0 Then
                For j = mCPM(l).ES To mCPM(l).ES + supAtiv(l).Dura - 1
                    mMatrRecSimul(k).ConsumoRecursos(j) =
mMatrRecSimul(k).ConsumoRecursos(j) + supAtiv(l).NecesRec(k)
                    If mMatrRecSimul(k).Dispon <
mMatrRecSimul(k).ConsumoRecursos(j) Then
                        mMatrRecSimul(k).Dispon =
mMatrRecSimul(k).ConsumoRecursos(j)
                        'guarda data inicio do pico
                        mMatrRecSimul(k).InicioPico = j
                        mMatrRecSimul(k).FimPico = j
                    End If
                Next
            End If
        Next
    End If
Next
End If
Next

'acha data fim do pico
For k = 0 To mProj(i).QtdRec - 1
    For j = mMatrRecSimul(k).InicioPico + 1 To mCPM(mProj(i).QtdAtiv -
1).EF
        If mMatrRecSimul(k).ConsumoRecursos(j) = mMatrRecSimul(k).Dispon
Then
            mMatrRecSimul(k).FimPico = j
        Else
            j = DataLimite
        End If
    Next
Next

'calcula o % pico / média dos recursos
For k = 0 To mProj(i).QtdRec - 1
    'se data fim projeto > data entrega
    If mCPM(mProj(i).QtdAtiv - 1).EF > DataEntrega Then
        ListaRecursos(k) = Right("000000" &
Trim$(Round((mMatrRecSimul(k).Dispon / (mMatrRecMelhorSol(k).SomaRecursos
/ mCPM(mProj(i).QtdAtiv - 1).EF)) * 100 * mProj(i).CustoRec(k), 0)), 6) &
Right("000" & Trim$(k), 3)
    Else

```



```

        'se data fim projeto <= data entrega
            ListaRecursos(k) = Right("000000" &
Trim$(Round((mMatrRecSimul(k).Dispon / mMatrRecMelhorSol(k).MediaNecess) *
100 * mProj(i).CustoRec(k), 0)), 6) & Right("000" & Trim$(k), 3)
        End If
    Next

'ordena por maior flexibilidade (maior % pico / média)
QS_Recursos_Desc 0, mProj(i).QtdRec - 1

'calculo FO
mValorFO_Simul.CustoRecursos = 0
mValorFO_Simul.CustoAtraso = 0
'calcula valor FO recursos
For k = 0 To mProj(i).QtdRec - 1
    mValorFO_Simul.CustoRecursos = mValorFO_Simul.CustoRecursos +
mMatrRecSimul(k).Dispon * mProj(i).CustoRec(k)
Next
'adiciona valor atraso, se houver atraso
If mSimulCPM(mProj(i).QtdAtiv - 1).EF > DataEntrega Then
    mValorFO_Simul.CustoAtraso = mValorFO_Simul.CustoAtraso +
CustoUnitAtraso * (mSimulCPM(mProj(i).QtdAtiv - 1).EF - DataEntrega)
End If
'retorna valor total FO
mValorFO_Simul.CustoTotal = mValorFO_Simul.CustoRecursos +
mValorFO_Simul.CustoAtraso

End Sub

Sub QS_Atividades(iMin As Long, iMax As Long)
'ordena a matriz ListaAtivCand em ordem crescente pela key
string(Folga/Qtd Sucess/Ativ)
Dim tReg As rListaAtivCand
Dim j As Long, k As Long
Dim Meio As String, Lmin As Long, LMax As Long
Lmin = iMin
LMax = iMax
k = iMin
j = iMax
Meio = mListaAtivCand((Lmin + LMax) \ 2).Chave
Do
    Do While mListaAtivCand(k).Chave < Meio
        k = k + 1
    Loop
    Do While Meio < mListaAtivCand(j).Chave
        j = j - 1
    Loop
    If k < j Then
        tReg = mListaAtivCand(k)
        mListaAtivCand(k) = mListaAtivCand(j)
        mListaAtivCand(j) = tReg
    End If
    If k <= j Then
        k = k + 1
    End If

```

```

        j = j - 1
    End If
Loop Until k > j
If iMin < j Then QS_Atividades iMin, j
If k < iMax Then QS_Atividades k, iMax
End Sub

Sub CalculaFlexAtiv()

Dim i As Long, j As Long, k As Long, l As Long, m As Long
Dim soma_prob As Double
Dim sup As Long
i = Inst_Atual
soma_prob = 0
sup = 0

'ordena atividades candidatas pela flexibilidade
QS_Atividades 0, cont_cand - 1

'ordena atividades candidatas novamente, mas agora considerando também a
ordem da flexibilidade dos recursos
'primeiro atualiza a chave - ordem do recurso + ordem da atividade
For k = 0 To mProj(i).QtdRec - 1
    For l = 0 To cont_cand - 1
        If
(supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(Right(ListaRecursos(k), 3) *
1) > 0) And mListaAtivCand(l).Chave_OK <> 0 Then
            mListaAtivCand(l).Chave = Right("000000" & k, 6) & Right("000"
& l, 3)
            mListaAtivCand(l).Chave_OK = 0
            sup = sup + 1
        End If
        If sup = cont_cand Then
            k = mProj(i).QtdRec
            l = cont_cand
        End If
    Next
Next
Next
'ordena atividades candidatas pela flexibilidade de recursos e atividades
QS_Atividades 0, cont_cand - 1

'calcula probabilidade de cada atividade - vies
If cont_cand = 1 Then
    mListaAtivCand(0).Probab = 1
Else
    'calcula probabilidade
    For l = 0 To cont_cand - 1
        mListaAtivCand(l).Probab = 0.8 ^ (2 * l + 1)
        soma_prob = soma_prob + mListaAtivCand(l).Probab
    Next
    'normaliza probabilidade
    For l = 0 To cont_cand - 1
        mListaAtivCand(l).Probab = mListaAtivCand(l).Probab / soma_prob
    Next

```

```

        'atualiza probabilidade
        For l = 1 To cont_cand - 1
            mListaAtivCand(l).Probab = mListaAtivCand(l).Probab +
mListaAtivCand(l - 1).Probab
        Next
    End If

End Sub

Function AchaMelhorJ(cand As Long, recurso As Long) As Long

Dim i As Long, k As Long, l As Long, m As Long
Dim sim_proxj As Long
Dim melhor_j As Long
i = Inst_Atual

'começa simulando com data inicio da candidata igual ao tempo mais cedo+1,
já que os valores atuais sao considerando data mais cedo
sim_proxj = mMatrRecSimul(recurso).FimPico + 1
melhor_j = -1

'procura dt inicio da candidata enqto estiver no pico
While melhor_j = -1
    'recalcula ES considerando tempo inicio da candidata
    For k = 1 To mProj(i).QtdAtiv - 1
        'limpa dados ES, EF, LS, LF
        mSimulCPM(k).ES = 0
        mSimulCPM(k).EF = 0
        'se for atividade candidata, considera sim_proxj
        If k = cand Then
            mSimulCPM(k).ES = sim_proxj
        Else
            'se atividade ainda não foi programada, calcula ES com base
nas atividades precedentes
            If mSolAtivAtual(k).TFim = -1 Then
                For l = 0 To supAtiv(k).QtdPreced - 1
                    If mSimulCPM(k).ES <
mSimulCPM(supAtiv(k).Preced(l)).EF Then
                        mSimulCPM(k).ES =
mSimulCPM(supAtiv(k).Preced(l)).EF
                    End If
                Next
            Else
                'se atividade já foi programada, considera ES = Data
Inicio
                mSimulCPM(k).ES = mSolAtivAtual(k).TInicio
            End If
        End If
        mSimulCPM(k).EF = mSimulCPM(k).ES + supAtiv(k).Dura
    Next
    'simula matriz de consumo de recursos do recurso analisado e acha data
inicio pico
    'limpa dados de Simulação de Disponibilidade de recursos - considera
somente dados das atividades já programadas

```

```

    mMatrRecSimul(recurso).InicioPico = 0
    mMatrRecSimul(recurso).FimPico = 0
    mMatrRecSimul(recurso).Dispon = mMatrRecSolAtual(recurso).Dispon
    For j = 0 To DataLimite
        mMatrRecSimul(recurso).ConsumoRecursos(j) =
mMatrRecSolAtual(recurso).ConsumoRecursos(j)
    Next
    'simula a disponibilidade de recursos com a data mais cedo das
atividades não programadas considerando o limite %pico/media
    For l = 0 To mProj(i).QtdAtiv - 1
        If ListaAtivNaoProgr(l) = 1 Then
            If supAtiv(l).NecesRec(recurso) > 0 Then
                For j = mSimulCPM(l).ES To mSimulCPM(l).ES +
supAtiv(l).Dura - 1
                    mMatrRecSimul(recurso).ConsumoRecursos(j) =
mMatrRecSimul(recurso).ConsumoRecursos(j) + supAtiv(l).NecesRec(recurso)
                    If mMatrRecSimul(recurso).Dispon <
mMatrRecSimul(recurso).ConsumoRecursos(j) Then
                        mMatrRecSimul(recurso).Dispon =
mMatrRecSimul(recurso).ConsumoRecursos(j)
                        mMatrRecSimul(recurso).InicioPico = j
                        mMatrRecSimul(recurso).FimPico = j
                    End If
                Next
            End If
        End If
    Next
    'acha data fim do pico
    For j = mMatrRecSimul(recurso).InicioPico + 1 To
mSimulCPM(mProj(i).QtdAtiv - 1).EF 'j instante de tempo
        If mMatrRecSimul(recurso).ConsumoRecursos(j) =
mMatrRecSimul(recurso).Dispon Then
            mMatrRecSimul(recurso).FimPico = j
        Else
            j = DataLimite
        End If
    Next
    'se dt inicio pico >= dt fim candidata --> programa candidata, se nao,
simula no proximo instante de tempo
    If mMatrRecSimul(recurso).InicioPico >= mSimulCPM(cand).EF Or
mMatrRecSimul(recurso).FimPico < mSimulCPM(cand).ES Or
mMatrRecSimul(recurso).Dispon = mMatrRecLB(recurso).Dispon Then
        melhor_j = sim_proxj
    Else
        sim_proxj = mMatrRecSimul(recurso).FimPico + 1
    End If
Wend

'retorna melhor j
AchaMelhorJ = melhor_j

End Function

Function SimulaFO(cand As Long, data As Long) As Long

```

```

Dim i As Long, j As Long, k As Long, l As Long, m As Long
i = Inst_Atual

'recalcula ES considerando tempo inicio da candidata
For k = 1 To mProj(i).QtdAtiv - 1
    'limpa dados ES, EF, LS, LF
    mSimulCPM(k).ES = 0
    mSimulCPM(k).EF = 0
    'se for atividade candidata, considera sim_proxj
    If k = cand Then
        mSimulCPM(k).ES = data
    Else
        'se atividade ainda não foi programada, calcula ES com base nas
atividades precedentes
        If mSolAtivAtual(k).TFim = -1 Then
            For l = 0 To supAtiv(k).QtdPreced - 1
                If mSimulCPM(k).ES < mSimulCPM(supAtiv(k).Preced(l)).EF
Then
                    mSimulCPM(k).ES = mSimulCPM(supAtiv(k).Preced(l)).EF
                End If
            Next
        Else
            'se atividade já foi programada, considera ES = Data Inicio
            mSimulCPM(k).ES = mSolAtivAtual(k).TInicio
        End If
    End If
    mSimulCPM(k).EF = mSimulCPM(k).ES + supAtiv(k).Dura
Next

'limpa dados de Simulação de Disponibilidade de recursos - considera
somente dados das atividades já programadas
For k = 0 To mProj(i).QtdRec - 1
    For j = 0 To DataLimite
        mMatrRecSimul(k).ConsumoRecursos(j) =
mMatrRecSolAtual(k).ConsumoRecursos(j)
    Next
    mMatrRecSimul(k).Dispon = mMatrRecSolAtual(k).Dispon
Next

'simula a disponibilidade de recursos com a data mais cedo das atividades
não programadas considerando o limite %pico/media
For l = 0 To mProj(i).QtdAtiv - 1
    If ListaAtivNaoProgr(l) = 1 Then
        For k = 0 To mProj(i).QtdRec - 1
            If supAtiv(l).NecesRec(k) > 0 Then
                For j = mSimulCPM(l).ES To mSimulCPM(l).ES +
supAtiv(l).Dura - 1
                    'j instante de tempo
                    mMatrRecSimul(k).ConsumoRecursos(j) =
mMatrRecSimul(k).ConsumoRecursos(j) + supAtiv(l).NecesRec(k)
                    If mMatrRecSimul(k).Dispon <
mMatrRecSimul(k).ConsumoRecursos(j) Then
                        mMatrRecSimul(k).Dispon =
mMatrRecSimul(k).ConsumoRecursos(j)
                    End If
                Next
            End If
        Next
    End If
Next

```

```

        End If
    Next
End If
Next
End If
Next

'calculo FO
mValorFO_Simul.CustoRecursos = 0
mValorFO_Simul.CustoAtraso = 0
'calcula valor FO recursos
For k = 0 To mProj(i).QtdRec - 1
    mValorFO_Simul.CustoRecursos = mValorFO_Simul.CustoRecursos +
mMatrRecSimul(k).Dispon * mProj(i).CustoRec(k)
Next
'adiciona valor atraso, se houver atraso
If mSimulCPM(mProj(i).QtdAtiv - 1).EF > DataEntrega Then
    mValorFO_Simul.CustoAtraso = mValorFO_Simul.CustoAtraso +
CustoUnitAtraso * (mSimulCPM(mProj(i).QtdAtiv - 1).EF - DataEntrega)
End If
'retorna valor total FO
SimulaFO = mValorFO_Simul.CustoRecursos + mValorFO_Simul.CustoAtraso

End Function

Sub Backward_Forward()

Dim i As Long, j As Long, k As Long, l As Long, m As Long, n As Long, q As
Long
Dim teste As Long
teste = 1
i = Inst_Atual

'Backward
'ordena a lista de atividades em ordem decrescente de TFim + decrescente
IdAtiv
'cria a lista
For l = 0 To mProj(i).QtdAtiv - 1
    mListaAtivCand(l).Chave = Right("000000" & Trim$(999 -
mSolAtivAtual(l).TFim), 6) & Right("000" & Trim$(999 - supAtiv(l).IdAtiv),
3)
    mListaAtivCand(l).IdAtiv = supAtiv(l).IdAtiv
    mListaAtivCand(l).Probab = 0
    mListaAtivCand(l).Chave_OK = -1
    'limpa dados ES, EF simulação - onde guarda dt inicio e fim da
simulação
    mSimulCPM(l).ES = -1
    mSimulCPM(l).EF = 0
Next
'ordena a lista
QS_Atividades 0, mProj(i).QtdAtiv - 1

'limpa dados de Simulação de Disponibilidade de recursos

```

```

For k = 0 To mProj(i).QtdRec - 1
    For j = 0 To DataLimite
        mMatrRecSimul(k).ConsumoRecursos(j) = 0
    Next
    mMatrRecSimul(k).Dispon = mMatrRecSolAtual(k).Dispon
Next

'programa no tempo mais tarde seguindo a lista, considerando Data Fim do
projeto original
'acha a menor data de inicio possivel, considerando atividades sucessoras
mSimulCPM(mProj(i).QtdAtiv - 1).ES = mSolAtivAtual(mProj(i).QtdAtiv -
1).TInicio
mSimulCPM(mProj(i).QtdAtiv - 1).EF = mSolAtivAtual(mProj(i).QtdAtiv -
1).TFim
For l = 1 To mProj(i).QtdAtiv - 1
    If mListaAtivCand(l).IdAtiv = 0 Then
        mListaAtivCand(l).IdAtiv = 0
    End If
    j = mSimulCPM(supAtiv(mListaAtivCand(l).IdAtiv).Sucess(1)).ES
    If supAtiv(mListaAtivCand(l).IdAtiv).QtdSucess > 1 Then
        For n = 2 To 3
            If supAtiv(mListaAtivCand(l).IdAtiv).Sucess(n) <> -1 Then
                If j >
mSimulCPM(supAtiv(mListaAtivCand(l).IdAtiv).Sucess(n)).ES Then
                    j =
mSimulCPM(supAtiv(mListaAtivCand(l).IdAtiv).Sucess(n)).ES
                End If
            End If
        Next
    End If
    While mSimulCPM(mListaAtivCand(l).IdAtiv).ES = -1
        estoura_rec = 0
        For q = 0 To mProj(i).QtdRec - 1
            If supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(q) > 0 Then
                For m = j - supAtiv(mListaAtivCand(l).IdAtiv).Dura To j -
1
                    If (mMatrRecSimul(q).ConsumoRecursos(m) +
supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(q)) >
mMatrRecSolAtual(q).Dispon Then
                        estoura_rec = -1
                        m = j
                        q = mProj(i).QtdRec
                    End If
                Next
            End If
        Next
        If estoura_rec = 0 Then
            mSimulCPM(mListaAtivCand(l).IdAtiv).ES = j -
supAtiv(mListaAtivCand(l).IdAtiv).Dura
        Else
            j = j - 1
        End If
    Wend

```

```

        mSimulCPM(mListaAtivCand(l).IdAtiv).EF =
mSimulCPM(mListaAtivCand(l).IdAtiv).ES +
supAtiv(mListaAtivCand(l).IdAtiv).Dura
        For k = 0 To mProj(i).QtdRec - 1
            If supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(k) > 0 Then
                For m = mSimulCPM(mListaAtivCand(l).IdAtiv).ES To
mSimulCPM(mListaAtivCand(l).IdAtiv).EF - 1
                    mMatrRecSimul(k).ConsumoRecursos(m) =
mMatrRecSimul(k).ConsumoRecursos(m) +
supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(k)
                Next
            End If
        Next
    Next
Next

'Forward
'ordena a lista de atividades em ordem crescente de TInicio + crescente
IdAtiv
'cria a lista
For l = 0 To mProj(i).QtdAtiv - 1
    mListaAtivCand(l).Chave = Right("000000" & Trim$(mSimulCPM(l).ES), 6)
& Right("000" & Trim$(supAtiv(l).IdAtiv), 3)
    mListaAtivCand(l).IdAtiv = supAtiv(l).IdAtiv
    mListaAtivCand(l).Probab = 0
    mListaAtivCand(l).Chave_OK = -1
    'limpa dados ES, EF simulação - onde guarda dt inicio e fim da
simulação
    mSolAtivAtual(l).TInicio = -1
    mSolAtivAtual(l).TFim = 0
Next
'ordena a lista
QS_Atividades 0, mProj(i).QtdAtiv - 1

'limpa dados de Simulação de Disponibilidade de recursos
For k = 0 To mProj(i).QtdRec - 1
    For j = 0 To DataLimite
        mMatrRecSolAtual(k).ConsumoRecursos(j) = 0
    Next
    mMatrRecSolAtual(k).Dispon = 0
Next

'programa no tempo mais cedo seguindo a lista
'acha a menor data de inicio possivel, considerando atividades precedentes
mSolAtivAtual(0).TFim = 0
mSolAtivAtual(0).TInicio = 0
For l = 1 To mProj(i).QtdAtiv - 1
    j = 0
    For n = 0 To supAtiv(mListaAtivCand(l).IdAtiv).QtdPreced - 1
        If j <
mSolAtivAtual(supAtiv(mListaAtivCand(l).IdAtiv).Preced(n)).TFim Then
            j =
mSolAtivAtual(supAtiv(mListaAtivCand(l).IdAtiv).Preced(n)).TFim
        End If
    Next
Next

```



```

While mSolAtivAtual(mListaAtivCand(l).IdAtiv).TInicio = -1
    estoura_rec = 0
    For q = 0 To mProj(i).QtdRec - 1
        If supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(q) > 0 Then
            For m = j To j + supAtiv(mListaAtivCand(l).IdAtiv).Dura -
1
                If (mMatrRecSolAtual(q).ConsumoRecursos(m) +
supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(q)) > mMatrRecSimul(q).Dispon
Then
                    estoura_rec = -1
                    m = j + supAtiv(mListaAtivCand(l).IdAtiv).Dura
                    q = mProj(i).QtdRec
                End If
            Next
        End If
    Next
    If estoura_rec = 0 Then
        mSolAtivAtual(mListaAtivCand(l).IdAtiv).TInicio = j
    Else
        j = j + 1
    End If
Wend
mSolAtivAtual(mListaAtivCand(l).IdAtiv).TFim =
mSolAtivAtual(mListaAtivCand(l).IdAtiv).TInicio +
supAtiv(mListaAtivCand(l).IdAtiv).Dura
For k = 0 To mProj(i).QtdRec - 1
    If supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(k) > 0 Then
        For m = mSolAtivAtual(mListaAtivCand(l).IdAtiv).TInicio To
mSolAtivAtual(mListaAtivCand(l).IdAtiv).TFim - 1
            mMatrRecSolAtual(k).ConsumoRecursos(m) =
mMatrRecSolAtual(k).ConsumoRecursos(m) +
supAtiv(mListaAtivCand(l).IdAtiv).NecesRec(k)
            If mMatrRecSolAtual(k).ConsumoRecursos(m) >
mMatrRecSolAtual(k).Dispon Then
                mMatrRecSolAtual(k).Dispon =
mMatrRecSolAtual(k).ConsumoRecursos(m)
            End If
        Next
    End If
Next
Next
End Sub

Sub Inicia()

Application.ScreenUpdating = False
Application.Calculation = xlCalculationManual
Application.EnableEvents = False

Dim i As Long, j As Long, k As Long, l As Long, m As Long, n As Long, o As
Long, p As Long, q As Long
Dim sup_cand As Long
Dim cont_rec As Long

```

```

Dim estoura_rec As Long
Dim sort As Double
Dim NumIterMelhorSol As Long
Dim sup_data As Long
Dim CustoES As Double
Dim CustoComAtraso As Double
Dim ultimoTFim As Long
Dim n_limite As Long
Dim NumUltIteracao
Dim sup_ciclo
Dim HoraMelhorSol As Date

```

CarregaDados

```

cont_rec = 2
cont_ativ = 2
NumIterMelhorSol = -1
NumUltIteracao = -1
n_limite = 500000

```

```

For i = 1 To QtdInst
    Inst_Atual = i
    sup_ciclo = 1
    'inicializa dados das soluções e outras informações para resolução do
    problema
    InicializaDados_Instancia
    InicializaDados_Solucao
    'calcula solução CPM
    CalculaCPM_Parte1
    'calcula makespan cpm, data entrega do projeto e matriz de recursos
    para tempo mais cedo ES - referência de pico (UB)
    CalculaCPM_Parte2
    'acha LB dos recursos e media necessidade de recursos
    CalculaMakespan_LBRec

    'versão probabilística
    For n = 1 To n_limite
        If n > 1 Then
            InicializaDados_Solucao
        End If
        'ciclos 4 e 5 - explora vizinhança da melhor solução
        If (n >= (5000 + 25000 * sup_ciclo) And n < (10000 + 25000 *
sup_ciclo)) Then
            For q = 0 To mProj(i).QtdRec - 1
                mMatrRecSolAtual(q).PercPicoMedia =
mMatrRecMelhorSol(q).Dispon
            Next
        Else
            If (n >= 10000 And ProjetoBaixoRecurso < mProj(i).QtdRec) Or
(n >= 5000 And ProjetoBaixoRecurso = mProj(i).QtdRec) Then
                For q = 0 To mProj(i).QtdRec - 1
                    If mProj(i).CustoRec(q) > (CustoUnitAtraso * 0.75)
Then

```

```

        mMatrRecSolAtual(q).PercPicoMedia = 5 * Rnd +
mMatrRecMelhorSol(q).Dispon - 2
    Else
        mMatrRecSolAtual(q).PercPicoMedia = 7 * Rnd +
mMatrRecMelhorSol(q).Dispon - 3
    End If
    If mMatrRecSolAtual(q).PercPicoMedia <
mMatrRecLB(q).Dispon Then
        mMatrRecSolAtual(q).PercPicoMedia =
mMatrRecLB(q).Dispon
    End If
    If mMatrRecSolAtual(q).PercPicoMedia >
mMatrizRecCPM_ES(q).Dispon Then
        mMatrRecSolAtual(q).PercPicoMedia =
mMatrizRecCPM_ES(q).Dispon
    End If
Next
End If
End If

'heurística construtiva - programa as atividades segundo
flexibilidade de atividades e dos recursos
'enquanto tempo menor que data limite de termino do projeto e
atividade ficticia final nao for programada
While mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim = -1
    'atualiza ES, EF, LS, LF e folgas
    CalculaCPM_Parte1
    'acha lista de atividades candidatas - sem precedencia ou com
precedentes já programadas
    cont_cand = 0
    For l = 0 To mProj(i).QtdAtiv - 1
        If ListaAtivNaoProgr(l) = 1 Then
            sup_cand = 0
            If supAtiv(l).QtdPreced = 0 Then
                mListaAtivCand(cont_cand).Chave = Right("000000" &
Trim$(999999 - supAtiv(l).CRR), 6) & Right("000" & Trim$(mCPM(l).LF), 3) &
Right("000" & Trim$(supAtiv(l).IdAtiv), 3)
                mListaAtivCand(cont_cand).IdAtiv =
supAtiv(l).IdAtiv
                mListaAtivCand(cont_cand).Probab = 0
                mListaAtivCand(cont_cand).Chave_OK = -1
                cont_cand = cont_cand + 1
                sup_cand = -1
            Else
                For m = 0 To supAtiv(l).QtdPreced - 1
                    If ListaAtivNaoProgr(supAtiv(l).Preced(m)) =
supAtiv(l).Preced(m) Then
                        sup_cand = -1
                        m = supAtiv(l).QtdPreced
                    End If
                Next
            End If
        End If
    Next
End If
If sup_cand = 0 Then

```

```

                                mListaAtivCand(cont_cand).Chave = Right("000000" &
Trim$(999999 - supAtiv(1).CRR), 6) & Right("000" & Trim$(mCPM(1).LF), 3) &
Right("000" & Trim$(supAtiv(1).IdAtiv), 3)
                                mListaAtivCand(cont_cand).IdAtiv =
supAtiv(1).IdAtiv
                                mListaAtivCand(cont_cand).Probab = 0
                                mListaAtivCand(cont_cand).Chave_OK = -1
                                cont_cand = cont_cand + 1
                        End If
                End If
        Next

        'ordena recursos pela flexibilidade
        CalculaFlexRec
        'ordena atividades candidatas pela flexibilidade e calcula
probabilidade
        CalculaFlexAtiv

        'sorteia atividade candidata
        sort = Rnd
        For o = 0 To cont_cand
            If o = 0 Then
                'primeira atividade candidata - nao existe "o-1"
                If sort <= mListaAtivCand(o).Probab Then
                    sup_cand = mListaAtivCand(o).IdAtiv
                    o = cont_cand + 1
                End If
            Else
                If sort <= mListaAtivCand(o).Probab And sort >
mListaAtivCand(o - 1).Probab Then
                    sup_cand = mListaAtivCand(o).IdAtiv
                    o = cont_cand + 1
                End If
            End If
        Next

        'nas primeiras iteracoes, acha o tempo mais cedo fora do pico
        If (n < 10000 And ProjetoBaixoRecurso < mProj(i).QtdRec) Or (n
< 5000 And ProjetoBaixoRecurso = mProj(i).QtdRec) Then
            'no minimo, a data inicio da atividade é igual ao tempo
mais cedo
            mSolAtivAtual(sup_cand).TInicio = mCPM(sup_cand).ES
            'acha o melhor instante de tempo para programar o inicio da
atividade
            For q = 0 To mProj(i).QtdRec - 1
                sup_data = 0
                If supAtiv(sup_cand).NecesRec(q) > 0 And
(mProj(i).CustoRec(q) > (CustoUnitAtraso * 0.25)) Then 'Or
mProj(i).CustoRec(q) > CustoUnitAtraso * 0.1
                    'se nao ha pico (data fim pico = 0) ou se
atividade no tempo inicio atual está fora do pico --> data inicio
atividade = data inicio atual
                    If mMatrRecSimul(q).FimPico = 0 Or
(mMatrRecSimul(q).InicioPico >= (mSolAtivAtual(sup_cand).TInicio +

```

```

supAtiv(sup_cand).Dura)) Or (mMatrRecSimul(q).FimPico <
mSolAtivAtual(sup_cand).TInicio) Then
    sup_data = -1
Else
    'acha tempo de inicio no qual a atividade está
fora do pico
    sup_data = AchaMelhorJ(sup_cand, q)
    'se melhor data simulacao <= data mais tarde
inicio da atividade --> data inicio atividade = melhor data simulacao
(atraso=0)
    If sup_data <= mCPM(sup_cand).LS Then
        If mSolAtivAtual(sup_cand).TInicio <
sup_data Then
            mSolAtivAtual(sup_cand).TInicio =
sup_data
        End If
    Else
        'se TInicio atual < data com atraso, então
avalia se é melhor TInicio atual ou data com atraso
        If mSolAtivAtual(sup_cand).TInicio <
sup_data Then
            If mSolAtivAtual(sup_cand).TInicio =
mCPM(sup_cand).ES Then
                CustoES =
mValorFO_Simul.CustoTotal
            Else
                CustoES = SimulaFO(sup_cand,
mSolAtivAtual(sup_cand).TInicio)
            End If
            'calcula custo com TInicio = sup_data
            CustoComAtraso = SimulaFO(sup_cand,
sup_data)
            'se custo atraso < custo do recurso da
atividade --> data inicio atividade = melhor data simulacao (atraso>0)
            If CustoComAtraso < CustoES Then
                mSolAtivAtual(sup_cand).TInicio =
sup_data
            Else
                'se nao vale a pena atrasar -->
data inicio atividade = data mais cedo
                sup_data = -1
            End If
        End If
    End If
End If
End If
Next
Else
    'nas últimas iterações, acha a data de inicio da atividade
mais cedo sem estourar o pico da melhor solução até então
    j = mCPM(sup_cand).ES
    While mSolAtivAtual(sup_cand).TInicio = -1
        estoura_rec = 0
    End While
End If

```

```

        For q = 0 To mProj(i).QtdRec - 1
            If (((n >= (25000 * sup_ciclo - 15000) And n <
(25000 * sup_ciclo)) Or (n >= (5000 + 25000 * sup_ciclo) And n < (7500 +
25000 * sup_ciclo))) And supAtiv(sup_cand).NecesRec(q) > 0 And
(mProj(i).CustoRec(q) > (CustoUnitAtraso * 0.25))) Or (((n >= (25000 *
sup_ciclo) And n < (5000 + 25000 * sup_ciclo)) Or (n >= (7500 + 25000 *
sup_ciclo) And n < (10000 + 25000 * sup_ciclo))) And
supAtiv(sup_cand).NecesRec(q) > 0) Or (n >= 5000 And n < 10000 And
ProjetoBaixoRecurso = mProj(i).QtdRec And supAtiv(sup_cand).NecesRec(q) >
0 And (mProj(i).CustoRec(q) > (CustoUnitAtraso * 0.25)))) Then
                For m = j To j + supAtiv(sup_cand).Dura -
1
                    If
(mMatrRecSolAtual(q).ConsumoRecursos(m) + supAtiv(sup_cand).NecesRec(q)) >
mMatrRecSolAtual(q).PercPicoMedia Then
                        estoura_rec = -1
                        m = j + supAtiv(sup_cand).Dura
                        q = mProj(i).QtdRec
                    End If
                Next
            End If
        Next
    End If
Next
If estoura_rec = 0 Then
    mSolAtivAtual(sup_cand).TInicio = j
Else
    j = j + 1
End If
Wend
End If

mSolAtivAtual(sup_cand).TFim = mSolAtivAtual(sup_cand).TInicio
+ supAtiv(sup_cand).Dura
For k = 0 To mProj(i).QtdRec - 1
    If supAtiv(sup_cand).NecesRec(k) > 0 Then
        For m = mSolAtivAtual(sup_cand).TInicio To
mSolAtivAtual(sup_cand).TFim - 1
            mMatrRecSolAtual(k).ConsumoRecursos(m) =
mMatrRecSolAtual(k).ConsumoRecursos(m) + supAtiv(sup_cand).NecesRec(k)
            If mMatrRecSolAtual(k).Dispon <
mMatrRecSolAtual(k).ConsumoRecursos(m) Then
                mMatrRecSolAtual(k).Dispon =
mMatrRecSolAtual(k).ConsumoRecursos(m)
            End If
        Next
    End If
Next
End If
Next
ListaAtivNaoProgr(sup_cand) = -1

'verifica se custo atual já é pior do que a melhor solução. se
sim, já vai para a proxima iteração
mValorFO_SolAtual.CustoRecursos = 0
'calcula valor FO recursos
For k = 0 To mProj(i).QtdRec - 1

```

```

        mValorFO_SolAtual.CustoRecursos =
mValorFO_SolAtual.CustoRecursos + mMatrRecSolAtual(k).Dispon *
mProj(i).CustoRec(k)
    Next
    'verifica se custo atual já é pior do que a melhor solução. se
sim, já vai para a próxima iteração
    If mValorFO_SolAtual.CustoRecursos >=
mValorFO_MelhorSol.CustoTotal Then
        mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim = -5
    End If
Wend

'só calcula FO se programa está completo
If mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim > 0 Then
    'tenta melhorar a data fim do projeto, se atraso > 0
    ultimoTFim = 999
    While ultimoTFim > mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim
And mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim > DataEntrega
        ultimoTFim = mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim
        Backward_Forward
    Wend

    'calcula FO solução atual melhorada
    mValorFO_SolAtual.CustoRecursos = 0
    mValorFO_SolAtual.CustoAtraso = 0
    'calcula valor FO recursos
    For k = 0 To mProj(i).QtdRec - 1
        mValorFO_SolAtual.CustoRecursos =
mValorFO_SolAtual.CustoRecursos + mMatrRecSolAtual(k).Dispon *
mProj(i).CustoRec(k)
    Next
    'adiciona valor atraso, se houver atraso
    If mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim > DataEntrega Then
        mValorFO_SolAtual.CustoAtraso = CustoUnitAtraso *
(mSolAtivAtual(mProj(i).QtdAtiv - 1).TFim - DataEntrega)
    End If
    'calcula valor total FO
    mValorFO_SolAtual.CustoTotal = mValorFO_SolAtual.CustoRecursos
+ mValorFO_SolAtual.CustoAtraso

    'verifica se é a melhor solução até agora, se sim, atualiza
dados melhor solução
    If mValorFO_SolAtual.CustoTotal <
mValorFO_MelhorSol.CustoTotal Then
        HoraMelhorSol = Now
        For l = 0 To mProj(i).QtdAtiv - 1
            mMelhorSolAtiv(l).IdAtiv = l
            mMelhorSolAtiv(l).TFim = mSolAtivAtual(l).TFim
            mMelhorSolAtiv(l).TInicio = mSolAtivAtual(l).TInicio
        Next
        For k = 0 To mProj(i).QtdRec - 1
            mMatrRecMelhorSol(k).IdRec = k
            mMatrRecMelhorSol(k).Dispon =
mMatrRecSolAtual(k).Dispon

```

```

        Next
        mValorFO_MelhorSol.CustoTotal =
mValorFO_SolAtual.CustoTotal
        mValorFO_MelhorSol.CustoRecursos =
mValorFO_SolAtual.CustoRecursos
        mValorFO_MelhorSol.CustoAtraso =
mValorFO_SolAtual.CustoAtraso
        NumIterMelhorSol = n
    End If
    'se solução tem o menor custo possível, encerra instancia
    If FO_LB = mValorFO_MelhorSol.CustoTotal Or (n >
NumIterMelhorSol + 75000 And n > 85000 And ProjetoBaixoRecurso =
mProj(i).QtdRec) Then 'Or (n > NumIterMelhorSol + 125000 And
ProjetoBaixoRecurso < mProj(i).QtdRec)
        NumUltIteracao = n
        n = n_limite
    Else
        'se solução tem o menor custo de recursos possível, vai
para busca na vizinhança
        If FO_LB = mValorFO_MelhorSol.CustoRecursos And n =
NumIterMelhorSol And n < (5000 + 25000 * sup_ciclo) Then
            n = (5000 + 25000 * sup_ciclo)
        End If
    End If
End If

    'grava horario termino execução
    HoraFim = Now
    'verifica se já estourou limite de tempo de execução do programa,
se sim, atualiza sup_tempo
    If DateDiff("s", HoraInicio, HoraFim) > (15 * 60) Then
        NumUltIteracao = n
        n = n_limite
    End If
    If n = (10000 + 25000 * sup_ciclo) - 1 Then
        sup_ciclo = sup_ciclo + 1
    End If
Next

'gera resultados
Sheets("Result_FO").Select
Cells(i + 1, 1) = mProj(i).IdProj
Cells(i + 1, 2) = mMelhorSolAtiv(mProj(i).QtdAtiv - 1).TFim
Cells(i + 1, 3) = mValorFO_MelhorSol.CustoTotal
Cells(i + 1, 4) = DataEntrega
Cells(i + 1, 5) = mValorFO_MelhorSol.CustoRecursos
Cells(i + 1, 6) = mValorFO_MelhorSol.CustoAtraso
Cells(i + 1, 7) = NumIterMelhorSol
Cells(i + 1, 8) = NumUltIteracao
Cells(i + 1, 9) = DateDiff("s", HoraInicio, HoraMelhorSol)
Cells(i + 1, 10) = DateDiff("s", HoraInicio, HoraFim)

Sheets("Result_Rec").Select
For k = 0 To mProj(i).QtdRec - 1

```



```

        Cells(cont_rec, 1) = mProj(i).IdProj
        Cells(cont_rec, 2) = k + 1
        Cells(cont_rec, 3) = mMatrRecMelhorSol(k).Dispon
        cont_rec = cont_rec + 1
    Next

    Sheets("Result_Ativ").Select
    For l = 0 To mProj(i).QtdAtiv - 1
        Cells(cont_ativ, 1) = mProj(i).IdProj
        Cells(cont_ativ, 2) = l
        Cells(cont_ativ, 3) = mMelhorSolAtiv(l).TInicio
        Cells(cont_ativ, 4) = supAtiv(l).Dura
        Cells(cont_ativ, 5) = mMelhorSolAtiv(l).TFim
        cont_ativ = cont_ativ + 1
    Next
Next

Sheets("Result_F0").Select
Cells(1, 16) = "Horário Término"
Cells(2, 16) = Now

Cells.Select
Cells.EntireColumn.AutoFit

Application.ScreenUpdating = True
Application.Calculation = xlCalculationAutomatic
Application.EnableEvents = True

End Sub

```

APÊNDICE B – Código fonte para os experimentos no CPLEX através do IBM ILOG CPLEX Optimization Studio

```
//mod Begin
//TUPLAS E RANGES PARA INPUTS DA RODADA
//cria mRun
range Rr=1..1;
tuple Run {
int iBeg;
int iEnd;
int TiLim;}
Run mRun[Rr]=...;

//RANGES INPUTS DO OPL Global
//cria mQtdGl
range Rq=1..1;
tuple QtdGl {
int iQtdGl;
int mQtdGl;
int jQtdGl;
int jmQtdGl;
int ijQtdGl;}
QtdGl mQtdGl[Rq]=...;

//cria ranges
range Ri=1..mQtdGl[1].iQtdGl; range Rm=1..mQtdGl[1].mQtdGl;
range Rj=1..mQtdGl[1].jQtdGl; range Rjm=1..mQtdGl[1].jmQtdGl;
range Rij=1..mQtdGl[1].ijQtdGl;

//TUPLAS INPUTS DAS INSTANCIAS
//cria mTab
tuple oi {
int Inst;
int QtdFx2;
int bFx2;
int eFx2;
int QtdFx3;
int bFx3;
int eFx3;
int QtdFx4;
int bFx4;
int eFx4;
int QtdFx5;
int bFx5;
int eFx5;
float Di;
float ct;
float Mx0;}
oi moi[Ri]=...;

//cria om
tuple om {
int Inst;
int m;
```

```

int Nm;
float Rm;
float cm;
float NmMax;}
om mom [Rm]=...;

//cria oj
tuple oj {
int Inst;
int j;
float Pj;
float Rj;}
oj moj [Rj]=...;

//cria ojm
tuple ojm {
int Inst;
int j;
int m;
int Njm;
int Linm;}
ojm mojm [Rjm]=...;

//cria oij
tuple oij {
int Inst;
int i;
int j;}
oij moij [Rij]=...;

//PARAMETROS DO CPLEX
execute PARAMS {cplex.tilim=mRun[1].TiLim;}

//DADOS GLOBAIS
//passa instancia para o Mod
int Insts[1..1]=...; int viBeg=mRun[1].iBeg;
int InstLin=Insts[1]-viBeg +1;

//DADOS DA INSTANCIA
float vMx0=moi[InstLin].Mx0;

//VARIAVEIS DE DECISAO
dvar boolean Z1[Rj][Rj]; dvar boolean Z2[Rj][Rj];
dvar boolean Z3[Rj][Rj]; dvar boolean Z4[Rj][Rj];
dvar int Y[Rjm]; dvar float C[Rj];
dvar float PCT;

dvar float PTC;
dvar int Nm[Rm];
dvar float+ Ti;

//FUNCAO OBJETIVO
minimize PTC;

```

```
//RESTRICOES
```

```
subject to{
```

```
forall(Linj in Rj: moj[Linj].Inst==Insts[1]){
    C[moj[Linj].j]>=moj[Linj].Pj;
    PCT-C[moj[Linj].j]>=0;}

```

```
forall(Lini in Rj, Linj in Rj: Lini<Linj && moj[Lini].Inst==Insts[1] &&
moj[Linj].Inst==Insts[1]){
    C[moj[Linj].j] -C[moj[Lini].j] -moj[Linj].Pj +(1-
Z1[moj[Lini].j][moj[Linj].j])*vMx0 >=0;
    C[moj[Lini].j] -C[moj[Linj].j] -moj[Lini].Pj +(1-
Z2[moj[Lini].j][moj[Linj].j])*vMx0 >=0;
    Z1[moj[Lini].j][moj[Linj].j]+Z2[moj[Lini].j][moj[Linj].j]+Z3[moj[Lini]
].j][moj[Linj].j]+Z4[moj[Lini].j][moj[Linj].j]== 1;}

```

```
forall(Linim in Rjm, Linjm in Rjm: mojm[Linim].Inst==Insts[1] &&
mojm[Linjm].Inst==Insts[1] && mojm[Linim].m==mojm[Linjm].m){
    Y[Linjm] -Y[Linim] -mojm[Linjm].Njm +(1-
Z3[mojm[Linim].j][mojm[Linjm].j])*mom[mojm[Linjm].Linm].Nm>=0;
    Y[Linim] -Y[Linjm] -mojm[Linim].Njm +(1-
Z4[mojm[Linim].j][mojm[Linjm].j])*mom[mojm[Linim].Linm].Nm>=0;}

```

```
forall(Linij in Rij:
moij[Linij].Inst==Insts[1])Z1[moij[Linij].i][moij[Linij].j]==1;

```

```
forall(Linjm in Rjm, Linm in Rm: mojm[Linjm].Inst==Insts[1] &&
mom[Linm].Inst==Insts[1]){
    Y[Linjm]>=mojm[Linjm].Njm;
    Y[Linjm]<=Nm[mojm[Linjm].Linm];
}

Ti>=PCT-moi[InstLin].Di;
PTC>=sum(Linm in Rm: mom[Linm].Inst==Insts[1]) mom[Linm].cm*Nm[Linm]+
moi[InstLin].ct*Ti;

}
```