# Programação de rotação de culturas – modelos e métodos de solução

***Lana Mara Rodrigues dos Santos***

# Programação de rotação de culturas – modelos e métodos de solução

***Lana Mara Rodrigues dos Santos***

***Orientador: Prof. Dr. Marcos Nereu Arenales***

***Co–orientador: Prof. Dr. Philippe Yves Paul Michelon***

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências de Computação e Matemática Computacional.

**USP – São Carlos**
**Abril de 2009**

*Todo ponto de vista é a vista de algum ponto.*
*Leonardo Boff*

# Agradecimentos

Aos meus pais, irmãos, tios e tias pelo apoio e incentivo. Em especial, à minha tia, Elisabeth Rodrigues Bernardo, pelo suporte desde o segundo grau.

Ao meu orientador, Marcos Arenales, pela paciência, confiança, respeito e ensinamento nesses anos.

Ao meu orientador Philippe Michelon pelo estágio na belíssima *Provence* e pelo apoio que ele e sua família me deram naquele período.

Ao professor Alysson Machado Costa pelas inúmeras sugestões no texto da tese, pelo incentivo e pelo muito que aprendi com ele no meu último ano de doutorado.

Ao casal, Carlos Diego e Simone K. Rodrigues pela grande generosidade com que me receberam em Avignon.

Ao amigo Ricardo H. Santos, pelas dicas em sustentabilidade, ecologia e afins e pela parceria em diversos trabalhos.

Às professoras Maristela dos Santos e Franklina Braginon de Toledo pelas sugestões nesses anos.

Aos colegas do grupo de otimização do ICMC. Em particular à Kelly e ao Douglas pelo socorro em vários momentos.

Ao aluno Marcio Furlani F. Carmosa pela ajuda na implementação dos algoritmos e pelas dicas de computação.

À CAPES pelo apoio financeiro durante o doutorado e à região *Provence-Alpes-Côte d'Azur* pelo auxílio financeiro no período em que estagiei na França.

Aos meus amigos sempre presentes e que tornam a minha vida muito melhor.

# Resumo

Nas últimas décadas, diversas propostas de técnicas e de processos visando aumentar a sustentabilidade da agricultura ganharam evidência. Tais propostas geram novos modelos de planejamento em que devem ser considerados aspectos técnicos e ecológicos de produção, bem como o acesso de pequenos agricultores familiares ao mercado consumidor. Neste tipo de planejamento da produção, a rotação de culturas desempenha um papel fundamental, pois contribui para a manutenção dos recursos produtivos, para a minimização do uso de recursos não-renováveis e para o controle biológico da população de herbívoros, patógenos e plantas espontâneas.

Nesta tese abordamos dois problemas de *Programação de Rotação de Culturas* (PRC) focados na produção de base sustentável de hortaliças: o problema de PRC *com restrições de Adjacências* (PRC-A) e o problema de PRC *com atendimento da Demanda* (PRC-D). O planejamento da produção de hortaliças é complexo pois envolve, em geral, um grande número de culturas com limitações específicas quanto à época de plantio e com períodos de cultivo e produtividades muito variáveis. A programação de rotação de culturas para as áreas de plantio é formulada como um modelo de otimização 0-1 e, para os dois problemas, em cada programação considera–se tanto aspectos técnicos (época de plantio e colheita etc.) quanto ecológicos (adubação verde, pousio etc.).

No problema PRC-A o objetivo é a maximização da ocupação das áreas produtivas em que as restrições de plantio são estendidas às áreas adjacentes. Como a formulação matemática para o problema tem, em geral, um número muito grande de restrições e variáveis, com matriz de restrições esparsa e bloco-diagonal, o modelo é reformulado com a Decomposição Dantzig-Wolfe, o que permitiu sua resolução por procedimentos baseados em geração de colunas, heurísticos e exatos.

No problema PRC-D deseja-se suprir a demanda de um conjunto de hortaliças tendo-se disponível um conjunto de áreas heterogêneas. As culturas passíveis de plantio, bem como as suas produtividades, dependem da área considerada. O problema foi formulado como um modelo de otimização linear em que cada variável está associada a uma programação de rotação de culturas. O modelo contém potencialmente um número grande de programações de rotação e é resolvido por geração de colunas.

Experimentos computacionais usando instâncias baseadas em dados reais confirmam a eficácia dos modelos e das metodologias propostos para os problemas.

# Abstract

Over the last decades, various proposals for techniques and processes to increase agricultural sustainability have been put forward. These proposals bring new planning models in which technical and ecological production aspects must be considered, as well as the access of small farmers to the consumer market. In this type of agricultural production planning, crop rotation plays a fundamental role as it contributes to maintaining productive resources, to reducing the use of non-renewable resources, and to biologically controlling the population of herbivores, pathogens and spontaneous plants.

In this thesis, two problems concerning the *Crop Rotation Schedule* (CRS) focusing on sustainable production vegetables are addressed: the problem of the CRS *having Adjacent constraints* (CRS-A) and the problem of the CRS *under Demand constraints* (CRS-D). Production planning of vegetables is complex as it generally involves a large number of crop species having specific limitations regarding the planting season and very varied production times and productivity. The crop rotation schedule problem is formulated as an optimization model 0-1, and for both problems, in each schedule technical (planting and harvesting season etc.) and ecological (green manure, fallow etc.) aspects are considered.

Concerning the CRS-A problem, the aim is to maximize the occupation of cropping areas in which planting constraints are extended to adjacent areas. As the mathematical formulation for the problem generally has a large number of restrictions and variables and the structure of the constraint matrix of the problem is sparse and block-diagonal, the model has been reformulated using the Dantzig-Wolfe Decomposition strategy, which has enabled the use of a heuristic and exact procedures based on the column generation approach for its resolution.

In the CRS-D problem, the aim is to meet the market demands for vegetables having a set of heterogeneous cropping areas available. The potential planting crops, as well as their productivity, depend on the considered cropping area. The problem is formulated as an optimization linear model in which each variable is associated to a crop rotation schedule. The model may include a large number of rotation schedules and is solved by the column generation approach.

Computational experiments using instances based on real-world data confirm the efficiency of models and methodologies proposed for the problems.

# Resume

Les dernières décennies ont vu l'apparition de divers processus et techniques visant à développer et améliorer une agriculture durable. Ces nouveaux processus ont entraîné l'usage de nouveaux modèles de planification où sont considérés les aspects techniques et écologiques de la production agricole et doivent permettre à des agricultures familiales d'avoir accès aux consommateurs. Dans ces modèles de production, la rotation des cultures joue un rôle fondamental en contribuant à gérer les ressources de productions, en minimisant l'usage de produits non écologiques, en contrôlant biologiquement la population de parasites et les plantes non souhaitées.

Cette thèse aborde deux problèmes de *Planification de la Rotation des Cultures* (PRC) dans le cadre de la production durable de légumes: le problème de PRC *avec contraintes d'Adjacences* (PRC-A) et le problème de PRC *avec demandes* (PRC-D). La planification des cultures légumineuses est complexe car il implique, en général, un nombre important de spécifications concernant les époques (limitées) de l'année où La semence est possible, les durées (variables selon les espèces) de croissances et les époques (variées également) de moisson. La planification des rotations des cultures sur les terres agricoles est formulée comme un problème en variables binaires où sont considérés les aspects techniques (époques de semence, de moissonages etc.) et écologiques (culture d'engrais vert, jachère, etc.).

Dans le problème PRC-A, l'objectif est de maximiser l'occupation des sols et les contraintes concernent l'adjacence des terres cultivées. Ce problème possède un nombre élevé de variables et de contraintes, avec une structure particulière (bloc-diagonale) permettant d'utiliser la décomposition de Dantzig-Wolfe et, ainsi, une résolution heuristique ou exacte par génération de colonnes.

Dans le problème PRC-D, on souhaite répondre à une demande en légumineux en ayant à notre disposition un ensemble de sols hétérogènes. Les cultures d'engrais dépendent du sol considéré. Nous avons formulé le problème comme un problème de programmation linéaire où chaque variable correspond à une rotation des cultures sur un surface agricole donnée. Ce modèle contient un nombre exponentiel de variables et est résolu par génération de colonnes.

Des expérimentations numériques basées sur des données réelles ont montré l'efficacité des modèles et techniques de résolution proposées.

# Conteúdo

# Capítulo 1

# Introdução

Os modelos convencionais de produção e distribuição agrícola baseiam-se principalmente em monoculturas e no uso intensivo de capital, pesticidas, fertilizantes sintéticos e outros recursos poluentes e não renováveis. Embora o uso cada vez mais intensivo de tais recursos em grandes monoculturas reduza os custos dos alimentos, existem importantes, e usualmente não computados, efeitos negativos nesse modo de produção. Exemplos disso são os custos ambientais e à saúde humana, com a poluição do solo, subsolo e da água, mortalidade de animais silvestres, contaminação dos alimentos, além dos custos sociais com o aumento da exclusão do mercado e do empobrecimento de agricultores familiares, com os crescentes volumes mínimos de produção para a viabilidade econômica (Ehlers, 1996; Gliessman, 2000).

Tais consequências levaram, já há algumas décadas, ao questionamento da sustentabilidade desses modelos de produção agrícola, fazendo surgir diversos modelos alternativos de produção e distribuição, que visam incrementar a sustentabilidade ecológica, econômica e social da agricultura (Gliessman, 2000; Altieri, 2002). De fato, um número crescente de consumidores tem se tornado mais cônscios sobre esses tópicos gerando um valor monetário extra associado a produtos oriundos de sistemas agrícolas mais amigáveis ao meio ambiente e a questões sociais (Makatouni, 2000; Tillman et al., 2002; Seyfang, 2006; Lyon, 2006). Este valor prêmio tem impulsionado ainda mais o crescimento de modelos de planejamento de produção e distribuição agrícolas em que se consideram tanto aspectos ecológicos como o acesso de pequenos produtores agrícolas ao mercado.

Um dos eixos centrais dos modelos de planejamento de produção de base sustentável é a diversidade das atividades e ambientes nas áreas de cultivo. A rotação de culturas é a principal estratégia para incrementar a diversificação da vegetação no tempo em uma mesma área. Estudos agronômicos e ecológicos demonstram que o aumento da diversidade de espécies vegetais melhora a exploração dos recursos produtivos e está associada a um menor ataque de pestes e maior estabilidade da produção frente às pressões ambientais (Gliessman, 2000; Vandermeer, 1989). A

diversificação da produção também resulta em menor vulnerabilidade dos agricultores frentes às oscilações dos preços dos produtos, melhor distribuição das receitas ao longo do ano e a oferta de alimentos variados à população. Assim, critérios que aumentem a eficiência da diversificação na alocação de culturas contribuem para a sustentabilidade da agricultura.

Dentre os critérios ecológicos para a rotação de culturas, podem ser relacionados: (a) não cultivar plantas de mesma família botânica em sequência imediata, pois embora existam organismos-pestes generalistas, muitas pragas e doenças atacam culturas de mesma família; (b) o plantio de culturas, nomeadas de adubação verde (usualmente leguminosas), para a fixação de nitrogênio no solo e (c) períodos de pousio, em que a vegetação espontânea cresce livremente por um período definido, o que contribui para o controle biológico de diversos insetos e para recuperar a estrutura física, química e biológica do solo (Altieri, 2002; Gliessman, 2000).

Nesta tese abordamos dois problemas focados no planejamento de produção de base sustentável de hortaliças: o problema da *Programação de Rotação de Culturas com restrições de Adjacências* (PRC-A) e o problema de *Programação de Rotação de Culturas com atendimento da Demanda* (PRC-D). O planejamento da produção de hortaliças é complexo se comparado à produção de espécies florestais perenes ou culturas anuais como milho e soja. Envolve, em geral, um grande número de culturas de diferentes famílias botânicas, com períodos de cultivo e produtividades com alta variabilidade, além de apresentar limitações específicas quanto à época de plantio. Mesmo onde é possível cultivar-se o ano inteiro (desde que haja irrigação disponível), existem culturas mais apropriadas às estações de primavera e verão e outras às estações de outono e inverno.

Na programação de rotação de culturas para os problemas PRC-A e PRC-D, consideram–se as restrições agronômicas (época de plantio e colheita etc.) e as restrições de base sustentável (a)–(c). No capítulo 2, apresentamos uma formulação matemática para a programação de rotação de culturas e discutimos alguns modelos de rotação encontrados na literatura.

No capítulo 3 caracterizamos o problema PRC-A, cujo objetivo é a maximização da ocupação de áreas de plantio particionada em lotes. Para cada lote, determina-se uma programação de rotação de culturas e as restrições de plantio para culturas de mesma família botânica são estendidas aos lotes adjacentes. Na Seção 3.1,

apresentamos o modelo de otimização 0–1 proposto para o problema e um exemplo ilustrativo. Na Seção 3.2, demonstramos que, para certas configurações especiais das áreas de plantio, uma solução ótima é construída a partir de uma solução particular para dois lotes adjacentes. Para outras configurações espaciais, uma solução factível é obtida com a solução para um número reduzido de lotes adjacentes.

No Capítulo 4, abordamos o problema PRC-D no qual deseja-se atender a demanda de um conjunto de hortaliças, tendo-se disponível um conjunto de áreas heterogêneas, em que as culturas passíveis de plantio têm suas produtividades, dependentes da área considerada e da época de plantio. Para suprir a demanda das culturas, as áreas são divididas em lotes e para cada lote, é obtida uma programação de rotação de culturas, sujeito a restrições (a)-(c). Na Seção 4.1, propomos uma formulação linear para o problema. Na Seção 4.2, discutimos extensões do modelo PRC-D que incorporam certas situações práticas.

No Capítulo 5, apresentamos os procedimentos usados na busca por soluções ótimas ou quase ótimas para os problemas PRC-A e PRC-D. Na Seção 5.1 fazemos uma breve revisão das metodologias empregadas. Na Seção 5.2 propomos procedimentos heurísticos e exatos para o problema PRC-A. Na Seção 5.3, exibimos um algoritmo de geração de colunas que resolve exatamente o problema linear PRC-D.

No capitulo 6, um conjunto de experimentos computacionais baseados em dados reais confirma a eficácia dos modelos e das metodologias propostas para os problemas PRC-A e PRC-D. Todos os algoritmos foram implementados em linguagem C++ com o auxílio da biblioteca ILOG Concert do *software* comercial CPEX 11.1 (ILOG, 2007). Os resultados computacionais para os problemas PRC-A e PRC-D são apresentados, respectivamente, nas Seções 6.1 e 6.2.

# Capítulo 2

# Programação de Rotação de Culturas

A rotação de culturas é um sistema de plantio em que uma sequência de culturas, cultivada em uma dada área em um intervalo de tempo, é sucessivamente repetida na mesma área. O tamanho da rotação é o intervalo de tempo necessário para que se tenha início a repetição da sequência de culturas. Considerando–se a rotação de 3 culturas *milho–trigo–trigo*, por exemplo, indica que, primeiramente, o milho é plantado, sendo seguido de dois cultivos sucessivos de trigo, retornando então ao milho e, assim, sucessivamente.

A escolha apropriada do plano de rotação possibilita o uso contínuo do solo de forma economicamente viável e sem redução da sua fertilidade e, com isto, mantendo a produtividade das culturas envolvidas. Além disto, contribui para o controle biológico de pragas e doenças e para a diminuição da vulnerabilidade da produção agrícola contra riscos econômicos e ambientais. Neste sentido, em muitos modelos matemáticos que tratam de rotações de culturas, são proibidas determinadas sequências (ou rotações) de culturas sabidamente não aconselháveis por algum critério ecológico ou agronômico. Dentre os critérios ecológicos para a rotação de culturas, podem ser relacionados: o cultivo de espécies de diferentes famílias botânicas em sequência e a inserção de culturas para adubação verde e de períodos de pousio.

Uma das principais finalidades do estabelecimento de áreas e de períodos com diferentes espécies é reduzir os recursos disponíveis tanto para artrópodes quanto microorganismos prejudiciais às lavouras, reduzindo assim suas populações e seus danos. Embora existam organismos-peste generalistas, muitos deles obtêm recursos de plantas da mesma família botânica. A adubação verde é o cultivo de culturas, geralmente de uma leguminosa, com a finalidade de melhorar a fertilidade e proteger o solo. Períodos de pousios contribuem para o controle biológico de diversos artrópodes (insetos e ácaros), na recuperação da estrutura física, química e biológica do solo, além de contribuírem para a reciclagem de nutrientes (Altieri, 2002; Gliessman, 2000).

Na seção seguinte, são discutidos alguns modelos de rotação de culturas encontrados na literatura. Na Seção 2.2 propomos um modelo de otimização 0–1 para determinar uma programação de rotação de culturas (PRC) que considera tantos aspectos técnicos quanto ecológicos.

## 2.1 Alguns modelos matemáticos de rotação de culturas

O problema de determinar a rotação de cultura para uma dada área de plantio não é novidade na Pesquisa Operacional. Em 1939, Kantorovich o relaciona como uns dos problemas que podem ser tratados por meio de ferramentas matemáticas (Kantorovich, 1960 – traduzido do original russo, datado de 1939). Hildreth e Reiter (1951), no primeiro congresso em aplicações de programação linear (Chicago, 1949), apresentam um modelo linear cujo objetivo é particionamento ótimo de uma área de plantio de modo que cada parcela da terra seja ocupada por uma rotação de culturas definida previamente. Eles argumentam que esta estratégia amortiza riscos econômicos e a necessidade de água, mão–de–obra, dentre outros. As variáveis de decisão são o tamanho das áreas com cada rotação. Os autores exemplificam com a construção de 4 das 8 possíveis rotações de 3 anos com as culturas milho (M) e feno (F). No caso, as rotações M-M-M, F-F-F, M-M-F e M-F-F.

El-Nazer e McCarl (1986), Haneveld e Stegeman (2005) e Detlefsen e Jensen (2007) eliminam a necessidade de rotações pré-definidas resolvendo o problema de rotação de culturas com sequências de culturas cultivadas uma após a outra. Para isto, assume–se que as áreas de plantio são homogêneas e existe única seção de plantio anual. As variáveis de decisão referem–se ao tamanho das áreas de plantio com as sequências. Tal definição permite uma grande variedade de rotações sem a necessidade de explicitá-las mas não se aplica a situações em que se têm várias seções de plantio anual ou a heterogeneidade espacial precisa ser considerada. (Dogliotti et al., 2003).

No modelo proposto por El-Nazer e McCarl (1986), para uma dada área de plantio, são definidas variáveis $X_{ijkr}$ associadas ao tamanho da área plantada com a cultura $i$ após o plantio sequencial das culturas $j$, $k$ e $r$ nos três anos anteriores. Sequências indesejáveis de culturas (por algum critério agronômico ou ecológico, por exemplo) podem ser eliminadas a *priori*. Os autores supõem que a produtividade de

uma cultura depende das culturas plantadas na mesma área nos 4 anos anteriores e a função objetivo é o retorno financeiro obtido com o plantio das sequências escolhidas. As restrições do modelo garantem a conservação de fluxo em termos de área de plantio, ou seja, a soma das áreas com o cultivo das sequências de culturas não ultrapassa o tamanho da área total disponível e a soma das áreas plantadas com as culturas que seguem a sequência ($jkr$) não ultrapassa a área com a sequência ($jkr$) no ano precedente.

Os autores ilustram a formulação para 2 culturas (C e P). As 16 variáveis são todas as possíveis sequências de 4 anos com essas culturas. Se, em particular, a solução é $X_{PCPC} = X_{CPCP} = 0.5$ (50% da área disponível para plantio é ocupada com a cultura P após a sequência CPC e os outros 50%, com a cultura C após a sequência PCP), isto implica que a área é ocupada com a rotação de dois anos C-P. O modelo é implementado em uma fazenda no nordeste do estado de Oregon–EUA com as culturas trigo, milho, batata e alfafa. Um modelo de regressão linear foi usado para estimar o retorno financeiro obtido com as 256 sequências necessárias para o modelo a partir de um conjunto de dados experimentais de rotações de 5 anos com estas culturas.

Em Haneveld e Stegeman (2005), para gerar as rotações, sequências de culturas factíveis de tamanho mínimo são construídas a partir de restrições sobre os plantios. Em um exemplo com as culturas algodão, sorgo e soja e um conjunto de 6 restrições de plantio, são geradas 20 sequências factíveis de 4 anos. As restrições de plantio são: após o plantio de algodão e do sorgo, deve-se ter, pelo menos, 1 ano de pousio; dois anos seguidos de plantio de algodão só deve ocorrer depois de 2 anos de pousio; sorgo pode suceder sorgo também somente após 2 anos de pousio; o plantio de soja só é sucedido pelo plantio de sorgo ou 1 ano de pousio. Por fim, em cada 5 anos, 3 devem ser de pousio. O modelo permite o uso de métodos de solução baseado em algoritmos de fluxo máximo.

Detlefsen e Jensen (2007) levam em conta que a quantidade de nitrogênio que deve ser adicionado ao solo para uma dada cultura é função das culturas ali plantadas nos anos anteriores. Os autores consideram que a quantidade de terra plantada com cada cultura em cada ano já é conhecida, o que permite a formulação do problema de rotação de culturas como um problema de transporte.

Diferentemente dos artigos citados anteriormente, no modelo linear em Clarke (1989), as culturas podem apresentar variadas épocas de plantio e diferentes intervalos de cultivo. No entanto, não existem restrições de plantio para as culturas. Castellazzi et al. (2008) apresentam um modelo matemático para determinar rotações de culturas usando matrizes de transição que podem representar processos estocásticos, facilitando incorporar incertezas nas rotações e previsões de longo prazo. Também, neste caso, a área de plantio é homogênea e a seção de plantio, anual.

Outros artigos de rotação de culturas apresentam ferramentas de apoio à tomada de decisão para avaliar o efeito de diversas rotações (Dogliotti et al., 2003, 2004; Jones et al., 2003; Stöckle et al., 2003, Bachinger e Zander, 2006). Nesses trabalhos, a implementação computacional usualmente atua como ferramenta de auxílio na escolha do plano de rotação e avaliando o efeito na produtividade das culturas e no solo de uma rotação. Em particular, o *software* ROTAT (Dogliotti et al., 2003, 2004) combina culturas de uma lista pré-definida para gerar todas as possíveis rotações, satisfazendo um dado conjunto de regras baseadas em critérios agronômicos e ecológicos, tais como, época de plantio, sequências indesejáveis de culturas, inserção de adubação verde e de pousio e frequência de culturas em uma rotação.

A rotação de culturas também aparece como uma das atividades em outros modelos de planejamento agrícolas mais generalistas ou envolvendo outras questões e objetivos. Annetts e Audsley (2002), por exemplo, apresentam um modelo de programação linear multi–objetivo que considera várias atividades agrícolas, dentre elas, a rotação de culturas, otimizando lucratividade e questões ambientais. Lucas e Chhajed (2004) fazem uma revisão de aplicações de pesquisa operacional a problemas de localização na agricultura, como transporte, distribuição e localização de facilidades. Ahumada (2008) apresenta uma revisão de modelos de cadeia de suprimentos na agricultura, o que inclui modelos de planejamento agrícola. Nesses artigos outros modelos de rotação de culturas inseridos ou não em um conjunto atividades agrícolas são discutidos.

## 2.2 Um modelo para a Programação de rotação de culturas

Considere uma rotação de culturas de $T$ anos e uma unidade de tempo que divide $T$ em $M$ períodos. Definimos *como programação de rotação de culturas* (ou simplesmente *programação de rotação*) de tamanho $M$ como um calendário de plantio e cultivo, na base de tempo adotada (dia, semana etc.), de cada cultura na rotação.

A programação de rotação implica em escolher que culturas plantar e em que período da rotação. Entre possíveis critérios na escolha do plano de rotação foram considerados os seguintes aspectos técnicos e restrições de base sustentável:

(a) Cada cultura tem um período mais cedo e mais tarde de plantio que deve ser respeitado.
(b) As culturas podem ter diferentes ciclos de cultivo (intervalo de tempo do plantio à colheita).
(c) Culturas de mesma família botânica não são plantadas em sequência.
(d) A inserção de culturas para adubação verde e de pousios de tamanho pré–determinado são exigidos na programação de cada rotação.

Como exemplo, considere uma programação de rotação de culturas de 1 ano, divididos em 12 períodos de 1 mês ($M = 12$) e 3 culturas, X, Y e Z com ciclos de cultivo de 5, 4 e 2, respectivamente. As culturas X e Y pertencem à família botânica 1 e Z, para adubação verde, pertence à família botânica 2. A cultura X pode ser plantada entre janeiro (período 1) e julho (período 7), enquanto as culturas Y e Z podem ser plantadas o ano inteiro. Um pousio (P) de 1 mês é exigido em cada rotação. A Figura 2.1 apresenta uma programação factível para este exemplo.

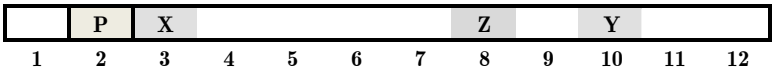



Figura 2.1. Uma programação de rotação de culturas

Na programação exibida na Figura 2.1, a cultura X é plantada no período 3 e ocupa a terra até o período 7 (já que seu ciclo de cultivo é de 5 meses). A cultura Z é cultivada nos períodos 8 e 9. A cultura Y é plantada no período 10 e sua colheita final ocorre no período 1 (do ano seguinte). No período 2 a terra fica em pousio (P). Esta programação é factível, pois os períodos de plantio das culturas são respeitados, as culturas X e Y não são cultivadas em sequência (pois são de mesma família) e um

pousio de um mês bem como o plantio da cultura para adubação verde foram incluídos na rotação.

Para descrever o conjunto de programações de rotações de culturas que respeitam as restrições (a)–(d), considere os seguintes parâmetros e variáveis:

*Parâmetros:*

$M$ número de períodos (em uma unidade de tempo pré–definida) de uma rotação;
$C$ conjunto de culturas que podem ser selecionadas para o plantio, excluindo as culturas para adubação verde;
$G$ conjunto de culturas que podem ser alocadas para adubação verde;
$N$ cardinalidade de $C \cup G$;
$N$+1 cultura fictícia que representa o pousio obrigatório de ciclo pré–definido.
$NF$ número de famílias botânicas;
$F(p)$ conjunto de culturas da família botânica $p$, $p = 1..NF$;
$t_i$ ciclo de cultivo da cultura $i$, incluindo os períodos estimados para preparo do solo e colheita;
$I_i$ conjunto de períodos em que a cultura $i$ pode ser plantada. Para o pousio, adotamos $I_{N+1} = \{1, ..., M\}$.

*Variáveis*:

$x_{ij} = 1$, se a cultura $i$ é plantada no período $j$ e 0, caso contrário, para $i = 1..N+1$ e $j \in I_i$.

A formulação matemática que caracteriza o conjunto de programações de rotações de culturas factíveis para uma dada área pode ser escrito como:

$$\sum_{i=1}^{N+1} \sum_{r=0}^{t_i-1} x_{i,j-r} \leq 1,\ j = 1..M \tag{2.1}$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i} x_{i,j-r} \leq 1,\ p = 1..NF,\ j = 1..M \tag{2.2}$$

$$\sum_{i \in G} \sum_{j \in I_i} x_{ij} = 1 \tag{2.3}$$

$$\sum_{j=1}^{M} x_{N+1,j} = 1 \tag{2.4}$$

$$x_{ij} \in \{0,1\},\ i = 1..N+1,\ j \in I_i \tag{2.5}$$

Obs.: nas equações (2.1)–(2.3), $j - r \leq 0$ é substituído por $j - r + M$.

As restrições (2.1) asseguram que, no máximo, uma cultura ocupe a terra em cada período (incluindo o pousio), enquanto as restrições (2.2) proíbem que culturas de mesma família botânica sejam cultivadas uma imediatamente após a outra.

As restrições (2.3) e (2.4) forçam o plantio de uma cultura para adubação verde e um pousio de tamanho pré–definido em cada programação. Dependendo do tamanho da programação pode ser desejável alterar tais valores. Neste caso, as restrições (2.3) e (2.4) são substituídas por:

$$\sum_{i \in G} \sum_{j \in I_i} x_{ij} = nav \tag{2.6}$$

$$\sum_{j=1}^{M} x_{N+1,j} = np \tag{2.7}$$

em que $nav$ é o número de culturas em G que devem ser plantadas por rotação e $np$ o número de pousios. Um espaço mínimo de tempo entre plantios de duas adubações verdes e pousios é assegurado pelas restrições:

$$\sum_{i \in G} \sum_{r=0}^{tav} x_{i,j-r} \leq 1,\ j = 1..M \tag{2.8}$$

$$\sum_{r=0}^{tp} x_{N+1,j-r} \leq 1,\ j = 1..M \tag{2.9}$$

em que $tav$ e $tp$ são, respectivamente, o intervalo de tempo mínimo entre duas adubações verdes e dois pousios (de tamanho pré–definido).

Naturalmente, outras restrições de interesse podem ser incluídas no modelo. Por exemplo, as programações de plantio envolvem rotação de culturas. Facilmente, o modelo (2.1)–(2.5) poderia referir–se a sequência de culturas sem a exigência de que esta seja repetida sucessivamente,.para tornar–se uma rotação. Para isto, é suficiente impor $x_{ij} = 0$ para $j - r \leq 0$.

A formulação (2.1)–(2.5) permite soluções em que um dado período não tenha nenhuma cultura associada. Isto implica em mais períodos em que a terra fica em pousio do que aqueles exigidos em para cada rotação.

Nos dois problemas de planejamento de produção agrícola abordados nesta tese e detalhados nos capítulos seguintes, a programação da rotação de culturas desempenha um papel fundamental. Em ambos, a formulação (2.1)–(2.5) foi usada para determinar as programações de rotação de culturas para as áreas de plantio.

# Capítulo 3

# Programação de Rotação de Culturas com restrições de Adjacências (PRC-A)

Um dos eixos centrais dos modelos alternativos de produção agrícola é a diversificação das atividades e ambientes. Como mencionado anteriormente, a principal estratégia para aumentar a diversificação da vegetação no tempo em uma mesma área é a rotação de culturas. A diversificação de culturas no espaço, considerando o mesmo período de tempo, pode ser obtida com a divisão da área de cultivo em vários lotes e a definição de uma rotação de culturas para cada um deles (Altieri, 2002; Gliessman, 2000).

A produção de hortaliças se adéqua bem à implantação de sistemas mais sustentáveis de produção pois os produtores de hortaliças estão acostumados a lidar com a produção continuada de um grande número de culturas alocadas simultaneamente em áreas adjacentes. Contudo, essa forma de diversificação nem sempre segue um planejamento que otimize as vantagens desta forma de cultivo. No Brasil, hortas comunitárias têm crescido bastante nos últimos anos. Diferente de hortas comerciais, a produtividade ou lucratividade não é tão importante, sendo valorizada uma produção contínua de alimentos saudáveis, com pouco ou nenhum uso de agrotóxico, produzido pela própria família ou pela comunidade (Silva et al., 1999; Makishima, 1993).

Neste capítulo, propomos um modelo de programação de rotação de cultura com restrições de adjacências (PRC-A), cujo objetivo é a maximização da ocupação de áreas de plantio divididas em lotes, sujeito a restrições de base ecológica tanto para as rotações de culturas quanto para os lotes vizinhos. Na Seção 3.1, apresentamos uma formulação para o problema e um exemplo ilustrativo. Na Seção 3.2 demonstramos que, sob certas condições, uma solução factível ou mesmo ótima para qualquer área de plantio é construída a partir de soluções com um número reduzido de lotes.

## 3.1 Um modelo para o problema PRC-A

Uma área de plantio particionada em lotes pode ser representada por um grafo conexo planar G(V, A), em que o conjunto de vértices V corresponde aos lotes e $(u, v) \in$ A se, e somente se, os lotes $u$ e $v$ são adjacentes. Consideramos que dois lotes são adjacentes se eles dividem uma fronteira que não é um conjunto discreto de pontos. A Figura 3.1(a) ilustra uma área de plantio particionada em 5 lotes e a Figura 3.1(b) o grafo associado a esta partição.

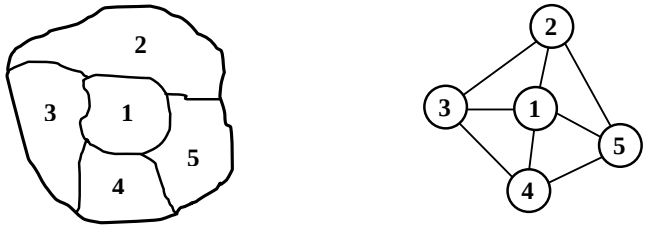



Figura 3.1. Uma particular área de plantio com 5 lotes.

Dado uma área de plantio particionada em lotes e um conjunto de culturas que podem ser selecionadas para as rotações nos lotes, o problema de encontrar a programação de rotações de culturas para este conjunto de lotes pode ser dado sob diferentes suposições, restrições e objetivos. O problema PRC-A é definido como o de determinar uma programação de rotação de culturas de mesma duração para cada um dos lotes, maximizando a ocupação da área de plantio. A rotação de culturas para cada lote está sujeita às restrições de base ecológica discutidas no Capítulo 2: (a) *proibição de plantio sequencial,* (b) *adubação verde* e (c) *pousio.*

Além destas restrições, para dificultar a propagação de pragas e doenças consideramos a configuração espacial dos lotes de modo que também a seguinte restrição deve ser atendida:

(d) *Proibição de plantio espacial:* culturas de mesma família não são cultivadas em uma mesma época em lotes adjacentes.

Esta restrição pode ser vista como uma equivalente "espacial" da *proibição de plantio sequencial* (a). Naturalmente, ela também se aplica a culturas para adubação verde.

*Exemplo Ilustrativo*

Seja uma área de plantio particionada em 5 lotes com configuração espacial como ilustrada na Figura 3.1 e um conjunto de 28 culturas em 10 famílias botânicas. Os dados para as culturas estão na Tabela A.1 no Apêndice A. As culturas 25 a 28 são para adubação verde. Considere o tamanho da rotação de 2 anos dividido em períodos de 10 dias e um pousio obrigatório de 3 períodos (1 mês) em cada rotação. O diagrama na Figura 3.2 exibe uma programação ótima para os 5 lotes nos 24 meses. O pousio está representado pela cultura fictícia 29.

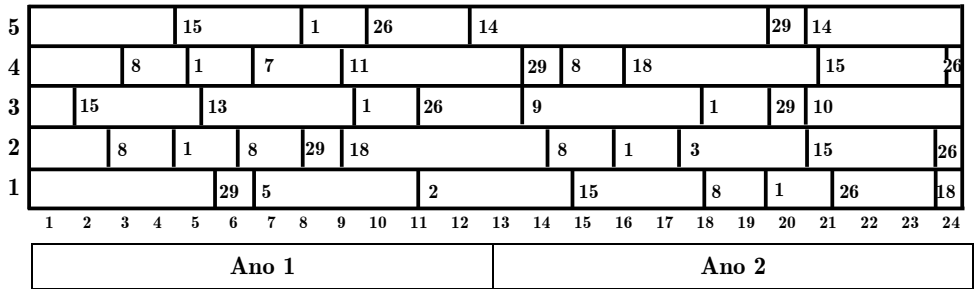



Figura 3.2. Programação ótima de 2 anos para os 5 lotes.

Como pode ser visto na Figura 3.2, as restrições (a)–(d) foram atendidas: para cada lote, a programação de rotação de culturas satisfaz as restrições (a)–(c) e lotes vizinhos não cultivam ao mesmo tempo culturas de mesma família botânica. Como a restrição (d) é somente imposta a lotes adjacentes, para aqueles que não o são, existem culturas de mesma família botânica cujo cultivo coincide em alguns períodos. Na programação para os lotes 2 e 4, por exemplo, a cultura 26, para adubação verde, tem parte do seu cultivo em períodos iguais em ambos os lotes.

### *Formulação Matemática*

Considere uma área de plantio particionada em lotes e representada por um grafo G(V, A) e um conjunto de $N$ culturas distribuídas em $NF$ famílias botânicas passíveis de plantio nesta área. Para apresentar a formulação matemática 0-1 para o problema PRC–A considere os parâmetros e variáveis dados a seguir. Por facilidade de leitura, além dos parâmetros associados à configuração espacial dos lotes, listamos outros já definidos na Seção 2.1.

*Parâmetros:*

$L$ número de lotes em que a área é particionada;

$M$ número de períodos em uma programação;

$C$ conjunto de culturas que podem ser selecionadas para o plantio, com exceção de culturas para adubação verde;

$G$ conjunto de culturas para adubação verde;

$N$ cardinalidade de $C \cup G$;

$F(p)$ conjunto de culturas da família botânica $p$, $p = 1..NF$;

$t_i$ ciclo de cultivo da cultura $i$, incluindo os períodos para o preparo do solo e colheita;

$I_i$ conjunto de períodos em que a cultura $i$ pode ser plantada.

$N$+1 cultura fictícia representando o pousio obrigatório de tamanho pré–definido. Com $I_{N+1} = \{1, ..., M\}$.

*Variáveis de decisão:*

$x_{ijk} = 1$, se a cultura $i$ é plantada durante o período $j$ no lote $k$ e 0, caso contrário, para $i = 1..N+1$, $j \in I_i$, $k = 1..L$.

*Formulação matemática*:

$$(PRC\text{-}A) \qquad z_{PRCA} = \max \sum_{k=1}^{L} \sum_{i \in C} \sum_{j \in I_i} t_i x_{ijk} \tag{3.1}$$

*Sujeito a*:

$$\sum_{i=1}^{N+1} \sum_{r=0}^{t_i - 1} x_{i,j-r,k} \leq 1, \; j = 1..M, k = 1..L \tag{3.2}$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i} x_{i,j-r,k} \leq 1, \; p = 1..NF, j = 1..M, k = 1..L \tag{3.3}$$

$$\sum_{j=1}^{M} \sum_{i \in G} x_{ijk} = 1, \; k = 1..L \tag{3.4}$$

$$\sum_{j=1}^{M} x_{N+1,j,k} = 1, \; k = 1..L \tag{3.5}$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i,j-r,u} + x_{i,j-r,v} \right] \leq 1, p = 1..NF, j = 1..M, (u,v) \in A, \tag{3.6}$$

$$x_{ijk} \in \{0,1\}, i = 1..N+1, j \in I_i, k = 1..L \tag{3.7}$$

Obs.: nas restrições (3.2), (3.3) e (3.6), $j - r \leq 0$ é substituído por $j - r + M$.

A função (3.1) fornece o tempo total de cultivo a ser maximizado. As variáveis são ponderadas pelo ciclo de cultivo da cultura para permitir que tenham possibilidade de serem escolhidas, sem influência significativa do tamanho do ciclo da cultura a que se referem. As variáveis relativas às culturas para a adubação verde foram excluídas da função objetivo, pois se exige apenas que, em cada rotação, uma delas seja cultivada uma única vez. O mesmo se aplica ao pousio.

As restrições (3.2)–(3.5) são equivalentes às restrições (2.1)-(2.4) propostas no Capítulo 2. As restrições (3.2) impedem que duas culturas ocupem simultaneamente o mesmo espaço, as restrições (3.3) proíbem que culturas de mesma família possam ser cultivadas consecutivamente no mesmo período, dentro de um mesmo lote e as restrições (3.4) e (3.5), referem-se ao plantio de uma cultura para adubação verde e um pousio na programação de cada lote.

As restrições (3.6) proíbem que culturas de mesma família botânica sejam cultivadas ao mesmo tempo em lotes adjacentes. Esse conjunto de restrições pode ser substituído pelas restrições:

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ m_k x_{i,j-r,k} + \sum_{(k,u) \in A} x_{i,j-r,u} \right] \leq m_k, p = 1..NF, j = 1..M, k = 1..L \tag{3.8}$$

em que a constante $m_k$ é o valor ótimo do seguinte problema $Q_k$, $k = 1..L$.

$$(Q_k) \qquad \begin{aligned} m_k = \max & \sum_{(k,u) \in A} y_u \\ \text{s.a} & \\ & y_u + y_v \leq 1, (u,v) \in A \\ & y_k = 0, y_u, y_v \in \{0,1\} \end{aligned}$$

Para o grafo representado na Figura 2, por exemplo, $m_k = 2$, $k = 1..5$.

O problema $Q_k$ é o problema clássico de encontrar um conjunto maximal independente de vértices, ou equivalentemente, um clique maximal no grafo complementar (Diestel, 2000).

De fato, as restrições (3.2) e (3.6) têm o mesmo conjunto solução das restrições (3.2) e (3.8). Como as variáveis $x_{ijk}$ são binárias e, por (3.2), $\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} x_{i(j-r)v} \leq 1$, podemos definir a seguinte variável binária $y_{pjv}$ para cada família botânica $p$, período $j$ e lote $v$ :

$$y_{pjv} = \sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} x_{i(j-r)v} \text{ , } p = 1..NF, \, j = 1..M, \, v \in V$$

Desta forma, reescrevemos as restrições (3.6) e (3.8) para $p = 1..NF$ e $j = 1..M$, respectivamente, como:

$$y_{pju} + y_{pjv} \leq 1, (u, v) \in A \qquad \text{e} \qquad m_k y_{pjk} + \sum_{(k,u) \in A} y_{pju} \leq m_k \text{ , } k = 1..L$$

Por definição do problema $Q_k$, segue a equivalência das restrições.

Se, no grafo G(V, A), o número de arestas for maior que o número de vértices (o que usualmente acontece em grafos conexos), a substituição das restrições (3.6) por (3.8) reduz (em alguns casos, significativamente) o número de restrições do modelo. Considere $L = |\text{V}|$ e $R = |\text{A}|$, respectivamente, o número de vértices e de arestas em G. A diferença entre o número de arestas e vértices ($R$–$L$) do grafo G corresponde a uma redução de ($R$–$L$)x$M$x$NF$ restrições no modelo. Em particular, para o exemplo da Figura 3.2 esta redução foi de 2160 ($R$–$L = 3$, $M = 72$ e $NF = 10$) restrições.

Além da possível redução do número de restrições, para muitas combinações de configuração espacial e conjunto de culturas, obtêm–se restrições "mais fortes" com a substituição das restrições (3.6) pelas restrições (3.8). Seja, por exemplo, o grafo completo $K_3$ (3 vértices dois a dois adjacentes). Neste caso, $m_k = 1$, $k = 1..3$. Assim, o conjunto de restrições (3.8) é:

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i(j-r)1} + x_{i(j-r)2} + x_{i(j-r)3} \right] \leq 1, p = 1..NF, \, j = 1..M \text{ , para } k = 1..3$$

e o conjunto de restrições (3.6) é:

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i(j-r)1} + x_{i(j-r)2} \right] \leq 1, p = 1..NF, \, j = 1..M$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i(j-r)1} + x_{i(j-r)3} \right] \leq 1, p = 1..NF, j = 1..M$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i(j-r)2} + x_{i(j-r)3} \right] \leq 1, p = 1..NF, j = 1..M$$

Testes computacionais com configurações espaciais geradas aleatoriamente mostram que, em média, o valor ótimo da relaxação linear do modelo PRC–A com a restrição (3.8) é menor que o obtido com as restrições (3.6). No entanto, um conjunto de restrições não necessariamente domina o outro. Considere $M = 3$ e 2 culturas em famílias botânicas distintas, ambas com ciclo de cultivo 1, sendo a segunda cultura para adubação verde. Para o grafo $K_3$, o valor da solução ótima do problema PRC–A é $z^* = 1$ (ou seja, só em um dos lotes a cultura 1 é alocada), a relaxação linear do modelo com as restrições (3.8) é também $z_{R(3.8)} = 1$ e, com as restrições (3.6), a relaxação linear é $z_{R(3.6)} = 1{,}5$. Por outro lado, para a área de plantio com 6 lotes em paralelo e com estes mesmo dados para as culturas, $z^* = z_{R(3.6)} = 3$ e $z_{R(3.8)} = 3{,}33$.

Outra característica do problema PRC–A é de ter, em geral, múltiplas soluções degeneradas. A degeneração ocorre porque a matriz de restrições do modelo é muito esparsa e, como consequência da definição das variáveis, poucas podem assumir valor não–nulo. As múltiplas soluções aparecem, dentre outras possibilidades, quando é possível "rotacionar" a programação para os $L$ lotes para "frente" ou para "atrás" no tempo. Considere uma solução ótima $(\hat{x}_{ijk})$, $i = 1..n$, $j \in I_i$, $k = 1..L$. Se $\forall \hat{x}_{ijk} = 1$, $j + r \in I_i$, com $r \in \mathbb{N}$, então a solução $(\hat{x}_{i,j+r,k})$ é também ótima para o PRC–A. Analogamente, uma solução $(\hat{x}_{i,j-s,k})$, com $s \in \mathbb{N}$ é construída. Se existe $r$ ou $s$ não nulos, então existe solução ótima alternativa. O número dessas soluções ótimas alternativas depende do valor máximo que $r$ e $s$ podem assumir. Outras soluções ótimas também podem ser construídas para aqueles grafos em que é possível atribuir uma programação de um lote a um conjunto de lotes (como discutido na próxima seção) ou atribuir a programação de um lote a outro lote e vice–versa. Por exemplo, dada uma solução ótima para o grafo $K_3$ também é ótima a solução obtida atribuindo–se a programação do lote 1 ao lote 2, a programação do lote 2 ao lote 3 e a do lote 3 ao lote 1.

## 3.2 Explorando a configuração espacial da área de plantio

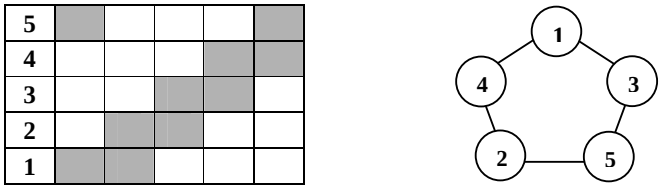
Considere G(V, A) o grafo da área de plantio. Como G é planar, pelo *Teorema das Quatro Cores*, é sempre possível particionar V em, no máximo, 4 subconjuntos independentes. Em um conjunto independente, não existem vértices adjacentes e, no contexto de coloração de grafos, todos podem receber a mesma cor (Diestel, 2000). No contexto de programação de rotação de culturas, a vértices de mesma cor é possível atribuir a mesma programação. O grafo ilustrado na Figura 3.1, por exemplo, pode ser particionado em 3 conjuntos independentes, {1}, {2, 4} e {3, 5} e, portanto, colorido com 3 cores diferentes.

Considere agora a programação para os lotes 1, 2 e 3, exibidas na Figura 3.2. Podemos atribuir ao lote 4 a programação do lote 2 e ao lote 5 a programação do lote 3 porque as restrições de vizinhança não são violadas: os lotes 2 e 4 não são adjacentes assim como não o são os lotes 3 e 5. Além disto, as restrições de vizinhança para os lotes adjacentes aos lotes 4 e 5 estão satisfeitas pois as programações para os lotes 2 e 3 já garantiram isto. Portanto, com apenas 3 programações podemos construir uma programação factível para os 5 lotes.

De maneira geral, se existe uma programação factível para um grafo completo $K_n$, isto é um grafo com $n$ vértices dois a dois adjacentes, então uma programação factível para um grafo G com, no máximo, $n$ subconjuntos independentes pode ser construída a partir daquela solução. Como quaisquer dois vértices de um subconjunto independente de vértices não são adjacentes, a mesma programação pode ser designada a todos os lotes de um mesmo subconjunto independente sem violar as restrições de adjacências (3.6). Assim, obtemos uma solução factível para G designando cada uma das programações dos vértices de $K_n$ a todos os lotes de um único conjunto independente de V, de modo que nenhuma programação seja atribuída a mais que um dos subconjuntos independentes de V. O resultado deste procedimento pode ser resumido na proposição 1 dada a seguir.

**Proposição 1**: *Se existe uma programação factível para o grafo completo $K_n$, então também existe uma programação factível para qualquer grafo G(V, A) com número de conjuntos independentes menor ou igual a n.*

No entanto, pode não existir uma programação factível para $K_n$ e, assim mesmo, existir uma programação para um grafo G particionado em $n$ subconjuntos independentes de vértices. Por exemplo, suponha $M = 5$, pousio obrigatório de 1 período e uma cultura X para adubação verde, sem restrições de época de plantio, e com ciclo de cultivo de 2 períodos. Então, como ilustrado na Figura 3.3(a), só existem 5 possíveis alocações distintas para X em uma rotação. Além disto, é fácil verificar que não existe programação factível para $K_3$. Seja agora o grafo exibido na Figura 3.3(b) que é 3-cromático, embora não tenha clique com 3 vértices. É possível construir uma programação factível para este grafo usando as alocações 1 a 5 de X, respectivamente, nos lotes 1 a 5 e com o pousio sucedendo o plantio da cultura X.




(a) (b)

Figura 3.3: (a) Cinco alocações no tempo para a cultura X e (b) Grafo 5–ciclo

### 3.2.1 Lotes em paralelo ou no formato tabuleiro

Uma situação comumente encontrada na prática é o particionamento da área de plantio em lotes paralelos ou no formato tabuleiro. Nestes casos, uma condição necessária e suficiente para existir uma solução factível é existir solução para dois lotes adjacentes. Como exemplo, considere uma área de plantio no formato de tabuleiro, como ilustrado na Figura 3.4, e uma programação para dois lotes adjacentes ($K_2$). Uma programação para os 9 lotes é obtida designando–se a programação de um dos lotes de $K_2$ aos lotes 1, 3, 5, 7 e 9 e a outra aos lotes 2, 4, 6 e 8. Por outro lado, se não existe uma programação para $K_2$, não pode existir uma programação para os 9 lotes.


(a) (b)

Figura 3.4. Área de plantio no formato tabuleiro com 9 lotes

Além disto, é possível construir uma solução ótima para esta área de plantio a partir de uma solução particular para $K_2$. As proposições 2 e 3, dadas a seguir, provam esta afirmação.

**Proposição 2**: *Uma solução ótima para o modelo de programação de rotação de culturas com restrições de adjacências (PRC–A) para L lotes paralelos ($L > 1$) pode ser construída a partir de uma solução ótima para 2 lotes adjacentes.*

**Prova**: por simplicidade, a formulação matemática descrita na Seção 3.1 para $L$ lotes é reescrita na forma matricial a seguir.

$$(PRC\text{-}A) \qquad \begin{aligned} \max \quad & T\mathrm{x}_1 + \cdots + T\mathrm{x}_L \\ s.a. \quad & P\mathrm{x}_u + P\mathrm{x}_v \leq \mathbf{1}, (u,v) \in A \\ & \mathrm{x}_u \in S, u \in V \end{aligned}$$

em que $S$ é o conjunto de todas as programações factíveis para um lote e $P$ a matriz associada às restrições de adjacências (3.6). O vetor $T$ indica o retorno associado a uma programação de S, ou seja, $T_{ij} = t_i$ para $i \in C$ e $T_{ij} = 0$, caso contrário.

Seja $P$ o problema $PRC$–$A$ para esta particular configuração espacial, ou seja,

$$(P) \qquad \begin{aligned} \max \quad & T\mathrm{x}_1 + \cdots + T\mathrm{x}_L \\ s.a. \quad & P\mathrm{x}_k + P\mathrm{x}_{k+1} \leq \mathbf{1}, k = 1..L-1 \\ & \mathrm{x}_k \in S, k = 1..L \end{aligned}$$

e considere os problemas $PP$ e $PI$ como definidos abaixo:

$$(PP) \qquad \begin{aligned} \max \quad & T\mathrm{x}_1 + T\mathrm{x}_2 \\ s.a. \quad & P\mathrm{x}_1 + P\mathrm{x}_2 \leq \mathbf{1} \\ & \mathrm{x}_1, \mathrm{x}_2 \in S \end{aligned} \qquad\qquad (PI) \qquad \begin{aligned} \max \quad & \frac{L+1}{2} T\mathrm{x}_1 + \frac{L-1}{2} T\mathrm{x}_2 \\ s.a. \quad & P\mathrm{x}_1 + P\mathrm{x}_2 \leq \mathbf{1} \\ & \mathrm{x}_1, \mathrm{x}_2 \in S \end{aligned}$$

Seja $L$ par e $(\tilde{\mathrm{x}}_1, \tilde{\mathrm{x}}_2)$ uma solução ótima de $PP$. A solução $(\tilde{\mathrm{x}}_1, \cdots, \tilde{\mathrm{x}}_L)$, em que $\tilde{\mathrm{x}}_k = \tilde{\mathrm{x}}_1$, para $k$ ímpar e $\tilde{\mathrm{x}}_k = \tilde{\mathrm{x}}_2$, para $k$ par é factível para o problema $P$ pois

$$P\tilde{\mathrm{x}}_k + P\tilde{\mathrm{x}}_{k+1} = P\tilde{\mathrm{x}}_1 + P\tilde{\mathrm{x}}_2 \leq \mathbf{1}, k = 1..L-1 .$$

Para esta solução,

$$T\tilde{\mathrm{x}}_1 + \cdots + T\tilde{\mathrm{x}}_L = \frac{L}{2}\left[T\tilde{\mathrm{x}}_1 + T\tilde{\mathrm{x}}_2\right] \tag{3.9}$$

Seja $(\mathrm{x}_1,\cdots,\mathrm{x}_L)$ factível para o problema *PRC-A*. Como $P\mathrm{x}_k + P\mathrm{x}_{k+1} \le \mathbf{1}$, então $(\mathrm{x}_k,\mathrm{x}_{k+1})$, $k = 1..L-1$ são factíveis para *PP*. Como $(\tilde{\mathrm{x}}_1,\tilde{\mathrm{x}}_2)$ é uma solução ótima de *PP*,

$$T\mathrm{x}_{2k-1} + T\mathrm{x}_{2k} \le T\tilde{\mathrm{x}}_1 + T\tilde{\mathrm{x}}_2\,,\ k = 1..\frac{L}{2} \tag{3.10}$$

Somando as $L/2$ equações (3.10) e usando (3.9) obtém-se

$$\sum_{k=1}^{L} T\mathrm{x}_k \le \frac{L}{2}\left[T\tilde{\mathrm{x}}_1 + T\tilde{\mathrm{x}}_2\right] \overset{(3.9)}{=} \sum_{k=1}^{L} T\tilde{\mathrm{x}}_k$$

Portanto, para $L$ par $(\tilde{\mathrm{x}}_1,\cdots,\tilde{\mathrm{x}}_L)$ é solução ótima para o problema *PRC-A*.

Seja $L$ ímpar e $(\hat{\mathrm{x}}_1,\hat{\mathrm{x}}_2)$ uma solução ótima de *PI*. A solução $(\hat{\mathrm{x}}_1,\cdots,\hat{\mathrm{x}}_L)$, em que $\hat{\mathrm{x}}_k = \hat{\mathrm{x}}_1$ para $k$ ímpar e $\hat{\mathrm{x}}_k = \hat{\mathrm{x}}_2$ para $k$ par é factível para o problema $P$ pois

$$P\hat{\mathrm{x}}_k + P\hat{\mathrm{x}}_{k+1} = P\hat{\mathrm{x}}_1 + P\hat{\mathrm{x}}_2 \le \mathbf{1},\ k = 1..L-1\,.$$

Para esta solução, como existem $\frac{L+1}{2}$ termos ímpares e $\frac{L-1}{2}$ termos pares,

$$T\hat{\mathrm{x}}_1 + \cdots + T\hat{\mathrm{x}}_L \;=\; \frac{L+1}{2}T\hat{\mathrm{x}}_1 + \frac{L-1}{2}T\hat{\mathrm{x}}_2 \tag{3.11}$$

Seja $(\mathrm{x}_1,\cdots,\mathrm{x}_L)$ uma solução factível para $P$. Como $P\mathrm{x}_k + P\mathrm{x}_{k+1} \le \mathbf{1}$, $k = 1..L-1$ então $(\mathrm{x}_k,\mathrm{x}_{k+1})$, $k = 1..L-1$ são factíveis para *PI*. Como $(\hat{\mathrm{x}}_1,\hat{\mathrm{x}}_2)$ é solução ótima de *PI*, então

$$\frac{L+1}{2}T\mathrm{x}_k + \frac{L-1}{2}T\mathrm{x}_{k+1} \le \frac{L+1}{2}T\hat{\mathrm{x}}_1 + \frac{L-1}{2}T\hat{\mathrm{x}}_2\,,\ k = 1..L-1\,.$$

Logo, para $j = 1..\frac{L-1}{2}$ segue que

$$\left(\frac{L+1}{2} - j\right)\left[\frac{L+1}{2}T\mathrm{x}_{2j-1} + \frac{L-1}{2}T\mathrm{x}_{2j}\right] \le \left(\frac{L+1}{2} - j\right)\left[\frac{L+1}{2}T\hat{\mathrm{x}}_1 + \frac{L-1}{2}T\hat{\mathrm{x}}_2\right] \tag{3.12}$$

$$j\left[\frac{L+1}{2}T\mathrm{x}_{2j+1} + \frac{L-1}{2}T\mathrm{x}_{2j}\right] \le j\left[\frac{L+1}{2}T\hat{\mathrm{x}}_1 + \frac{L-1}{2}T\hat{\mathrm{x}}_2\right] \tag{3.13}$$

Somando as equações (3.12) e (3.13), para $j = 1..\frac{L-1}{2}$ e por (3.11), obtém-se:

$$\left(\frac{L^2-1}{4}\right)\sum_{k=1}^{L} T\mathrm{x}_k \le \left(\frac{L^2-1}{4}\right)\left[\frac{L+1}{2}T\hat{\mathrm{x}}_1 + \frac{L-1}{2}T\hat{\mathrm{x}}_2\right] \overset{(3.11)}{=} \left(\frac{L^2-1}{4}\right)\sum_{k=1}^{L} T\hat{\mathrm{x}}_k$$

Portanto, para $L$ ímpar $(\hat{\mathrm{x}}_1,\cdots,\hat{\mathrm{x}}_L)$ é solução ótima para *PRC-A*.

**Proposição 3**: *Uma solução ótima do modelo de programação de rotações de culturas (PRC-A) cuja configuração espacial da área de plantio tem formato de um tabuleiro pode ser construída a partir de uma solução ótima para 2 lotes adjacentes.*

**Prova**: Seja uma área de plantio no formato de um tabuleiro com $m$ x $n$ lotes. Seja $PT$ o problema $PRC$–$A$ para esta configuração espacial particular, isto é,

$$
(PT)\qquad
\begin{aligned}
\max \quad & T\mathrm{x}_{11} + \cdots + T\mathrm{x}_{1n} + \cdots + T\mathrm{x}_{mn} \\
s.a. \quad & P\mathrm{x}_{ij} + P\mathrm{x}_{i,j+1} \le \mathbf{1},\ i = 1..m,\ j = 1..n-1 \\
& P\mathrm{x}_{ij} + P\mathrm{x}_{i+1,j} \le \mathbf{1},\ i = 1..m-1,\ j = 1..n \\
& \mathrm{x}_{ij} \in S,\ i = 1..m,\ j = 1..n
\end{aligned}
$$

em que $\mathrm{x}_{ij}$ indica a programação de rotação para o lote que ocupa a posição $ij$ no tabuleiro, $S$ o conjunto de todas as programações factíveis para um lote, $T$ o vetor de custos associado a programação para um lote e $P$ a matriz associada às restrições de adjacências (3.6).

Seja o problema $PC$ definido como segue:

$$
(PC)\qquad
\begin{aligned}
\max \quad & T\mathrm{x}_{1,1} + \cdots + T\mathrm{x}_{1,n} + \cdots + T\mathrm{x}_{m,n} \\
s.a. \quad & P\mathrm{x}_{2i-1,j} + P\mathrm{x}_{2i-1,j+1} \le \mathbf{1},\ i = 1..\frac{m+1}{2},\ j = 1..n-1 \\
& P\mathrm{x}_{2i-1,n} + P\mathrm{x}_{2i,n} \le \mathbf{1},\ i = 1..\frac{m-1}{2} \\
& P\mathrm{x}_{2i,1} + P\mathrm{x}_{2i+1,1} \le \mathbf{1},\ i = 1..\frac{m-1}{2} \\
& \mathrm{x}_{i,j} \in S
\end{aligned}
$$

Desta forma, o problema $PC$ define um caminho no grafo G(V, A) percorrendo os $m \times n$ vértices de V da forma: parte da posição (1,1) e percorre a primeira linha até a posição (1, $n$). Em seguida, segue para a posição (2, $n$) e percorre a segunda linha no sentido inverso e assim, sucessivamente.

Seja $n$ par e $(\tilde{\mathrm{x}}_1, \tilde{\mathrm{x}}_2)$ uma solução ótima de $PP$. Pela proposição 1, a solução $(\tilde{\mathrm{x}}_{1,1}, \cdots, \tilde{\mathrm{x}}_{m,n})$, em que $\tilde{\mathrm{x}}_{i,j} = \tilde{\mathrm{x}}_1$ para $i + j$ par e $\tilde{\mathrm{x}}_{i,j} = \tilde{\mathrm{x}}_2$ para $i + j$ impar, é uma solução ótima para $PC$.

Por outro lado, esta solução é factível para o problema $PT$ pois

$$P\hat{\mathrm{x}}_{i,j} + P\hat{\mathrm{x}}_{i,j+1} = P\hat{\mathrm{x}}_1 + P\hat{\mathrm{x}}_2 \le \mathbf{1},\ i = 1..m,\ j = 1..n-1$$

$$P\hat{\mathrm{x}}_{i,j} + P\hat{\mathrm{x}}_{i+1,j} = P\hat{\mathrm{x}}_1 + P\hat{\mathrm{x}}_2 \le \mathbf{1},\ i = 1..m-1,\ j = 1..n\,.$$

Como o problema $PC$ é uma relaxação do problema $PT$, então esta solução é ótima para $PT$. Analogamente, mostra-se que para $m$ par, uma solução ótima é construída a partir de uma solução ótima de $PP$.

Sejam $m.n$ ímpar e $(\hat{x}_1, \hat{x}_2)$ uma solução ótima de $PI$. A solução $(\hat{x}_{1,1}, \cdots, \hat{x}_{m,n})$, em $\hat{x}_{i,j} = \hat{x}_1$ para $i + j$ par e $\hat{x}_{i,j} = \hat{x}_2$ para $i + j$ impar, é ótima para o problema $PC$ pela proposição 1. Além disto, esta solução é factível para o problema $PT$. Como o problema $PC$ é uma relaxação do problema $PT$, então esta solução é ótima para $PT$.

■

Como mencionado anteriormente, para grafos quaisquer G com $n$ subconjuntos independentes, mesmo que exista uma solução para $K_n$, não se tem garantia de que a solução construída para G com a solução de $K_n$ seja ótima. Por outro lado, pode ser desejável mudar a função objetivo do problema PRC–A ou incluir outros conjuntos de restrições no modelo, tais como, favorecer ou obrigar o plantio de determinadas culturas ou ainda considerar o dimensionamento dos lotes de plantio etc. Nesses casos, a solução de $K_n$ pode não ter relação com a solução de G.

No Capítulo 5 propomos procedimentos heurísticos e exatos para encontrar boas ou ótimas soluções para o problema PRC–A. Naturalmente, os procedimentos são abrangentes e não consideram somente o caso da redução do problema em G no problema associado $K_n$. No Capítulo 6 apresentamos e discutimos os resultados computacionais obtidos com os procedimentos propostos.

# Capítulo 4

# Programação de Rotação de Culturas com atendimento de Demanda (PRC–D)

Em resposta aos inúmeros problemas sociais, econômicos e ambientais gerados pela agricultura moderna, diversos modelos alternativos de agricultura foram propostos e estão sendo desenvolvidos em todo o mundo (Ehlers, 1996). Estas propostas têm diferentes pontos de foco, ora mais técnicos, ora mais ecológicos ou econômicos, ou por vezes mais centrados na organização dos agricultores. Contudo, a maioria deles apresenta princípios semelhantes e que consideram além de aspectos ecológicos de produção, o acesso de pequenos agricultores ao mercado.

Os produtores familiares possuem, em geral, pequenas áreas de plantio e pequeno volume de produção. A produção é usualmente descontínua devido, principalmente, a condições climáticas e do solo que restringem o cultivo de determinadas espécies em determinadas regiões ou reduzem em muito sua produtividade. Por outro lado, aspectos econômicos muitas vezes exigem que seja atingido um volume mínimo de produção para viabilizar custos de logística.

Para lidar com estes problemas uma possibilidade é a união de pequenos produtores agrícolas em movimentos coletivos, tais como cooperativas ou associações, para que os requisitos mínimos de constância, volume e uniformidade da produção sejam atingidos. No sudeste do Brasil existem exemplos dessas associações como Sítio do Moinho (RJ), Horta e Arte (SP), além de feiras coletivas (Campanhola e Valarini, 2001). Esses empreendimentos coletivos devem organizar não somente o tamanho das áreas de plantio com cada cultura, para responder a demanda de mercado, mas também decidir onde e quando a produção ocorrerá. Por eventuais preferências do produtor, tipo de solo e condições climáticas, a produtividade de uma cultura, bem como o conjunto de culturas para o plantio, é dependente da área de plantio.

Neste Capítulo consideramos um problema de planejamento de produção de hortaliças em que deseja-se atender a demanda de um conjunto de hortaliças em um

dado intervalo de tempo. Como o problema é voltado para uma produção de base ecológica, foram inseridas na programação do plantio das culturas as restrições de plantio discutidas no Capítulo 2. Na Seção 4.1, apresentamos o problema e a formulação matemática utilizada para descrevê–lo. Na Seção 4.2 algumas extensões do modelo que incorporam certas situações práticas são discutidas.

## 4.1 Um modelo para o problema PRC-D

Como definido anteriormente, uma programação de rotação de cultura é um calendário de plantio (em uma base de tempo escolhida) de uma rotação de cultura de tamanho *M*. O plantio de cada cultura na rotação implica em um ou mais períodos de colheita. Nomeamos *programação de produção de rotação de culturas* (ou simplesmente, *programação de produção*) ao calendário de colheita das culturas na rotação com as quantidades (em kg, unidade, ou outra medida apropriada) que se espera colher por unidade de área.

Considere o exemplo apresentado na Seção 2.2. Suponha que a cultura X tenha 1 período entre o período de plantio e o período da primeira colheita e, até o final do seu ciclo de 5 períodos, tenha 3 períodos sucessivos de colheitas, com produções estimadas em 1 kg, 2 kg e 1 kg por $m^2$. A cultura Y tem sua produção estimada em 3 $kg/m^2$, com a colheita ocorrendo no ultimo período do seu ciclo de cultivo de 4 períodos. Com estes dados e com a programação de rotação exibida na Figura 2.1, a programação de produção para uma área de plantio de $2m^2$ é construída. Na Figura 4.1, em itálico, são exibidas as quantidades que se espera colher por período das culturas X e Y para a área de $2m^2$, além da programação da rotação de culturas associada a esta programação de produção.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *6* | **P** | **X** | | | *2* | *4* | *2* | **Z** | | **Y** | |
| **Período:** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |

Figura 4.1: programação de rotação e associada produção.

Como exibido na Figura 2.1, a cultura X é plantada no período 3, Z no período 8, Y em 10 e um pousio (P) de 1 período foi programado para o 2. A nova informação na Figura 4.1, diz respeito às colheitas das culturas. Nos períodos 5, 6 e 7, estima–se uma colheita de 2, 4 e 2 kg da cultura X e, no período 1, a colheita de 6 kg da cultura Y. A cultura Z é uma adubação verde e, portanto, não tem colheita associada.

***Formulação Matemática***

Considere um conjunto de culturas $C$ com uma demanda conhecida em um intervalo de tempo, tendo–se disponível para o plantio das culturas um conjunto de áreas $A_1, \cdots, A_L$. Devido, principalmente, a condições climáticas e características do solo, as culturas passíveis de plantio, bem como as suas produtividades, dependem da área de plantio. Para o atendimento da demanda das culturas as áreas de plantio podem ser divididas em lotes e, para cada um deles, determina–se uma programação de produção de culturas (associada a uma programação de rotação) do tamanho do intervalo de tempo considerado.

Este problema de dimensionamento de lotes de plantio, nomeado *Programação de Rotação de Culturas com atendimento de Demanda* (PRC–D), é formulado com um modelo de otimização linear no qual as variáveis correspondem ao tamanho dos lotes das áreas de plantio. Para a construção da formulação matemática para este problema, definimos os seguintes parâmetros:

$C_k$ conjunto de culturas que podem ser plantadas na área $k$;
$S_k$ conjunto das possíveis programações de produção para a área $k$;
$\overline{I}_i$ conjunto de possíveis períodos de colheita para a cultura $i$.

e variáveis:

$\lambda_{ks}$ tamanho do lote associado à programação $s \in S_k$.

Para definir a programação de produção $s \in S_k$, considere $a^s_{ijk}$ a quantidade por unidade de área da cultura $i \in C_k$ produzida no período $j \in \overline{I}_i$ na área $k$ de acordo com a programação $s$.

Desta forma, o modelo para o problema PRC-D é escrito como:

$$z_{PRCD} = \max \sum_{k=1}^{L} \sum_{s \in S_k} c_{ks} \lambda_{ks} \tag{4.1}$$

$$s.a. \quad \sum_{k=1}^{L} \sum_{s \in S_k} a^s_{ijk} \lambda_{ks} \geq d_{ij}\,,\ j \in \overline{I}_i\,,\ i \in C_k\,,\ k = 1..L \tag{4.2}$$

$$\sum_{s \in S_k} \lambda_{ks} \leq A_k\,,\ k = 1..L \tag{4.3}$$

$$\lambda_{ks} \geq 0\ ,\ s \in S_k\,,\ k = 1..L \tag{4.4}$$

em que $d_{ij}$ é a demanda (em uma unidade apropriada de medida) da cultura $i$ no período $j$ e $c_{ks}$, o retorno obtido com a programação $s$ por unidade de área $k$.

O objetivo do modelo (4.1)–(4.4) é maximizar o retorno obtido com todas as programações de produção o que pode ser, por exemplo, volume de produção, receita etc. As restrições (4.2) garantem que as demandas para todas as culturas em todos os períodos sejam atendidas enquanto as restrições (4.3) asseguram que não mais que a área total disponível seja usada.

O modelo contém um número de grande de variáveis devido ao fato de que o número de possíveis programações (de produção) de rotações de culturas para cada área, cresce exponencialmente com o número de períodos e de culturas. Por esta razão, o problema foi resolvido usando a técnica de geração de colunas. O procedimento é descrito na Seção 5.2 e, na Seção 6.2, os resultados computacionais obtidos são analisados.

Como uma programação de produção é completamente definida por uma programação de rotação, a formulação (4.1)–(4.4) pode ser escrita em função das programações de rotações para os lotes. Para isto, considere $\mathbf{x}_k^s = (\hat{x}_{ijk}^s)$ a $s$–ésima programação de rotação de culturas para a área $k$, em que $\hat{x}_{ijk}^s = 1$, se a cultura $i \in C_k$ é plantada no período $j \in I_i$ na área $k$ e zero, caso contrário. Em relação à colheita das culturas, sejam:

$o_i$ número de períodos entre o plantio e a primeira colheita da cultura $i \in C$;

$p_{ikr}$ quantidade obtida na $r$–ésima colheita da cultura $i \in C_k$ por unidade de área $k$, para $r = 1 \cdots t_i - o_i - 1$

Assim, a programação de produção $\mathbf{a}_k^s = (a_{ijk}^s)$ associada à programação $\hat{\mathbf{x}}_k^s$ é definida por:

$$a_{i,j+r+o_i,k}^s = p_{ikr}\hat{x}_{ijk}^s, \text{ para, } j + o_i + r \in \overline{I}_i,\ i \in C_k,\ s \in S_k,\ k = 1..L.$$

Para o caso em que se deseja maximizar a produção na área de plantio, o retorno $c_{ks}$ obtido com a programação de produção $\mathbf{a}_k^s = (a_{ijk}^s)$ pode ser escrito como:

$$c_{ks} = \sum_{i \in C_k} \sum_{j \in I_i} \sum_{r=1}^{t_i - o_i - 1} p_{ikr}\hat{x}_{ijk}^s$$

Portanto, em termos de uma programação de rotação de culturas $\mathbf{x}_k^s = (\hat{x}_{ijk}^s)$, o modelo (4.1)–(4.4) torna–se:

$$z_{PRCD-p} = \max \sum_{k=1}^{L} \sum_{s \in S_k} \sum_{i \in C_k} \sum_{j \in I_i} \sum_{r=1}^{t_i - o_i - 1} p_{ikr} \hat{x}_{ijk}^s \lambda_{ks}$$

*sujeito a:*

$$\sum_{k=1}^{L} \sum_{s \in S_k} p_{ir} \hat{x}_{i,j-o_i-r,k}^s \lambda_{ks} \geq d_{ij}, \; j - o_i - r \in I_i, \; i \in C_k$$

$$\sum_{s \in S_k} \lambda_{ks} \leq A_k, \; k = 1..L$$

$$\lambda_{ks} \geq 0 \; , \; s \in S_k, \; k = 1..L$$

Obs.: $j - o_i - r$ é substituído por $j - o_i - r + M$ se $j - o_i - r \leq 0$.

Na Seção 2.2 foi apresentada uma formulação matemática para as programações de rotações de culturas sujeito a restrições de base ecológica. Para o problema PRC–D utilizamos aquele modelo. Desta forma, considere $G_k$ o conjunto de culturas para adubação verde para a área $k$ e $\tilde{G}_k = G_k \cup \{N+1\}$, em que $N+1$ é a cultura fictícia representando o pousio obrigatório em cada lote. Como já observado anteriormente, as culturas $i \in \tilde{G}_k$ não são consideradas na função objetivo (4.1) mas devem ser obrigatoriamente produzidas em função restrições de base ecológica. Assim, uma programação de rotação de culturas para um lote $k$ é qualquer solução do sistema:

$$\sum_{i \in C_k \cup \tilde{G}_k} \sum_{r=0}^{t_i - 1} x_{i,j-r} \leq 1, \; j = 1..M \tag{4.5}$$

$$\sum_{i \in F(p) \cap C_k} \sum_{r=0}^{t_i} x_{i,j-r} \leq 1, \; p = 1..NF, j = 1..M \tag{4.6}$$

$$\sum_{i \in G_k} \sum_{j \in I_i} x_{ij} = 1 \tag{4.7}$$

$$\sum_{j=1}^{M} x_{N+1,j} = 1 \tag{4.8}$$

$$x_{ij} \in \{0,1\}, \; i \in C_k \cup \tilde{G}_k, \; j \in I_i \tag{4.9}$$

Obs.: nas equações (4.5)–(4.6), $j - r \leq 0$ é substituído por $j - r + M$.

**Exemplo**: considere 1 ano particionado em 12 meses ($M = 12$) e as 8 culturas listadas na Tabela 4.1. Para cada cultura $i$, a tabela exibe a família botânica à qual pertence ($F_i$), o conjunto de possíveis períodos de plantio ($I_i$), o ciclo de cultivo ($t_i$), o número de períodos entre o plantio e a primeira colheita ($o_i$) e a quantidade que se espera colher na $r$–ésima colheita parcial ($p_{ir}$). A tabela também exibe a demanda das culturas 1 a 6 em cada período. As culturas 7 e 8 são para adubação verde e, portanto, não tem demanda associada.

Tabela 4.1: dados do problema

| | *Dados das culturas* | | | | | | | *Demanda por período* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_i$ | $I_i$ | $t_i$ | $o_i$ | $p_{i1}$ | $p_{i2}$ | $p_{i3}$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **1** | 1 | 1-7; 11-12 | 4 | 2 | 1 | 1 | | | | 1 | 1 | | | | | | | | |
| **2** | 1 | 1-9 | 3 | 2 | 1 | | | | | | 1 | | | | | | | | |
| **3** | 2 | 3-11 | 3 | 1 | 1 | 1 | | 2 | | | 1 | 1 | | | 1 | 1 | | | |
| **4** | 2 | 2-11 | 4 | 1 | 1 | 2 | 1 | 4 | | | | 8 | 6 | 8 | | | 2 | 8 | 10 |
| **5** | 3 | 7-12; 1-5 | 3 | 1 | 1 | 1 | | | | 1 | 1 | | | | | | | | |
| **6** | 3 | 3-10 | 3 | 1 | 2 | 1 | | | | | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 2 |
| **7** | 4 | 2-12 | 3 | 2 | 1 | | | | | | | | | | | | | | |
| **8** | 4 | 8-12; 1-6 | 2 | 1 | 1 | | | | | | | | | | | | | | |

Sejam três áreas com tamanhos $A_1 = 6$, $A_2 = 4$ e $A_3 = 5$ m$^2$ para o plantio das culturas. A cultura 1 não pode ser plantada na área 1, as culturas 2, 3 e 5, não são plantadas na área 2 e nem as culturas 2 e 4 na área 3. A produtividade nas áreas 1, 2 e 3 são 110%, 100% e 80% dos valores apresentados na Tabela 4.1, respectivamente.

Estabelecendo–se um período de 1 mês de pousio obrigatório em cada rotação, para o atendimento da demanda as áreas 1 e 2 foram divididas em 2 lotes e a área 3 foi dividida em 3 lotes. A solução ótima é dada por: $\lambda_{11} = 1\,\text{m}^2$, $\lambda_{12} = 5\,\text{m}^2$, $\lambda_{21} = 1\,\text{m}^2$, $\lambda_{22} = 2\,\text{m}^2$, $\lambda_{31} = 2{,}5\,\text{m}^2$, $\lambda_{32} = \lambda_{33} = 1{,}25\,\text{m}^2$. A programação de rotação para cada lote, assim como a colheita esperada de cada cultura na programação, são exibidos na Figura 4.2. O período de 1 mês de pousio em cada rotação está indicado pela cultura fictícia 9.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Area 3 (5 m²)** | **3** | **(1.25 m²)** | **8** | | **3** | *1* | *1* | **6** | *2* | *1* | **9** | **6** | *2* | *1* |
| | **2** | **(1.25 m²)** | **9** | **5** | *1* | *1* | **8** | | **3** | *1* | *1* | **6** | *2* | *1* |
| | **1** | **(2.5m²)** | *2* | **9** | **6** | *4* | *2* | **8** | | **6** | *4* | *2* | **3** | *2* |
| **Area 2 (4 m²)** | **2** | **(3m²)** | *3* | **9** | | **4** | *3* | *6* | *3* | **8** | | **4** | *3* | *6* |
| | **1** | **(1m²)** | **1** | | *1* | *1* | **9** | **8** | | **4** | *1* | *2* | *1* | |
| **Area 1 (6 m²)** | **2** | **(5m²)** | *5.5* | **9** | | **4** | *5.5* | *10.1* | *5.5* | **8** | | **4** | *5.5* | *10.1* |
| | **1** | **(1m²)** | | **2** | | *1.1* | **6** | *2.2* | *1.1* | **9** | **6** | *1.1* | *1.1* | **8** |
| | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |

Figura 4.2. Programações de produção para as áreas de plantio.

## 4.2 Discussões sobre o modelo e extensões

Na seção anterior, um modelo linear foi apresentado para o problema PRC-D. Nesta seção, algumas características do modelo são analisadas. Na subseção 4.2.1, mostramos como o modelo (4.1)–(4.4) pode ser simplificado, caso as áreas sejam homogêneas, ou seja, as culturas podem ser plantadas em todas áreas e a produtividade de uma cultura independe da área. Na subseção 4.2.2 identificamos certas situações encontradas na prática e propomos como integrá–las ao modelo. Finalmente, na subseção 4.2.3 uma questão referente ao número de programações de produções na solução ótima é discutida.

### 4.2.1 Áreas homogêneas

Se não existe distinção entre as áreas $A_1, \cdots, A_L$, ou seja, qualquer cultura pode ser plantada em qualquer uma delas, com o mesmo rendimento, o modelo (4.1)–(4.4) pode ser simplificado e as $L$ áreas tratadas como uma única área $A = A_1 + \cdots + A_L$. Assim, uma mesma programação de produção pode ser atribuída a qualquer lote. Neste caso, podemos ignorar o índice $k$ na formulação e reescrever o modelo como:

$$z_{PRCD-h} = \max \sum_{s \in S} c_s \lambda_s$$

*sujeito a:*

$$\sum_{s \in S} a_{ij}^s \lambda_s \geq d_{ij},\ i \in C, j \in \overline{I}_i$$

$$\sum_{s \in S} \lambda_s \leq A$$

$$\lambda_s \geq 0\,,\ s \in S$$

As novas variáveis $\lambda_s$ indicam o tamanho do lote usado com a programação $s$, considerando $S$ o conjunto de todas as possíveis programações para $A$. Seja $\hat{\lambda}_1, \cdots, \hat{\lambda}_K$ uma solução ótima para o modelo *PRCD–h*. Uma solução para a divisão da área de plantio em $L$ áreas, $A_1, \cdots, A_L$, pode ser obtida com a ajuda do seguinte sistema de restrições:

$$\sum_{k=1}^{L} \lambda_{ks} = \hat{\lambda}_s\,,\ s = 1..K$$

$$\sum_{s=1}^{K} \lambda_{ks} \leq A_k\,,\ k = 1..L$$
$$\lambda_{ks} \geq 0,\ s = 1..K,\ k = 1..L$$

### 4.2.2 Não atendimento da demanda e limitante superior para a produção

Em situações práticas pode ser desejável permitir soluções em que nem todas as demandas são atendidas. Esta opção pode ser incorporada ao modelo (4.1)–(4.4) com a inclusão de variáveis artificiais nas restrições (4.2) e penalizando–as na função objetivo. O modelo (4.1)–(4.4) pode ser então modificado da forma:

$$z_{PRCD-r} = \max \sum_{k=1}^{L} \sum_{s \in S_k} c_{ks}\lambda_{ks} - \sum_{i \in C} \sum_{j \in \overline{I}_i} r_{ij}\theta_{ij}$$

*sujeito a:*

$$\sum_{k=1}^{L} \sum_{s \in S_k} \mathbf{a}_k^s \lambda_{ks} \geq \mathbf{d} - \boldsymbol{\theta}$$

$$\sum_{s \in S_k} \lambda_{ks} \leq A_k, k = 1..L$$

$$\boldsymbol{\theta} = (\theta_{ij}) \geq 0\,,\ \lambda_{ks} \geq 0,\ s \in S_k,\ k = 1..L$$

em que $\theta_{ij}$ indica a quantidade de demanda não atendida da cultura $i$ no período $j$ e $r_{ij}$ é a penalidade associada a este não atendimento, para $i \in C$ e $j \in \overline{I}_i$.

Além de permitir que parte da demanda não seja atendida, pode ser conveniente estabelecer um limite para a produção das culturas. A motivação pode ser a capacidade de absorção do mercado, por exemplo. Para isto, basta introduzir limitantes superiores nas restrições (4.2). Na Seção 6.2, os efeitos de permitir o não–atendimento da demanda e inserir limitantes na produção são analisados.

### 4.2.3 Considerações sobre o número de lotes

O modelo (4.1)–(4.4) não impõe restrições sobre o número de programações de produções associadas a uma área de plantio, permitindo soluções com um número grande de lotes. Para reduzir este tipo de ocorrência, que pode ser uma exigência em

situações reais, um custo fixo $z_k$ pode ser associado às variáveis $\lambda_{ks} > 0$. Com o auxílio de uma variável binária $\delta_{ks}$, que indica se a $s$–ésima programação para a área $k$ foi ou não usada, o valor escolhido para $z_k$ fornece um balanço entre o número de lotes para a área $k$ na solução e o retorno obtido com as programações para esta área. Desta forma, a função objetivo é então substituída por:

$$\max \sum_{k=1}^{L} \sum_{s \in S_k} c_{ks}\lambda_{ks} + z_k \sum_{s \in S_k} \delta_{ks} \qquad (4.10)$$

com a seguinte restrição adicional:

$$\lambda_{ks} \leq A_k \delta_{ks}\,,\ \delta_{ks} = \{0,1\}\,, s \in S_k\,,\ k = 1..L \qquad (4.11)$$

Naturalmente, a formulação (4.10), sujeito a (4.2)-(4.3) e (4.11) não é um problema linear e, sim, um problema inteiro-misto bem mais complexo. Na Seção 5 são discutidos procedimentos heurísticos para determinar soluções para este modelo.

No modelo PRC–D, a programação de produção é função de uma rotação de culturas. Desta forma, estamos considerando um planejamento em longo prazo em que a demanda se estabiliza. A formulação proposta pode ser facilmente adaptada para o caso em que se deseja apenas a programação de uma sequência de culturas, sem a imposição de que esta sequência seja repetida sucessivamente.

# Capítulo 5

# Métodos de solução para os problemas PRC–A e PRC–D

Nos capítulos 3 e 4 caracterizamos os problemas PRC–A e PRC–D de planejamento de produção de hortaliças. Neste capítulo apresentamos os procedimentos heurísticos e exatos usados para resolvê–los. Na Seção 5.1 fazemos uma breve introdução das metodologias utilizadas e, nas seções subsequentes, 5.2 e 5.3, como elas foram aplicadas, respectivamente, aos problemas PRC–A e PRC–D.

## 5.1 Métodos de Solução

Para o problema PRC–A foi proposto um modelo de Programação Linear Inteira (PLI) em que as variáveis são binárias (PL 0–1) e, para o problema PRC–D, um modelo de Programação Linear (PL), em que cada coluna é obtida resolvendo–se um PL 0–1.

Em geral, a resolução de um modelo de otimização linear torna–se bem mais complexa com a presença de variáveis inteiras. Boa parte dos esforços de pesquisa nesta área concentra-se no desenvolvimento de métodos e algoritmos para classes particulares de problemas. Os princípios dos métodos desenvolvidos para estes exemplos particulares podem, em muitos casos, ser estendidos a outras classes, produzindo bons (excelentes) resultados. Neste contexto, um estudo das características do problema é essencial quando se deseja obter um algoritmo de resolução eficiente. Muitas vezes, a melhor estratégia pode ser resultado da combinação de várias metodologias. A literatura sobre os principais conceitos e metodologias para problemas lineares (inteiros) é bastante vasta e pode ser encontrada em, por exemplo, Nemhauser e Wolsey (1988), Wolsey (1998) e Johnson et al. (2000).

Para a resolução de problemas lineares, seja no caso em que o PL é o problema em estudo ou que a resolução do PL é parte integrante de um método para um PLI, o método simplex é um dos mais utilizados. O método explora a propriedade de que uma solução ótima pode ser pesquisada nos pontos extremos da região factível. Nos casos em que existe uma lei de formação para as colunas do modelo e o custo de uma variável é dado em função da coluna correspondente, não é necessário que as variáveis (colunas) sejam definidas explicitamente, podendo ser obtidas com a resolução de um subproblema. Esta metodologia de enumeração implícita, nomeada geração de colunas, é bastante empregada para resolver problemas lineares ou a relaxação linear de problemas lineares inteiros em que o número de variáveis é muito grande e sua enumeração completa é impraticável ou pouco eficiente. O método itera entre um problema mestre restrito (contendo algumas colunas já geradas) e um subproblema gerador de colunas. A solução do problema mestre fornece as variáveis duais que são usadas para guiar os subproblemas que determinam novas colunas para o problema mestre.

Para um PLI, em geral, não existe uma condição necessária e suficiente de otimalidade. Uma forma de determinar se uma solução é ótima é demonstrar que não existe solução factível com melhor valor. No entanto, o número de soluções viáveis pode ser infinito ou, mais comumente, enorme. Uma alternativa é enumerar parcialmente um número tratável de possibilidades e enumerar implicitamente todo o resto. O método *branch-and-bound*, proposto inicialmente por Land e Doig (1960), é o método mais adotado para resolver um PLI com a abordagem de enumeração implícita. Com uma estratégia de *divida–e–conquiste*, os ramos de uma árvore de enumeração das possíveis soluções são percorridos de forma sistemática, evitando ramos que levam a infactibilidades ou a soluções piores que as já encontradas, usando limitantes superiores e inferiores para a função objetivo como mecanismo para a eliminação de nós da árvore.

Os principais fatores que afetam a eficiência de um algoritmo *branch–and–bound* são a qualidade dos limitantes, a regra de ramificação e a escolha do próximo nó a ser ramificado. Achterberg et al. (2004) discutem um conjunto de regras de ramificação eficientes para classes de PLI. Dentre os procedimentos comumente empregados para obtenção de limitantes de melhor qualidade estão a inserção de planos de cortes na árvore de ramificação ou a reformulação do problema, usando o princípio da Decomposição de Dantzig–Wolfe (Dantzig e Wolfe, 1960), por exemplo.

O método de Decomposição Dantzig–Wolfe foi originalmente proposto para resolver um PL em que o conjunto de restrições pode ser decomposto em subconjuntos independentes ligados por um conjunto de restrições. As restrições de acoplamento, reformuladas pelas combinações lineares dos pontos e raios extremos dos politopos definidos pelos subconjuntos independentes, com as restrições de convexidade formam um problema mestre equivalente ao problema original enquanto as restrições dos subconjuntos independentes definem os subproblemas. Quando um PLI é reformulado usando o principio de decomposição de Dantzig–Wolfe, o limitante obtido com a relaxação linear do problema mestre é igual ao obtido pela relaxação lagrangiana quando as restrições dualizadas são as de acoplamento (Wolsey, 1998). O método é geralmente resolvido por geração de colunas pois o número de vértices e raios extremos dos politopos é, usualmente, muito grande.

Quando a relaxação linear de um PLI é resolvida por geração de colunas não se tem garantia de obtenção de uma solução inteira. Para isto, uma opção é a utilização de um algoritmo exato *branch–and–price* no qual a relaxação linear nos nós da árvore de enumeração é resolvida por geração de colunas. Algoritmos do tipo *branch–and–price* têm sido aplicados com sucesso na resolução de várias classes de problemas inteiros difíceis e de grande dimensão, desde a sua primeira aplicação bem sucedida em um problema de roteamente de veículo por Desrosiers et al. (1984). Barnhart et al. (1998), Wilhelm (2001), Maculan et al. (2003) e Lübbecke e Desrosiers (2005) apresentam revisões da literatura de aplicações de geração de colunas (incluindo *branch–and–price*) na resolução de um PLI.

Os métodos de plano de corte consistem em introduzir sucessivamente novas restrições (desigualdades válidas) na relaxação linear do PLI de modo que o conjunto das soluções factíveis seja restringido, eliminando algumas soluções mas sem eliminar soluções inteiras. Os métodos baseiam-se no fato de que a solução ótima de um PL ocorre em vértices do espaço de soluções. Desta forma, quanto mais próxima a região das soluções factíveis de um PL estiver do envoltório convexo da região das soluções factíveis do PLI, mais rapidamente pode-se chegar a uma solução ótima ou a um limitante de melhor qualidade. Os métodos *branch–and–cut* incorporam uma fase de plano de cortes em alguns (ou todos) nós da árvore de enumeração de um algoritmo *branch–and–bound* e estão presentes em vários pacotes de otimização comerciais (CPLEX, XPRESS etc.) e não comerciais (SYMPHONY, MINTO etc.). Marchand et al. (2002) fazem uma revisão da literatura sobre esta metodologia.

Como a maioria dos PLI são NP–difíceis, heurísticas e meta–heurísticas bem como algoritmos aproximativos são usualmente considerados, dependendo da complexidade do problema ou quando é necessário que uma solução factível seja obtida rapidamente. A combinação de meta–heurísticas com métodos exatos tem se mostrado uma estratégia eficiente na resolução de problemas dessa natureza. Raidl e Puchinger (2008) apresentam um conjunto de aplicações dessas combinações.

Na próxima seção apresentamos os procedimentos usados para resolver as formulações propostas para o problema PRC–A bem como estratégias propostas a fim de melhorar a eficiência dos algoritmos dos tipos *branch–and–bound* e *branch–and–price* implementados para os modelos. Na Seção 5.3, as metodologias empregadas para resolver os modelos propostos para o problema PRC–D são discutidas.

## 5.2 Métodos exatos e heurísticos para o problema PRC-A

O modelo de otimização 0–1 proposto para o problema PRC–A no Capítulo 3 apresenta um número grande de restrições e variáveis dificultando sua resolução por um algoritmo *branch–and–bound* padrão. Uma alternativa para diminuir a dimensão do problema é decompô–lo em subproblemas, cada um com um número menor de lotes, garantindo–se que ao agrupar os subconjuntos de lotes as restrições de adjacências (3.6) para os lotes estejam satisfeitas. Esta estratégia de decomposição é utilizada tanto nas heurísticas propostas para o problema quanto na reformulação do modelo 0–1 permitindo sua resolução pela técnica de geração de colunas em procedimentos heurísticos e exatos.

### 5.2.1 Heurísticas Construtivas

Nesta subseção descrevemos duas heurísticas construtivas gulosas para o problema PRC–A, em que o conjunto de lotes foi dividido em subconjuntos. As heurísticas foram utilizadas para determinar rapidamente boas soluções factíveis para o problema e também foram combinadas com algoritmos *branch–and–bound* e *branch–and–price* na tentativa de melhorar a eficiência desses métodos quando usados para resolver o problema PRC–A.

Dado um conjunto de $L$ lotes, uma programação factível para $R \leq L$ lotes, pode ser obtida com o seguinte procedimento heurístico:

---

1. O conjunto de $L$ lotes é particionado nos subconjuntos $S_1, S_2, \ldots, S_K$.
2. Busque uma boa programação factível (ou ótima) para $S_1$. Caso não exista, então **PARE**. O problema PRC–A não tem solução. Caso contrário, faça $k = 1$.
3. Se $k$ = K, então PARE. Uma solução factível para os $L$ lotes foi encontrada. Caso contrário, para $S_{k+1}$ são fixadas em zero as variáveis que garantem que as restrições (3.6) sejam satisfeitas, ou seja, com a alocação já fixada das culturas para os subconjuntos $S_1, S_2, \cdots S_k$, a alocação das culturas em $S_{k+1}$ deve ser de modo que não viole as restrições de adjacências para culturas de mesma família botânica.
4. Busque uma boa programação factível (ou ótima) para $S_{k+1}$. Se tal programação não existe, então **PARE**. Neste caso, uma solução para $R = \mid S_1 \mid + \mid S_2 \mid + \cdots + \mid S_k \mid$ lotes foi obtida. Caso contrário, faça $k = k + 1$ e retorne ao passo em 3.

---

A eficiência desta heurística gulosa depende tanto do particionamento do conjunto de lotes quanto da ordem estabelecida para percorrer os subconjuntos de lotes. Em particular, para $K = L$ (ou seja, $\mid S_k \mid = 1$, para $k = 1..L$), as programações de rotação para os lotes são determinadas, lote por lote, usando uma estratégia de busca em largura começando por um lote com maior número de adjacências. Para a maioria das instâncias geradas aleatoriamente, uma solução para os $L$ lotes foi obtida e em tempo computacional bem pequeno. Isto ocorreu pois cada subconjunto de lotes tem apenas um lote e, portanto, as restrições (3.6) foram eliminadas dos subproblemas associados a esses subconjuntos tornando–os bem mais simples de serem resolvidos.

Esta heurística nomeada *lote-a-lote* foi inserida em alguns dos procedimentos baseados em geração de colunas implementados e que são discutidos nas subseções subsequentes. Quando a garantia de uma solução factível ($R = L$) é necessária, uma heurística similar foi utilizada. A principal diferença é um passo adicional no início do procedimento em que se busca uma solução factível para o problema PRC–A com apenas adubação verde e pousio. Desta forma, fixando os valores das variáveis

associadas a esta solução inicial, tenta–se encontrar programações de melhor qualidade para os $L$ lotes.

Os principais passos da heurística *lote–a–lote–completa* são descritos por.

---

**Algoritmo: Heurística *lote–a–lote–completa***

---

5. Busque uma programação factível para os $L$ lotes com apenas adubação verde e pousio em cada lote. Se não existe tal programação, então PARE. O problema PRC–A não tem solução. Caso contrário, faça $k = 1$ e ordene os $L$ lotes.
6. Se $k = L$, então PARE. Uma programação para os $L$ lotes foi obtida. Caso contrário, para o lote $k$+1, fixe em zero as variáveis que garantam que:
   - 1.1 As restrições (3.1) de ocupação de espaço sejam satisfeitas com as variáveis fixadas pelo passo anterior.
   - 1.2 As restrições (3.3) de vizinhança temporal para culturas de mesma família sejam satisfeitas para este lote;
   - 1.3 As restrições (3.6) de vizinhança espacial sejam satisfeitas para o lote $k$+1 e os lotes adjacentes a ele, já alocados nas iterações anteriores.
7. Encontre uma programação factível (ou ótima) para o lote $k+1$. Faça $k = k+1$ e volte em 2.

---

A heurística *lote–a–lote–completa* foi usada para gerar soluções iniciais para certos algoritmos *branch–and–bound* e *branch–and–price* implementados. Além disto, para alguns deles também foi usada em nós da árvore de enumeração na tentativa de obtenção de soluções factíveis mais rapidamente para o problema. Nestes casos, o passo 1 da heurística foi substituído por:

1'. Considerando as variáveis já fixadas em 0 ou 1 (em função da ramificação) do nó, busque uma programação factível para os $L$ lotes com apenas adubação verde e pousio. Se não existe programação, então PARE. O nó pode ser podado. Caso contrário, faça $k = 1$.

### 5.2.2 Reformulação do modelo

O problema PRC–A para $L$ lotes também pode ser visto como o problema de escolher dentre todas as programações factíveis para um lote (isto é, satisfazendo as restrições (3.1)–(3.5)), $L$ programações que satisfazem as restrições de adjacências (3.6) ou (3.8). Isto corresponde a aplicar, convenientemente, o princípio de decomposição Dantzig–Wolfe à formulação original.

Seja $S = \{\hat{\mathrm{x}}^1, \hat{\mathrm{x}}^2, \cdots, \hat{\mathrm{x}}^h\}$ o conjunto de todas as programações de rotações factíveis para um lote. As componentes da $s$–ésima programação $\hat{\mathbf{x}}^s$ de $S$ são $\hat{x}^s_{ij} = 1$, se a cultura $i$ é plantada no período $j$, e $\hat{x}^s_{ij} = 0$, caso contrário, $i = 1..\ n,\ j \in I_i$.

Sendo $\mathbf{x}_k = (x_{ijk})$ uma programação para o lote $k$, então:

$$\mathbf{x}_k = \sum_{s=1}^{h} \lambda_{sk} \hat{\mathbf{x}}^s \text{ , com } \sum_{s=1}^{h} \lambda_{sk} = 1 \text{ e } \lambda_{sk} \in \{0, 1\}.$$

Portanto,

$$x_{ijk} = \sum_{s=1}^{h} \lambda_{sk} \hat{x}^s_{ij} \text{ , com } \sum_{s=1}^{h} \lambda_{sk} = 1 \text{ e } \lambda_{sk} \in \{0, 1\},\ i = 1..N+1, j \in I_i, k = 1..L.$$

Desta forma, o modelo é reformulado em um problema mestre:

$$\text{(PM)} \qquad z_{PM} = \max \sum_{k=1}^{L} \sum_{i \in C} \sum_{j \in I_i} \sum_{s=1}^{h} t_i \hat{x}^s_{ij} \lambda_{sk} \tag{5.1}$$

*Sujeito a*:

$$\sum_{i \in F(p)} \sum_{s=1}^{h} \sum_{\substack{r=0, \\ j-r \in I_i}}^{t_i - 1} \left[ m_k \hat{x}^s_{i(j-r)} \lambda_{sk} + \sum_{(u,k) \in A} \hat{x}^s_{i(j-r)} \lambda_{su} \right] \leq m_k,\ p = 1..NF,\ j = 1..M,\ k = 1..L \tag{5.2}$$

$$\sum_{s=1}^{h} \lambda_{sk} = 1,\ k = 1..L \tag{5.3}$$

$$\lambda_{sk} \in \{0, 1\},\ s = 1..h,\ k = 1..L \tag{5.4}$$

em que o conjunto $S$ das programações de rotações de culturas factíveis para um lote é como foi definido e modelado no Capítulo 2.

Com a reformulação do modelo PRCA, a diferença entre os limitantes obtidos com a relaxação linear do problema mestre e a relaxação linear da formulação original pode ser muito grande. Considere, por exemplo, $M = 3$ e duas culturas em famílias botânicas distintas, uma delas para adubação verde. Ambas com ciclo de cultivo 1 e sem restrições de época de plantio. Para uma configuração de área de plantio que corresponde a $K_3$, o valor ótimo do modelo PRCA e o valor ótimo da relaxação linear do problema mestre são iguais a zero, enquanto o valor ótimo da relaxação linear do modelo PRCA, igual a dois. No entanto, para o conjunto de instâncias geradas aleatoriamente, a diferença entre as duas relaxações, em média, não foi tão significativa e diminui com o aumento do número de culturas, provavelmente porque um número grande de culturas c ciclos distintos permitem combinações melhores.

O problema PRC–A permite outras decomposições que não foram implementadas, tais como, (a) por família botânica e (b) por família botânica e por lote. Em (a), o problema mestre é definido pelas restrições de ocupação de espaço (3.2) para os lotes e cada subproblema associado a uma programação para os $L$ lotes e única família botânica. Desta forma, a cada iteração são resolvidos $NF$ subproblemas. Na decomposição (b), o problema mestre é definido pelas restrições (3.2) de ocupação de espaço e pelas restrições (3.6) de adjacências dos lotes e são gerados $NF \times L$ subproblemas por iteração, cada um deles associado a uma rotação para um único lote e uma única família botânica.

### 5.2.3 Geração de colunas para o problema reformulado

O número de programações factíveis para um lote é usualmente muito grande e sua enumeração explícita pode ser proibitiva. Neste caso, um método de geração de colunas é usado para resolver a relaxação linear do problema mestre (PML).

No procedimento de geração de colunas, uma solução ótima para um problema mestre restrito, ou seja, um problema mestre com um número restrito de colunas, é também ótima para o problema mestre se não existe nenhuma coluna que, se inserida, melhore o valor da função objetivo do problema mestre. Para o PML, a busca por estas colunas implica em verificar a existência de, pelo menos, uma programação factível para algum dos $L$ lotes com custo reduzido positivo.

Um algoritmo de geração de colunas para o PML é descrito por:

**Algoritmo: Geração de colunas TGC (Tradicional Geração de Colunas)**

1. Escolha um conjunto de colunas para o problema mestre restrito (PMLR).
2. Resolva o PMLR e obtenha os valores duais $\pi_{kpj}$, associados às restrições de adjacência (5.2), para $k = 1..\ L$, p $= 1..\ NF$, $j = 1..\ M$, e os valores duais $\alpha_k$, da restrição de convexidade (5.3) para o lote $k$, para $k = 1..\ L$.
3. Para $k = 1..\ L$:

   (a) resolva o seguinte subproblema $P_k$:

$$z_k = \max \sum_{p=1}^{NF} \sum_{i \in F(p)} \sum_{j=1}^{M} t_i x_{ij} - \sum_{r=0}^{t_i - 1} [m_k \pi_{k,p,j-r} + \sum_{u \in A(k)} \pi_{u,p,j-r}] x_{ij}$$

   *Sujeito a:*

$$\sum_{i=1}^{N+1} \sum_{r=0}^{t_i - 1} x_{i,j-r} \leq 1,\ j = 1..M \qquad (5.5)$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i} x_{i,j-r} \leq 1,\ p = 1..NF, j = 1..M \qquad (5.6)$$

$$\sum_{i \in G} \sum_{j \in I_i} x_{ij} = 1 \qquad (5.7)$$

$$\sum_{j=1}^{M} x_{N+1,j} = 1 \qquad (5.8)$$

$$x_{ij} \in \{0,1\}, i = 1..N+1, j \in I_i \qquad (5.9)$$

   (b) calcule o calcule o custo reduzido $\hat{c}_k = z_k - \alpha_k$. Se $\hat{c}_k > 0$, uma nova coluna associada à solução de $P_k$ é gerada e inserida em PMLR.

4. Se nenhuma coluna foi adicionada ao PMLR em (3), então PARE. A solução atual é ótima para o PML. Caso contrário, volte ao passo 2.

A solução ótima de PML pode ser fracionária. Neste caso, uma opção foi tentar encontrar uma solução factível para o problema PRC–A construindo um PL 0–1 com as colunas da solução ótima do PMLR. Os testes computacionais mostraram que este procedimento nomeado **HTGC** (**H**eurística baseada em geração de colunas **TGC**) não foi eficiente, gerando soluções factíveis muito distantes da ótima. Esses resultados são discutidos no próximo capítulo.

### 5.2.4 Combinando geração de colunas com uma heurística construtiva

Para encontrar boas soluções para o problema PRC–A, uma estratégia que se mostrou bastante eficiente foi combinar a heurística *lote-a-lote* com o procedimento de geração de colunas.

Para a resolução do problema PML, em cada iteração da geração de colunas são obtidas $L$ colunas das quais aquelas com custo relativo positivo são inseridas no PMLR. Por outro lado, em cada passo da heurística *lote–a–lote* busca–se uma programação para o lote $k$, garantindo–se que as restrições de adjacência (3.6) não são violadas para todos os lotes adjacentes a $k$ e para os quais uma programação foi obtida nos passos anteriores. Ou seja, dada as programações $(\hat{x}_{ijv})$ para $(v,k) \in A$ com $v < k$, considere o subproblema $H_k$ como o subproblema $P_k$ com a fixação em zero das seguintes variáveis:

---

para $q = 1..N,\ j = 1..M$, com $(v,k) \in A$, $v < k$
se $\hat{x}_{qjv} = 1,\ q \in F(p)$, então $x_{i,j-r,k} = 0,\ \forall i \in F(p),\ r = 0..t_i - 1$

---

A combinação das metodologias é aplicada então a cada iteração da geração de colunas e consiste em resolver o problema $H_k$ para cada $k > 1$ em alternativa ao subproblema $P_k$. Caso $H_k$ não tenha solução ou o custo relativo da coluna associada à solução de $H_k$ seja negativo, o problema $P_k$ é então resolvido. Se, ao final da iteração do algoritmo, uma solução para o problema PRC–A foi obtida (ou seja, tem solução $P_1$ e também $H_k$, para $k = 2..L$), então a solução é armazenada.

O algoritmo abaixo descreve este procedimento.

---

**Algoritmo: Geração de colunas HGC (Heurística com Geração de Colunas)**

---

1. Ordene o conjunto de lotes por algum critério. Escolha um conjunto de colunas iniciais para o problema mestre restrito (PMLR). Se essas colunas estão associadas a uma solução completa para o problema PRC–A, armazene–a.

2. Resolva o PMLR e obtenha os valores duais $\pi_{kpj}$, associados às restrições de adjacência (5.2), para $k = 1..\ L$, p = 1.. $NF$, $j = 1..\ M$, e os valores duais $\alpha_k$, da restrição de convexidade (5.3) para o lote $k$, para $k = 1..\ L$.

3. Faça $sol = 0$ e para $k = 1..L$

(a) Se $sol < k$ então:
   - Resolva o subproblema $P_k$ e calcule o custo reduzido $\hat{c}_k$.
   - Se $\hat{c}_k > 0$, uma nova coluna associada à solução $\mathbf{x}_k$ de $P_k$ é gerada e inserida em PMLR. Se $k = 1$, faça $sol = 2$ e armazena $\mathbf{x}_1$.

(b) Se $sol = k$ então:
   - Resolva $H_k$, em que as variáveis fixadas em zero são definidas a partir da solução dos subproblemas armazenados $\mathbf{x}_1, \cdots, \mathbf{x}_{k-1}$.
   - Se $H_k$ tem solução, armazene a programação $\mathbf{x}_k$ e faça $sol = sol + 1$. Calcule o custo reduzido $\hat{c}_k$.
   - Se $\hat{c}_k > 0$ uma coluna associada à $\mathbf{x}_k$ é inserida em PMLR.

(c) Se $sol = L$ então:
   - As $L$ programações $\mathbf{x}_1, \cdots, \mathbf{x}_L$ formam uma solução para o problema PRC–A. Armazene–a, caso seja a melhor solução até o momento.

4. Se nenhuma coluna foi adicionada ao PMLR em (3), então PARE. A solução atual é ótima para o PML. Caso contrário, volte ao passo 2.

---

Analogamente à construção da heurística **HTGC**, com as colunas da solução ótima do PMLR construímos a heurística **HHGC** (**H**eurística baseada na geração de colunas **HGC**).

De acordo com o passo 3 do procedimento **HGC**, se o subproblema $H_k$ tem solução mas o custo da coluna associada é não–positivo, esta coluna não é adicionada ao PMLR. Em uma versão alternativa do procedimento **HGC**, a coluna é inserida, desde que o subproblema $P_k$, resolvido neste passo, retorne uma coluna com custo positivo. Neste caso, duas colunas associadas ao lote $k$ são adicionadas ao PMLR nesta iteração.

De maneira geral, o procedimento **HGC**, em qualquer das suas versões, foi bem mais eficiente que o procedimento **TGC**. Além da obtenção de soluções factíveis para o problema PRC–A, reduziu, em média, o número de colunas geradas, o número de iterações e o tempo computacional do procedimento. Os resultados obtidos com estas estratégias são discutidos na Seção 6.1.

A combinação de geração de colunas com a heurística *lote–a–lote* também foi usada em um conjunto de algoritmos *branch–and–price* e para gerar limitantes para um dos algoritmos *branch–and–cut* implementados.

### 5.2.5 *Branch–and–bound*

Para problemas PL 0–1 como é o caso da formulação para o problema PRC–A, a árvore de enumeração de um algoritmo *branch–and–bound* é usualmente binária. A cada nó da árvore está associado um PL 0–1, definido em um subconjunto da região factível do modelo, fixando–se um subconjunto de variáveis em 0 ou 1. Com um limitante superior para o nó (maximização), obtido com a relaxação linear do PL 0–1, determina se o nó deve ser ramificado ou podado. As podas na árvore ocorrem quando o PL 0–1 associado a um nó da árvore é infactível ou o limitante superior do nó é menor ou igual ao valor da melhor solução inteira já encontrada (que fornece um limitante inferior para todos os nós da árvore). Quando todos os nós foram podados, o algoritmo termina e a solução inteira de maior valor da função objetivo é a solução ótima do PL 0–1.

Devido ao tamanho que a árvore de enumeração pode ter, para muitos PLI não é possível provar a otimalidade da melhor solução factível armazenada, ou mesmo encontrar uma solução factível em um tempo computacional razoável. Dentre as estratégias usualmente consideradas para aumentar a eficiência de um algoritmo *branch–and–bound* investigamos as seguintes possibilidades: (a) escolha de uma regra de ramificação alternativa, (b) inserção de planos de cortes e (c) limitantes.

#### (a) *Regra de ramificação*

Em uma árvore binária de um PL 0–1 uma regra de ramificação usual é escolher a variável com valor mais próximo de ½ (ou de um dos extremos 0 ou 1) para ramificar. Em alguns dos algoritmos *branch–and–bound* e *branch–and–price* implementados para o PRC–A, como é necessário alocar adubação verde e pousio em cada programação, uma opção foi ramificar primeiro nas variáveis associadas a estas culturas. Caso não exista nenhuma (todas as variáveis associadas à adubação verde e pousio têm valores 0 ou 1), percorre–se o conjunto restante de variáveis. Além disto, a fim de diminuir a dimensão do PL 0–1 nos nós descendentes, no ramo em que uma variável é fixada em 1, todas as variáveis que, pelo conjunto das restrições do modelo assumiriam valor zero, são previamente fixadas neste valor.

**(b) *Desigualdades válidas***

Em um algoritmo *branch–and–cut*, planos de corte são gerados e adicionados ao longo da execução do método visando reduzir a região de factibilidade do modelo relaxado. O algoritmo *branch–and–cut* padrão do pacote CPLEX, por exemplo, utiliza diferentes tipos de planos de corte dependendo do modelo (ILOG, 2007).

Para o PRC–A, aproveitamos os dois possíveis conjuntos de restrições (3.6) e (3.8) que proíbem o plantio de culturas de mesma família em lotes adjacentes para gerar possíveis desigualdades válidas. No capítulo 3 mostramos que as restrições de adjacências (3.6) podem ser substituídas pelas restrições (3.8) com a possível redução do número de restrições e diminuição no valor da relaxação linear do modelo, ainda que um conjunto de restrições não domine o outro. Por outro lado, pode não ser eficiente inserir os dois grupos de restrições pela dimensão que o modelo pode assumir.

Assim, na tentativa de reduzir o número de restrições do modelo e ao mesmo tempo possibilitar a obtenção de melhores limitantes superiores (o que seria eventualmente possível com a inserção dos dois conjuntos de restrições), uma opção é definir o modelo com as restrições (3.8) e adicionar restrições do tipo (3.6) durante a execução do algoritmo *branch–and–cut* da seguinte forma: seja um nó da árvore em que a ramificação ocorre na variável $x_{qdk}$ referente à cultura $q \in F(p)$ no período $d$ e no lote $k$. Se $m_k > 1$, então as seguintes desigualdades são incluídas no PL 0–1 do nó:

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i - 1} \left[ x_{i,d-r,k} + x_{i,d-r,u} \right] \leq 1, (u,k) \in A$$

Como a diferença entre as relaxações lineares dos modelos com as restrições (3.8) e com as restrições (3.6) foi, em média, inferior a 1%, a inclusão destas desigualdades não trouxe benefícios significativos (ver Seção 6.1).

**(c) *Limitantes superiores e inferiores***

Uma estratégia para melhorar a eficiência de algoritmos *branch-and-bound* é iniciá-los com bons limitantes, ou seja, encontrar uma boa solução factível e um bom limitante superior (maximização). Experimentos computacionais mostraram que, com o aumento da dimensão do modelo 0–1 para o problema, o número de exemplos para os

quais nenhuma solução factível é encontrada no limite de tempo considerado aumenta drasticamente. Por outro lado, o procedimento de geração de colunas **HGC** produz tanto uma solução factível para o problema PRC–A quanto um limitante superior (no mínimo igual, mas geralmente melhor, que o obtido com a relaxação linear do modelo) em um tempo computacional bem pequeno. Desta forma, o procedimento **HGC** pode ser usado para fornecer limitantes iniciais de boa qualidade para algoritmos *branch–and–cut*.

Assim, a partir das estratégias (a)–(c) é possível criar várias combinações gerando algoritmos *branch–and–cut* distintos. Implementamos alguns deles e os resultados são discutidos na Seção 6.1.

### 5.2.6 *Branch–and–price*

Algoritmos *branch–and–price* e *branch–and–cut–price* (plano de cortes com *branch–and–price*) podem ser visto como uma generalização de métodos *branch–and–bound* em que os subconjuntos, nos quais o espaço de soluções factíveis é sucessivamente dividido, não precisam ser definidos explicitamente para que seja provada a otimalidade do PL resolvido por geração de colunas em cada nó da árvore de enumeração.

A regra de ramificação e a inserção de desigualdades válidas são questões cruciais na implementação desse tipo de algoritmo, pois podem destruir a estrutura do subproblema ou torná-lo bem mais complexo. Em cada nó $t$ da árvore do algoritmo um problema mestre (PMt) é resolvido por geração de colunas e, para isto, pode ser necessária a inclusão de novas variáveis (colunas). Se a solução ótima do PMt é fracionária, restrições de partição são adicionadas aos nós descendentes e/ou aos subproblemas e isto exige compatibilidade entre o subproblema e a regra de ramificação.

Quando um PLI é reformulado e as restrições correspondentes às ramificações nos nós da árvore de enumeração são nas variáveis originais e mantidas no problema mestre, as modificações nos subproblemas podem ser apenas na função objetivo. Analogamente se são inseridas desigualdades válidas expressas nas variáveis originais. Para um PL 0–1 é possível manter as decisões de ramificação restritas aos subproblemas, sem modificar o problema mestre. No ramo em que a variável original

é fixada em zero (um) todas as colunas em que esta variável é um (zero) são removidas. Além disto, a variável é fixada em zero (um) nos subproblemas, o que garante que as novas colunas geradas para os nós descendentes têm as mesmas características.

Para um PLI resolvido por *branch–and–price*, além de ramificação nas variáveis originais, outras regras para tipos específicos de problema (envolvendo ou não as variáveis originais) foram desenvolvidas (Barnhart et al., 1998; Vanderbeck, 2000). Vanderbeck (2008) apresenta um esquema genérico de uma regra de ramificação que generaliza a regra de Ryan e Foster para o problema de particionamento de conjuntos. Essas e outras questões pertinentes à implementação de um algoritmo *branch–and(–cut)–price* são discutidas em Barnhart et al. (1998), Lübbecke e Desrosiers (2005) e Vanderbeck (2005).

Na Seção 5.2.2 o modelo para o PRC–A foi reformulado e, na seção subsequente, algoritmos de geração de colunas foram propostos para resolver a relaxação linear do problema mestre. A seguir discutimos algumas questões referentes aos algoritmos *branch–and–price* implementados.

***Solução inicial para um nó da árvore***

Em cada nó da árvore *branch–and–price* um problema mestre restrito inicial com garantia de que sua relaxação linear seja factível deve estar disponível a fim de assegurar que a informação dual seja passada aos subproblemas. Para isto, uma opção foi manter um conjunto de colunas (variáveis) artificiais em todos os nós da árvore. Outra estratégia foi usar a heurística *lote–a–lote–completa* para gerar colunas para um problema mestre, caso a remoção de colunas referentes à ramificação do nó, o tornem infactível. Se, com a fixação de todas as variáveis impostas pela ramificação, a heurística retorna uma solução, colunas associadas são inseridas no problema mestre, tornando–o factível. Caso contrário, o nó é podado.

***Regra de ramificação***

As ramificações implementadas foram a padrão (a ramificação ocorre na variável com valor mais próximo de ½) e a descrita no item (a) da Seção 5.2.5.

***Seleção do próximo nó***

Foram implementadas as estratégias de busca em profundidade, largura e a de melhor limitante.

***Limitante para o problema mestre***

Um limitante da função objetivo do problema mestre pode ser útil para finalizar o algoritmo de geração de colunas ou para podar um nó, caso a solução inteira armazenada seja maior ou igual à parte inteira desse limitante superior. Lasdon (1970) e Farley (1990), por exemplo, descrevem limitantes para o valor ótimo do problema mestre. O limitante dado em Lasdon (1970) foi utilizado nos procedimentos baseados em geração de colunas que foram implementados.

***Desigualdades válidas***

As desigualdades válidas discutidas no item (b) da Seção 5.2.5 foram inseridas em um dos algoritmos *branch–and–price*. Para cada variável escolhida para ramificar a desigualdade associada àquela variável foi inserida no problema mestre de todos os nós ativos.

***Geração de colunas***

Os procedimentos de geração de colunas propostos nas subseções 5.2.3 e 5.2.4 foram usados em diferentes algoritmos *branch–and–price* para resolver a relaxação linear do PML 0–1 em cada nó da árvore.

## 5.3 Métodos exatos e heurísticos para o problema PRC-D

A formulação (4.1)-(4.4) proposta para o problema PRC-D na Seção 4.1 pode ser vista como um problema mestre no contexto de geração de colunas, pois é um problema de otimização linear com um número grande de variáveis, cada uma associada a uma programação de produção que pode ser obtida com a resolução de um subproblema. Portanto, colunas podem ser adicionadas ao problema buscando–se por programações de rotações que maximizam o custo relativo. Obtendo–se os valores

duais $\pi_{ij}$ de cada uma das restrições de adjacências (4.2) e os valores duais $\alpha_k$ das restrições de convexidade (4.3), o maior custo relativo de uma coluna (programação de produção) para a área $k$ é descrito por:

$$\hat{c}_k = \max \{c_{ks} - \pi \mathbf{a}_k^s, s \in S_k\} - \alpha_k$$

Como uma programação de produção é completamente definida por uma programação de rotação, podemos escrever os $L$ subproblemas em função de programações de rotação de culturas. Considere o conjunto $X_k$ de todas as programações de rotação factíveis para a área $k$, ou seja, satisfazendo (4.5)–(4.9).

Um algoritmo de geração de colunas para o modelo PRCD é então dado por:

---

**Algoritmo: Geração de colunas para o modelo PRCD**

---

1. Escolha um conjunto de colunas iniciais para o problema mestre restrito (PMR).

2. Resolva o PMR e obtenha $\pi$ e $\alpha$, respectivamente, os conjuntos de variáveis duais associadas às restrições (4.2) e (4.3).

3. Para cada $k = 1..L$, calcule o custo reduzido $\hat{c}_k = z_k - \alpha_k$, resolvendo o subproblema $P_k$

   $$z_k = \max \{c(\mathbf{x}) - \pi a(\mathbf{x}), \mathbf{x} \in X_k\}$$

   e obtenha uma coluna $\mathbf{a}_k = a(\mathbf{x}_k)$ com custo $c_k = c(\mathbf{a}_k)$.

4. Se, para $k = 1..L$, $\hat{c}_k = 0$, então PARE. A solução atual é ótima. Caso contrário, para $\hat{c}_k > 0$ insira a coluna $\mathbf{a}_k$ em PMR e volte em 2.

---

Ao final do algoritmo de geração de colunas tem–se um conjunto de programações de produção ($\mathbf{a}_k^s$) associadas aos tamanhos dos lotes ($\hat{\lambda}_{ks}$). Nesta solução pode existir um número grande de valores $\hat{\lambda}_{ks} > 0$, alguns dos quais considerados pequenos (por algum critério de interesse), o que pode não ser desejável. A seguir são propostas algumas estratégias para lidar com esta situação.

Dada uma solução para o modelo (4.1)–(4.4), uma opção simples para reduzir o número de $\hat{\lambda}_{ks} > 0$ é fixar em zero todos os tamanhos $\hat{\lambda}_{ks} < \alpha$, em que $\alpha$ é um valor mínimo desejável para o tamanho de um lote. Nomeamos esta estratégia trivial de *α–lote*. Outra opção é usar a heurística *redução–lote* que determina um conjunto mínimo de colunas no PMR ótimo para atender a demanda resolvendo–se o seguinte problema inteiro–misto:

$$(PRCD\text{-}min) \quad z_{PRCD-\min} = \sum_{k=1}^{L} \sum_{s \in \hat{S}_k} \delta_{ks}$$

$$s.a. \quad \sum_{k=1}^{L} \sum_{s \in \hat{S}_k} \mathbf{a}_k^s \lambda_{ks} \geq \mathbf{d}$$

$$\sum_{s \in S_k} \lambda_{ks} \leq A_k, k = 1..L$$

$$\lambda_{ks} \leq A_k \delta_{ks}, k = 1..L$$

$$\delta_{ks} = \{0,1\}, \ \lambda_{ks} \geq 0, \ s \in \hat{S}_k, k = 1..L$$

em que $\mathbf{a}_k^s, \ s \in \hat{S}_k, k = 1..L$ são as colunas do PMR ótimo.

Os procedimentos *α–lote* e *redução–lote* levam a um eventual aumento de áreas não ocupadas com nenhuma programação. Estratégias *residuais* para o uso das áreas ainda disponíveis (e para redução da demanda não atendida) podem ser desenvolvidas. Considere, por exemplo, $\hat{\lambda}_{ks}$ e $\mathbf{a}_k^s$, para $s \in \hat{S}_k$ e $k = 1..\ L$, no PMR ótimo. Definimos o resíduo absoluto ($r_{ij}$) e o resíduo relativo ($rr_{ij}$) de uma cultura $i$ no período $j$ por:

$$r_{ij} = d_{ij} - \sum_{k=1}^{L} \sum_{s \in \hat{S}_k} a_{ijk}^s \hat{\lambda}_{ks}$$

$$rr_{ij} = \frac{r_{ij}}{d_{ij}}$$

A área eventualmente disponível após a aplicação das heurísticas pode ser utilizada para aumentar a área de alguns lotes, na tentativa de diminuir o não atendimento da demanda. A prioridade pode ser culturas com os maiores resíduos (absoluto ou relativo). Outra opção é resolver uma versão do modelo (4.1)–(4.4) no qual os valores das demandas, $d_{ij}$, são substituídos pelos seus resíduos, $r_{ij}$. Este procedimento pode ser usado recursivamente até não existir área residual disponível ou seja alcançado outro critério de parada apropriado.

Na Seção 6.2 são analisados os resultados obtidos com o algoritmo de geração de colunas e com as heurísticas de pós–otimização, *α–lote* e *redução–lote*. Para estas, os resultados são apresentados em termos de pior caso, em que procedimentos residuais não foram utilizados.

# Capítulo 6

# Experimentos Computacionais

No Capítulo 5 foram propostos procedimentos heurísticos e exatos para os problemas PRC–A e PRC–D. Neste capítulo analisamos os resultados computacionais obtidos com as implementações dos procedimentos. Todos os algoritmos foram implementados em linguagem C++ e os modelos lineares inteiros foram resolvidos com o pacote de otimização CPEX 11.1. Os testes computacionais foram realizados em um PC Intel Core 2 Duo 2.33 GHZ com 4 GB de RAM.

## 6.1 Resultados computacionais para o problema PRC-A

Para análise dos modelos matemáticos e das metodologias de solução propostos para o problema PRC–A, apresentados, respectivamente, nas Seções 3.1 e 5.2, testes computacionais foram realizados com instâncias geradas aleatoriamente. Para essas instâncias, o tamanho da rotação foi fixado em 2 anos, dividido em períodos de 10 dias, e o pousio obrigatório em cada rotação, de 1 mês (3 períodos). Como não existem instâncias na literatura para este tipo de problema, criamos dois geradores aleatórios baseados em dados reais: um para o conjunto de culturas e outro para a configuração espacial dos lotes de plantio.

Em relação ao número de culturas, as instâncias dividem–se em 3 grupos com $N = 6$, 16 e 26. No primeiro grupo, o número de culturas para adubação verde é 1, no segundo de 2 a 3 e, no terceiro, de 3 a 5. Cada grupo de culturas é constituído por 10 exemplares e os dados para as 28 culturas usadas para gerar os exemplares foram obtidos em uma horta de base ecológica situada em Viçosa–MG e são exibidos na Tabela A.1 do Apêndice A. Para a configuração espacial da área de plantio, as instâncias dividem–se em 5 grupos com $L = 4$, 8, 12, 16 e 20 vértices (lotes). Um grupo de lotes é composto de 4 grafos com densidades variadas. No grupo com 8

lotes, por exemplo, a densidade variou de 0.2 a 0.6. O limite superior foi determinado pela condição de planaridade do grafo.

Com a combinação dos conjuntos de culturas ($N = 6$, 16 e 26) e configuração espacial dos lotes ($L = 4$, 8, 12, 16 e 20) foram geradas 15 classes cada uma com 40 instâncias, atingindo um total de 600 instâncias testadas para cada algoritmo. Dentre as classes, as com $N = 6$ culturas e $L \geq 12$ lotes são mais hipotéticas pois, em situações reais, o número de lotes é comumente menor que 10 e o número de culturas em hortas de base ecológica usualmente é maior que uma dezena.

Como a solução inteira ótima não é conhecida para todas as instâncias, o *gap* do algoritmo foi calculado por:

$$gap = 100 \frac{limsup - solint}{|solint| + 10^{-10}}$$

em que *limsup* e *solint* são, respectivamente, os limitantes superior e inferior obtidos com o procedimento.

Para todos os experimentos, o tempo computacional foi limitado em 30 minutos. A escolha deste limitante foi devido ao grande número de algoritmos que seriam executados para cada conjunto de instâncias. Além disto, testes computacionais preliminares mostraram que, dobrá–lo, por exemplo, não alterariam a comparação do desempenho dos algoritmos.

Com o aumento do número de lotes, aumentou a porcentagem de instâncias para os quais alguns dos algoritmos não conseguiram solução factível no limite de tempo estabelecido. Em outra situação, quando uma solução inicial formada apenas por adubação verde e pousio foi fornecida, certos algoritmos não conseguiram nenhuma solução de melhor qualidade. Para estas duas situações, essas instâncias foram excluídas no cálculo do *gap* médio do algoritmo e a porcentagem de ocorrência mencionada. Com esta estratégia de cálculo, eventualmente o *gap* de determinados algoritmos pode ser muito pequeno com uma porcentagem de instâncias eliminadas muito grande.

Alguns resultados para os procedimentos descritos nas Subseções 5.2.3 a 5.2.6 são apresentados e discutidos a seguir.

### 6.1.1 Heurísticas baseadas em geração de colunas

Nas Subseções 5.2.3 e 5.2.4 foram descritas as heurísticas **HTGC** e **HHGC** nas quais, após o procedimento de geração de colunas (respectivamente **TGC** e **HCG**), um PL 0–1 construído com as colunas da solução ótima do problema mestre restrito (PMLR) é resolvido a fim de se encontrar uma solução inteira (nomeada *solBB*) para o PRC–A. Para os testes computacionais, esse PMLR 0–1 é resolvido usando o algoritmo *branch–and–cut* padrão do *software* CPLEX limitado em 30 minutos.

Para avaliar o desempenho das heurísticas **HTGC** e **HHGC** e a influência da solução inicial, dois conjuntos de testes foram realizados. No primeiro, a solução inicial é formada apenas com a alocação de adubação verde e pousio em cada lote e foi nomeada *sol0*. No segundo, uma solução inicial *solini* é obtida com a heurística *lote–a–lote–completa*. As Tabelas 6.1 a 6.4 exibem resultados para o primeiro conjunto de testes e, as Tabelas 6.5 a 6.8, para o segundo.

A Tabela 6.1 refere–se à heurística **HTGC** e exibe os valores médios do tempo (em segundos) de execução do procedimento de geração de colunas (*tempoGC*), do número de colunas geradas (*#col*), do número de iterações da geração de colunas (*#it*), do tempo de execução do PMLR 0–1 (*tempoBB*) e do *gap* do procedimento (*gap*).

Tabela 6.1: Resultado para **HTGC** com *sol0*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***gap*** |
|---|---|---|---|---|---|---|
| | **6** | 278,08 | 433,30 | 113,80 | 79,97 | 32,14 |
| **4** | **16** | 7,13 | 89,40 | 24,40 | 0,56 | 40,69 |
| | **26** | 9,25 | 82,50 | 22,60 | 0,37 | 26,79 |
| | **6** | 180,89 | 342,40 | 51,60 | 370,33 | 58,34 |
| **8** | **16** | 8,41 | 98,10 | 15,00 | 44,61 | 28,86 |
| | **26** | 11,48 | 92,70 | 14,30 | 36,72 | 46,30 |
| | **6** | 60,65 | 259,00 | 24,40 | 203,09 | 75,27 |
| **12** | **16** | 9,06 | 88,40 | 9,20 | 8,95 | 59,00 |
| | **26** | 12,38 | 85,30 | 8,80 | 14,52 | 38,14 |
| | **6** | 157,10 | 392,80 | 34,30 | 381,98 | 53,16 |
| **16** | **16** | 17,91 | 129,50 | 10,90 | 8,50 | 57,03 |
| | **26** | 23,89 | 122,10 | 10,30 | 9,50 | 54,62 |
| | **6** | 207,73 | 488,40 | 34,70 | 857,08 | 62,67 |
| **20** | **16** | 20,63 | 148,20 | 10,00 | 24,64 | 57,84 |
| | **26** | 26,89 | 140,10 | 9,20 | 36,42 | 58,26 |

Como pode ser visto na tabela, para classes de culturas com $N = 6$ são significativamente maiores os valores médios do tempo de execução, número de colunas e de iterações da geração de colunas bem como do tempo de resolução do PMLR 0–1. Além disto, com o aumento do número de lotes, o tempo de resolução do PMLR 0–1, em geral, aumentou.

No final do procedimento de geração de colunas **HGC**, diferentemente de **TGC**, sem resolver o PMLR 0–1 geralmente se tem disponível uma solução factível (*solbest*) melhor que *sol0*. Por outro lado, nem todas as colunas associadas com as eventuais soluções factíveis para o problema PRC–A geradas durante a geração de colunas **HGC** são inseridas no PMLR. Em função disto, nos resultados apresentados para a heurística **HHGC**, são exibidos os valores médios para o *gap* do procedimento de geração de colunas (*gapGC*), *gap* do PMLR 0–1 (*gapBB*) e o *gap* calculado com a melhor solução obtida com a heurística **HHGC**, ou seja, a melhor solução entre *solbest* e *solBB*.

A Tabela 6.2 exibe os valores médios para *tempoGC*, *#col, #it* e *tempoBB, gapGC*, *gapBB* e *gap* para a heurística **HHGC**. Para o cálculo do *gapGC* de cada classe foram excluídas as instâncias para as quais *solbest* = *sol0*.

Tabela 6.2: Resultado para **HHCG** com *sol0*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***gapGC*** | ***gapBB*** | ***gap*** |
|---|---|---|---|---|---|---|---|---|
| | **6** | 177,55 | 252,70 | 65,90 | 33,24 | 3,48 | 8,42 | **3,48** |
| **4** | **16** | 3,02 | 27,70 | 8,30 | 0,02 | 0,13 | 0,17 | **0,13** |
| | **26** | 3,51 | 22,70 | 7,20 | 0,00 | 0,00 | 0,00 | **0,00** |
| | **6** | 133,26 | 254,40 | 35,00 | 85,15 | 5,09 | 12,97 | **5,09** |
| **8** | **16** | 4,93 | 37,80 | 6,50 | 0,09 | 0,27 | 0,57 | **0,27** |
| | **26** | 7,14 | 45,90 | 7,50 | 0,00 | 0,00 | 0,00 | **0,00** |
| | **6** | 61,36 | 266,00 | 24,60 | 103,92 | 5,49 | 14,70 | **5,49** |
| **12** | **16** | 8,24 | 51,80 | 6,80 | 0,00 | 0,00 | 0,00 | **0,00** |
| | **26** | 10,82 | 52,70 | 6,70 | 0,07 | 0,22 | 0,22 | **0,22** |
| | **6** | 87,60 | 371,90 | 30,40 | 196,68 | 4,75 | 20,36 | **4,75** |
| **16** | **16** | 11,67 | 66,50 | 6,30 | 0,44 | 0,23 | 1,79 | **0,23** |
| | **26** | 15,53 | 67,60 | 6,30 | 0,00 | 0,00 | 0,00 | **0,00** |
| | **6** | 92,93 | 419,20 | 27,60 | 270,32 | 9,27* | 31,81 | **23,78** |
| **20** | **16** | 16,40 | 85,10 | 6,80 | 0,63 | 0,37 | 0,32 | **0,32** |
| | **26** | 24,69 | 112,30 | 8,80 | 20,12 | 0,88 | 5,83 | **0,88** |

(*) **HGC** não encontrou solução melhor que *sol0* para 2 instâncias.

Para a heurística **HHGC**, os parâmetros $L$ e $N$ influenciaram as medidas avaliadas de forma similar ao ocorrido com a heurística **HTGC**. A estratégia de se obter soluções factíveis para o PRC–A durante o processo de geração de colunas mostrou–se, em geral, mais eficiente que resolver o PMLR 0–1, gerando soluções de melhor qualidade. A média geral do *gapGC* foi 2,01% e do *gapB,* 6,48%. Por outro lado, para 0,3% das instâncias o procedimento **HGC** não foi capaz de encontrar uma solução melhor que *sol0* e, somente para estas instâncias, *solBB* foi melhor que *solbest*. Esta situação ocorreu apenas em uma classe com $N = 6$ e $L = 20$.

Em comparação com a heurística **HTGC**, a heurística **HHGC** mostrou–se mais eficiente, confirmando que as colunas adicionais, além de cumprirem seu objetivo inicial de redução do *gap* (a média geral com a **HTGC** foi de 49,94% e, com a **HHGC**, 2,97%), diminuíram, em geral, o valor médio do número de colunas geradas (199,48 da primeira e 142,29 da segunda), número de iterações (26,23 e 16,98, respectivamente), tempo de execução da geração de colunas (68,93 na primeira e 43,91, na segunda) e o tempo total de execução da heurística (207,16 e 91,29, respectivamente).

Como mencionado na Subseção 5.2.4, no procedimento **HGC** as colunas construídas com soluções do problema $H_k$ (relativo à heurística *lote–a–lote*) e com custo relativo não–positivo não foram inseridas no PMLR. Além disto, quando a solução do problema $H_k$ forneceu uma coluna com custo positivo, o subproblema $P_k$ não foi resolvido. Como alternativa a esta estratégia, foram gerados dois outros algoritmos **HGCa** e **HGCb** nos quais as colunas advindas de $H_k$ são inseridas, independentemente do seu custo relativo. Para a heurística **HGCb**, além disto, o problema $P_k$ é sempre resolvido.

As Tabelas 6.3 e 6.4 referem–se, respectivamente, às heurísticas **HHGCa** e **HHGCb** e exibem os valores médios do tempo de execução (*tempoGC*), número de colunas (*#col*), *número de iterações (#it*), tempo de resolução do PMLR 0–1 (*tempoBB*), *gap* da geração de colunas (*gapGC*) e o *gap* da heurística. Como nas duas heurísticas o *gap* obtido com o PMLR 0–1 (*gapBB)* coincide com o *gap* da heurística (*gap*), os resultados relativos a este *gap* foram omitidos. Em ambas foi possível encontrar uma solução melhor que *sol0* para todas as instâncias.

Tabela 6.3: Resultado para **HHGCa** com *sol0*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***gapGC*** | ***gap*** |
|---|---|---|---|---|---|---|---|
| | **6** | 314,02 | 190,50 | 68,10 | 41,70 | 2,78 | **2,78** |
| **4** | **16** | 3,38 | 14,40 | 8,90 | 0,01 | 0,08 | **0,08** |
| | **26** | 3,48 | 11,30 | 6,80 | 0,00 | 0,00 | **0,00** |
| | **6** | 199,19 | 196,20 | 41,80 | 158,42 | 4,52 | **4,50** |
| **8** | **16** | 4,42 | 12,70 | 6,00 | 0,11 | 0,27 | **0,27** |
| | **26** | 6,42 | 16,20 | 6,50 | 0,04 | 0,17 | **0,17** |
| | **6** | 51,61 | 165,70 | 23,90 | 182,48 | 4,88 | **4,88** |
| **12** | **16** | 6,76 | 10,90 | 5,40 | 0,23 | 0,18 | **0,18** |
| | **26** | 9,85 | 13,40 | 6,00 | 0,13 | 0,32 | **0,28** |
| | **6** | 92,40 | 270,60 | 33,90 | 260,72 | 4,06 | **4,00** |
| **16** | **16** | 12,63 | 25,30 | 6,20 | 0,13 | 0,04 | **0,04** |
| | **26** | 22,85 | 26,10 | 6,40 | 76,75 | 0,52 | **0,52** |
| | **6** | 74,20 | 183,90 | 19,80 | 405,74 | 4,76 | **4,76** |
| **20** | **16** | 16,21 | 19,60 | 5,70 | 0,12 | 0,03 | **0,03** |
| | **26** | 72,40 | 183,70 | 20,10 | 396,18 | 4,74 | **4,74** |

Tabela 6.4: Resultado para **HHGCb** com *sol0*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***gapGC*** | ***gap*** |
|---|---|---|---|---|---|---|---|
| | **6** | 315,70 | 247,30 | 67,60 | 44,19 | 3,18 | **3,18** |
| **4** | **16** | 3,47 | 23,70 | 7,10 | 0,04 | 0,13 | **0,13** |
| | **26** | 4,30 | 18,70 | 6,10 | 0,00 | 0,21 | **0,21** |
| | **6** | 232,89 | 238,20 | 35,80 | 164,89 | 6,17 | **6,10** |
| **8** | **16** | 5,67 | 32,50 | 5,70 | 0,02 | 0,15 | **0,15** |
| | **26** | 7,54 | 32,80 | 5,70 | 0,07 | 0,27 | **0,27** |
| | **6** | 67,44 | 215,60 | 21,80 | 259,37 | 5,89 | **5,89** |
| **12** | **16** | 7,86 | 37,70 | 4,80 | 0,40 | 0,20 | **0,20** |
| | **26** | 11,85 | 46,20 | 5,50 | 0,26 | 0,15 | **0,15** |
| | **6** | 115,42 | 338,30 | 29,40 | 464,72 | 4,68 | **4,65** |
| **16** | **16** | 13,12 | 58,20 | 5,30 | 0,89 | 0,29 | **0,27** |
| | **26** | 16,52 | 53,50 | 4,90 | 0,29 | 0,13 | **0,13** |
| | **6** | 142,93 | 409,40 | 31,60 | 1348,64 | 8,67 | **8,67** |
| **20** | **16** | 14,64 | 63,60 | 4,60 | 0,52 | 0,37 | **0,32** |
| | **26** | 21,78 | 70,60 | 5,60 | 0,44 | 0,18 | **0,18** |

Na média geral, a heurística **HHGC** apresentou um tempo total de execução inferior (91,29 segundos) ao das heurísticas **HGCa** e **HHGCb** (respectivamente, 160,94 e 217,72 segundos). Tanto o valor médio do *gap* quanto do número de colunas foram menores com a heurística **HHGCa**. Na heurística **HHGC**, o *gap* foi de 2,97%, na **HHGCa**, 1,82% e, na **HHGCb**, 2,03%. O número médio de colunas nas heurísticas foram 142,29, 83,37 e 125,75, respectivamente. O menor valor médio do número de iterações foi alcançado pela heurística **HHGCb**, com 16,10. Nas heurísticas **HHGC** e **HHGCa**, os valores médios foram, respectivamente, 16,98 e 17,70.

Portanto, usando *sol0* como solução inicial, as heurísticas **HHGC**, **HHGCa** e **HHGCb** não diferem muito entre si e tiveram um desempenho bem superior à heurística **HTGC** em todas as medidas avaliadas. Nas 4 heurísticas, com exceção do tempo para se resolver o PMLR 0–1, nenhuma das medidas aumentou significativamente com o número de lotes.

Para o segundo conjunto de testes comparamos somente as heurísticas **HTGC** e **HHGC**. A solução inicial *solini* é obtida com a heurística *lote–a–lote–completa*. As Tabelas 6.5 e 6.6 exibem resultados para esses procedimentos. A Tabela 6.5 mostra o valor médio do *tempoGC*, *#col, #it, tempoBB*, da porcentagem de instâncias para as quais a otimalidade da solução foi provada (*%opt*) e do *gap* para a heurística **HTGC**.

Tabela 6.5: Resultado para **HTGC** com *solini*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***%opt*** | ***gap*** |
|---|---|---|---|---|---|---|---|
| | **6** | 263,32 | 443,20 | 114,60 | 84,42 | 0,00 | **15,09** |
| **4** | **16** | 6,17 | 77,40 | 21,00 | 0,28 | 20,00 | **3,21** |
| | **26** | 6,59 | 56,70 | 15,70 | 0,08 | 20,00 | **1,10** |
| | **6** | 200,50 | 307,60 | 43,20 | 132,83 | 0,00 | **24,95** |
| **8** | **16** | 6,68 | 68,90 | 10,10 | 2,14 | 0,00 | **4,78** |
| | **26** | 8,61 | 62,30 | 9,10 | 1,60 | 10,00 | **1,64** |
| | **6** | 69,88 | 250,60 | 23,40 | 219,20 | 0,00 | **19,16** |
| **12** | **16** | 9,60 | 77,20 | 7,90 | 0,88 | 10,00 | **3,54** |
| | **26** | 10,78 | 59,70 | 6,30 | 0,81 | 10,00 | **1,39** |
| | **6** | 86,09 | 239,20 | 20,10 | 14,75 | 0,00 | **9,51** |
| **16** | **16** | 21,28 | 99,70 | 8,50 | 0,74 | 0,00 | **2,01** |
| | **26** | 75,43 | 253,60 | 20,20 | 283,06 | 0,00 | **11,20** |
| | **6** | 148,65 | 467,10 | 29,50 | 252,13 | 0,00 | **20,58** |
| **20** | **16** | 23,35 | 124,10 | 9,00 | 1,98 | 0,00 | **4,51** |
| | **26** | 24,96 | 105,20 | 7,00 | 0,71 | 0,00 | **1,46** |

Com a *solini*, houve um pequeno decréscimo no valor médio do número de colunas e de iterações do procedimento **HTCG** passando, respectivamente, de 199,48 para 179,50 e de 26,23 para 23,04. O valor médio do tempo computacional aumentou de 207,25 para 440,00 segundos e do *gap* diminuiu de 49,94% para 8,27%. Vale notar que, usando o limitante superior obtido com o procedimento de geração de colunas, o valor médio do *gap* da heurística *lote–a–lote–completa* é 9,26%. A porcentagem de instâncias em que foi possível provar a otimalidade da solução foi pequena (6%) e ocorreu em classes com $N = 16$ e $N = 26$ com $L \leq 12$ lotes.

A Tabela 6.6 exibe os valores médios do *tempoGC*, *#col*, *#it*, *tempoBB*, *%opt*, *gapGC*, *gapBB* e do *gap* para a heurística **HHGC**.

Tabela 6.6: Resultado para **HHGC** com *solini*

| ***L*** | ***N*** | ***tempoGC*** | ***#col*** | ***#it*** | ***tempoBB*** | ***%opt*** | ***gapGC*** | ***gapBB*** | ***gap*** |
|---|---|---|---|---|---|---|---|---|---|
| | **6** | 214,92 | 257,50 | 67,00 | 40,29 | 0,00 | 3,62 | 7,55 | **3,52** |
| **4** | **16** | 3,08 | 26,10 | 7,80 | 0,01 | 80,00 | 0,17 | 0,21 | **0,17** |
| | **26** | 2,61 | 13,60 | 4,70 | 0,00 | 100,00 | 0,00 | 0,00 | **0,00** |
| | **6** | 121,83 | 263,70 | 36,80 | 101,04 | 0,00 | 4,58 | 13,59 | **4,58** |
| **8** | **16** | 4,87 | 40,10 | 6,50 | 0,12 | 60,00 | 0,33 | 0,63 | **0,33** |
| | **26** | 5,48 | 24,60 | 5,00 | 0,00 | 100,00 | 0,00 | 0,00 | **0,00** |
| | **6** | 59,89 | 269,80 | 24,80 | 207,57 | 0,00 | 5,07 | 10,78 | **5,03** |
| **12** | **16** | 7,13 | 45,10 | 5,90 | 0,05 | 80,00 | 0,06 | 0,07 | **0,06** |
| | **26** | 8,79 | 37,40 | 5,00 | 0,03 | 90,00 | 0,07 | 0,18 | **0,07** |
| | **6** | 76,94 | 359,00 | 28,60 | 171,41 | 0,00 | 3,82 | 12,38 | **3,82** |
| **16** | **16** | 12,24 | 66,30 | 6,40 | 0,08 | 70,00 | 0,08 | 0,26 | **0,08** |
| | **26** | 14,17 | 51,50 | 5,30 | 0,10 | 80,00 | 0,07 | 0,21 | **0,07** |
| | **6** | 110,54 | 458,40 | 29,60 | 487,82 | 0,00 | 10,31 | 18,11 | **10,27** |
| **20** | **16** | 16,99 | 104,00 | 7,20 | 0,75 | 40,00 | 0,21 | 0,65 | **0,21** |
| | **26** | 17,56 | 71,20 | 5,40 | 0,07 | 80,00 | 0,17 | 0,17 | **0,17** |

Para a heurística **HHGC** com *solini* (em relação à *sol0*) houve uma pequena redução, em média, de quase todas as medidas observadas. O valor médio do número de colunas passou de 142,29 para 139,22, número de iterações de 16,98 para 16,40 e *gap* de 2,97% para 1,89%. O valor médio do tempo total de execução do algoritmo, no entanto, aumentou de 91,29 para 112,42 segundos. Para 55,33% das instâncias foi possível provar a otimalidade da solução. Nenhuma destas ocorrências em classes com $N = 6$ culturas.

Para 93,33% das instâncias, resolver o PMLR 0–1 não resultou em melhoria da solução armazenada e, para as instâncias para as quais a otimalidade foi provada, isto ocorreu sem resolver o PLML 0–1. Vale notar que, no mínimo, $solbest = solini$ e, no PMLR 0–1 estão também inseridas as colunas associadas à *solini*.

Ainda que as heurísticas **HHGC** e variantes tenham se mostrado eficientes, a otimalidade não pode ser provada com estes algoritmos para todas as instâncias, pois existem aquelas para as quais o valor ótimo da função objetivo do problema PRC–A é menor que o valor ótimo da relaxação linear do problema mestre do problema reformulado. Para este fim, os métodos exatos *branch–and–cut* e *branch–price* foram implementados. A próxima subseção discute alguns resultados obtidos com os algoritmos *branch–and–cut* implementados e, na subseção 6.1.3, com os algoritmos *branch–and–price*.

### 6.1.2 *Branch–and–cut*

Nesta subseção investigamos se específicos (a) regra de ramificação, (b) desigualdades válidas e (c) limitantes inferiores e superiores afetam o desempenho do algoritmo *branch–and–cut* na resolução do modelo 0–1 para o problema PRC–A. Para isto, usando combinações de (a)–(c), definimos 4 programas.

**Regra**

Neste programa, a regra de ramificação padrão do algoritmo *branch–and–cut* do CPLEX é substituída pela regra proposta em (a) na Subseção 5.2.5. Todos os outros parâmetros padrões do *software* CPLEX são mantidos.

**Regra–corte**

Neste programa, além da ramificação, desigualdades válidas discutidas na Subseção 5.2.5, são inseridas ao longo da árvore de enumeração. Os outros parâmetros do CPLEX não foram alterados.

**Regra–solini**

A solução factível obtida com a *heurística lote–a–lote–completa* é fornecida ao algoritmo *branch–and–cut* como solução inicial. Além disto, a regra de ramificação é o outro parâmetro alterado no algoritmo.

**Regra–GC**

O procedimento **HGC** com *solini* fornece limitantes superiores e inferiores para o problema PRC–A. Estes são inseridos no início do algoritmo *branch–and–cut*. A regra de ramificação é também modificada.

Além destes 4 algoritmos, definimos o algoritmo *branch–and–cut* **Padrão** em que todos os parâmetros padrões do CPLEX foram mantidos.

As Tabelas 6.7 e 6.8 exibem resultados para os algoritmos **Padrão**, **Regra** e **Regra–corte** e mostram os valores médios, em segundos, do tempo de execução do algoritmo (*tempo*), do número de nós na árvore de enumeração (#*nós*), da porcentagem de instâncias em que otimalidade foi provada em 30 minutos (%*opt*), da porcentagem de instâncias em que uma solução factível é obtida no limite de tempo estabelecido (%*fac*) e do *gap* do procedimento. As instâncias para as quais não foi possível obter uma solução factível foram excluídas do cálculo do *gap*.

Tabela 6.7: Resultados para o algoritmo **Padrão**.

| | | **Padrão** | | | | |
|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#nós*** | ***%opt*** | ***%fac*** | ***gap*** |
| | **6** | 1019,69 | 20414,80 | 30,00 | 100,00 | 0,81 |
| **4** | **16** | 291,60 | 1996,50 | 40,00 | 100,00 | 0,04 |
| | **26** | 86,87 | 39,00 | 50,00 | 100,00 | 0,00 |
| | **6** | 1800,67 | 649,00 | 0,00 | 100,00 | 4,20 |
| **8** | **16** | 1800,81 | 1,80 | 0,00 | 100,00 | 3,49 |
| | **26** | 1800,58 | 0,00 | 0,00 | 100,00 | 8,52 |
| | **6** | 1800,89 | 79,30 | 0,00 | 100,00 | 11,26 |
| **12** | **16** | 1800,14 | 0,00 | 0,00 | 100,00 | 17,55 |
| | **26** | 1800,71 | 0,00 | 0,00 | 100,00 | 11,57 |
| | **6** | 1800,84 | 11,27 | 0,00 | 90,91 | 18,83 |
| **16** | **16** | 1800,60 | 0,00 | 0,00 | 44,44 | 18,58 |
| | **26** | 1800,26 | 0,00 | 0,00 | 50,00 | 9,59 |
| | **6** | 1800,61 | 0,00 | 0,00 | 20,00 | 30,82 |
| **20** | **16** | 1800,42 | 0,00 | 0,00 | 0,00 | – |
| | **26** | 1800,21 | 0,00 | 0,00 | 0,00 | – |

Para o algoritmo **Padrão**, de maneira geral, as medidas observadas crescem com o número de lotes. Somente para classes com $L = 4$ a otimalidade foi provada para uma média de 40% das instâncias. Para 33,7% das instâncias com $L = 16$ e 93,6% das instâncias com $L = 20$, nenhuma solução factível foi obtida em 30 minutos.

Tabela 6.8: Resultados para os algoritmos **Regra e Regra–corte**

| | | **Regra** | | | | | **Regra–corte** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#nós*** | ***%opt*** | ***%fac*** | ***gap*** | ***tempo*** | ***#nós*** | ***%opt*** | ***%fac*** | ***gap*** |
| | **6** | 1440,14 | 26560,80 | 10,00 | 100,00 | 1,71 | 1440,18 | 26335,20 | 10,00 | 100,00 | 1,71 |
| **4** | **16** | 62,83 | 35,30 | 50,00 | 100,00 | 0,00 | 62,72 | 35,30 | 50,00 | 100,00 | 0,00 |
| | **26** | 97,03 | 15,90 | 50,00 | 100,00 | 0,00 | 99,55 | 15,90 | 50,00 | 100,00 | 0,00 |
| | **6** | 1800,85 | 571,60 | 0,00 | 100,00 | 3,55 | 1800,07 | 407,90 | 0,00 | 100,00 | 4,34 |
| **8** | **16** | 1800,77 | 0,30 | 0,00 | 100,00 | 5,49 | 1800,89 | 0,10 | 0,00 | 100,00 | 5,49 |
| | **26** | 1800,73 | 0,00 | 0,00 | 100,00 | 8,30 | 1800,76 | 0,00 | 0,00 | 90,00 | 10,03 |
| | **6** | 1800,95 | 129,70 | 0,00 | 100,00 | 11,74 | 1800,91 | 92,90 | 0,00 | 100,00 | 11,67 |
| **12** | **16** | 1800,97 | 0,00 | 0,00 | 100,00 | 14,50 | 1800,07 | 0,00 | 0,00 | 100,00 | 14,50 |
| | **26** | 1800,18 | 0,00 | 0,00 | 80,00 | 9,51 | 1800,09 | 0,00 | 0,00 | 80,00 | 9,51 |
| | **6** | 1800,89 | 37,64 | 0,00 | 100,00 | 15,40 | 1800,85 | 23,18 | 0,00 | 100,00 | 15,68 |
| **16** | **16** | 1800,87 | 0,00 | 0,00 | 44,44 | 19,83 | 1800,75 | 0,00 | 0,00 | 44,44 | 19,83 |
| | **26** | 1800,69 | 0,00 | 0,00 | 70,00 | 8,74 | 1800,64 | 0,00 | 0,00 | 60,00 | 8,61 |
| | **6** | 1800,11 | 0,00 | 0,00 | 80,00 | 29,00 | 1800,01 | 0,00 | 0,00 | 80,00 | 29,00 |
| **20** | **16** | 1800,00 | 0,00 | 0,00 | 0,00 | $\infty$ | 1800,00 | 0,00 | 0,00 | 0,00 | $\infty$ |
| | **26** | 1800,00 | 0,00 | 0,00 | 0,00 | $\infty$ | 1800,00 | 0,00 | 0,00 | 0,00 | $\infty$ |

Para 70% das instâncias os algoritmos **Padrão**, **Regra** e **Regra–e–corte** tiveram o mesmo desempenho pois, com o tempo limitado em 30 minutos, os algoritmos finalizaram no nó zero e a mudança na regra de ramificação e a inserção de específicas desigualdades válidas só se aplicariam a árvores com mais nós.

O valor médio do *gap* com o algoritmo **Padrão** foi um pouco maior, 13,26%, em comparação com o *gap* dos algoritmos **Regra** (12,27%) e **Regra–e–corte** (12,38%). No entanto, **Padrão** foi um pouco mais eficiente em relação à porcentagem de instâncias para as quais a otimalidade foi provada (8%), enquanto nos outros dois, 7,35%. Em resumo, a mudança na regra de ramificação ou a inserção das desigualdades válidas propostas não alteraram significativamente o desempenho do algoritmo *branch–and–cut*.

A Tabela 6.9 exibe os seguintes valores médios para os algoritmos **Regra-GC** e **Regra–solini**: *tempo*, *#nós*, *%opt* e *gap*. Por definição destes algoritmos, a porcentagem de instâncias para as quais uma solução factível é obtida em 30 minutos é 100% para todas as classes.

Tabela 6.9: Resultados para os algoritmos **Regra–solini** e **Regra-GC**.

| | | **Regra–solini** | | | | **Regra-GC** | | | |
|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#nós*** | ***%opt*** | ***gap*** | ***tempo*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1433,58 | 19887,90 | 10,00 | 1,83 | 1464,13 | 11574,40 | 20,00 | 1,52 |
| **4** | **16** | 71,55 | 29,80 | 50,00 | 0,00 | 14,59 | 3,70 | 100,00 | 0,00 |
| | **26** | 134,90 | 20,30 | 50,00 | 0,00 | 0,00 | 0,00 | 100,00 | 0,00 |
| | **6** | 1800,55 | 346,50 | 0,00 | 3,44 | 1798,62 | 281,20 | 0,00 | 3,60 |
| **8** | **16** | 1800,85 | 10,00 | 0,00 | 4,27 | 359,96 | 1,70 | 80,00 | 0,06 |
| | **26** | 1800,99 | 1,10 | 0,00 | 1,75 | 179,96 | 0,00 | 100,00 | 0,11 |
| | **6** | 1800,45 | 108,20 | 0,00 | 8,49 | 1799,48 | 55,00 | 0,00 | 4,78 |
| **12** | **16** | 1801,99 | 0,20 | 0,00 | 4,20 | 359,92 | 0,00 | 80,00 | 0,08 |
| | **26** | 1802,20 | 0,00 | 0,00 | 1,56 | 0,00 | 0,00 | 100,00 | 0,00 |
| | **6** | 1800,68 | 33,70 | 0,00 | 13,45 | 1799,60 | 6,10 | 9,09 | 5,15 |
| **16** | **16** | 1804,28 | 0,00 | 0,00 | 3,72 | 359,87 | 0,00 | 77,78 | 0,05 |
| | **26** | 1804,38 | 0,00 | 0,00 | 1,26 | 361,08 | 0,00 | 80,00 | 0,07 |
| | **6** | 1802,37 | 0,50 | 0,00 | 14,52 | 1799,55 | 0,00 | 0,00 | 11,02 |
| **20** | **16** | 1805,47 | 0,00 | 0,00 | 4,83 | 359,89 | 0,00 | 80,00 | 0,05 |
| | **26** | 1807,57 | 0,00 | 0,00 | 1,56 | 0,00 | 0,00 | 100,00 | 0,00 |

Em relação ao **Regra–GC**, o algoritmo **Regra–solini** foi bem menos eficiente. Os valores médios do tempo computacional com **Regra–solini** foram de 1551,32 segundos e com **Regra–GC**, 710,44 segundos, do *gap* 4,32% e 1,76%, respectivamente e a porcentagem das instâncias para as quais a otimalidade pôde ser provada foi 14% no primeiro e 61,33% no segundo, evidenciando a importância de bons limitantes superiores e inferiores na resolução do problema. Similarmente ao ocorrido com os procedimentos de geração de colunas, o algoritmo **Regra–GC** teve um desempenho fraco para classes com $N = 6$ culturas. Para estas classes, o *gap* aumenta com o número de lotes. O algoritmo **Regra–Ini** teve comportamento similar, mas com influência menos significativa do número de culturas e mais do número de lotes.

Portanto, dentre os cinco algoritmos *branch–and–cut*, **Regra–GC** apresentou um desempenho superior, conseguindo melhorar, em média, os limitantes iniciais produzidos por **HGC**. De fato, para o procedimento **HGC** com *solini,* o valor médio do *gap* foi de 1,89% e, para 55,33% das instâncias, a otimalidade da solução foi provada em um tempo médio de 45,13 segundos.

Além dos algoritmos discutidos nesta subseção, outros *branch–and–cut* foram implementados, tais como, inserir os dois conjuntos de restrições de adjacências (3.6) e (3.8) e substituir as restrições (3.6) pelas restrições (3.8). Resultados preliminares com esses algoritmos mostram um desempenho similar ao do algoritmo **Padrão**.

### 6.1.3 *Branch–and–price*

Nesta subseção discutimos os resultados obtidos com a implementação de um conjunto de algoritmos tipo *branch–and–price* combinando as estratégias propostas na Subseção 5.2.6.

Em um primeiro conjunto de testes, definimos quatro algoritmos *branch–and–price.* Em dois, as colunas iniciais do PML 0–1 do nó zero são dadas por *sol0* e, nos outros dois, por *solini*. Em relação ao procedimento de geração de colunas em cada nó da árvore foi usado o procedimento **TGC** ou o procedimento **HGC**. Para os quatro algoritmos, a regra de ramificação é aquela proposta na subseção 5.2.5, os nós da árvore são ordenados pelo melhor limitante (**Z**) e o problema mestre restrito inicial para cada nó, excluindo o nó raiz, é formado por colunas artificiais. A Tabela 6.10 resume as características de cada algoritmo e a nomenclatura usada para identificá–lo.

Tabela 6.10: Algoritmos *branch–and–price*.

| **Nome** | ***Geração de colunas*** | ***Colunas Iniciais*** | ***Ramificação*** | ***Ordem dos nós*** | ***Colunas no PML do nó*** |
|---|---|---|---|---|---|
| **ToZArtR** | **TGC (T)** | *sol0* (**o**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) |
| **TiZArtR** | **TGC (T)** | *solini* (**i**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) |
| **HoZArtR** | **HGC (H)** | *sol0* (**o**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) |
| **HiZArtR** | **HGC(H)** | *solini* (**i**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) |

Na Tabela 6.11 exibimos os seguintes valores médios para os algoritmos **ToZArtR** e **TiZArtR**: tempo computacional (*tempo*), número de iterações (*#it*)

número de nós (*#nós*), porcentagem de instâncias em que otimalidade foi provada (*%opt*) e o *gap* do algoritmo (*gap*). Para o algoritmo **ToZArtR**, foi incluída uma coluna ($\%sol0^+$) indicando a porcentagem de instâncias em cada classe em que a solução obtida foi melhor que a solução inicial *sol0*. Para o cálculo do *gap* de cada classe foram excluídas as instâncias para as quais $solbest = sol0$.

Tabela 6.11:Resultados dos algoritmos **ToZArtR** e **TiZArtR**.

| | | **ToZArtR** | | | | | | **TiZArtR** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#it*** | ***#nós*** | ***%sol0⁺*** | ***%opt*** | ***gap*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1476,87 | 5999,90 | 137,50 | 30,00 | 10,00 | 0,14 | 1471,80 | 6309,60 | 155,70 | 0,00 | 17,78 |
| **4** | **16** | 153,27 | 1126,20 | 105,00 | 100,00 | 100,00 | 0,00 | 46,42 | 433,00 | 36,20 | 100,00 | 0,00 |
| | **26** | 84,79 | 459,30 | 49,80 | 100,00 | 100,00 | 0,00 | 109,83 | 545,90 | 60,70 | 100,00 | 0,00 |
| | **6** | 1827,11 | 3731,00 | 111,60 | 0,00 | 0,00 | – | 1697,40 | 3528,00 | 119,30 | 0,00 | 26,77 |
| **8** | **16** | 1484,85 | 5032,10 | 379,80 | 20,00 | 20,00 | 0,00 | 1595,51 | 5444,80 | 424,00 | 20,00 | 4,61 |
| | **26** | 1725,25 | 4354,00 | 378,50 | 10,00 | 10,00 | 0,00 | 1207,95 | 3252,20 | 258,50 | 50,00 | 3,03 |
| | **6** | 1834,58 | 4220,70 | 141,60 | 0,00 | 0,00 | – | 1807,10 | 4847,40 | 182,90 | 0,00 | 18,18 |
| **12** | **16** | 1478,14 | 4046,20 | 301,20 | 20,00 | 20,00 | 0,00 | 1803,10 | 4793,40 | 358,10 | 0,00 | 4,20 |
| | **26** | 1561,51 | 3110,20 | 264,40 | 20,00 | 20,00 | 0,00 | 1802,76 | 3628,10 | 307,90 | 0,00 | 1,56 |
| | **6** | 1827,06 | 3090,70 | 96,30 | 0,00 | 0,00 | – | 1807,80 | 4240,60 | 144,30 | 0,00 | 18,06 |
| **16** | **16** | 1803,48 | 4025,30 | 278,40 | 0,00 | 0,00 | – | 1805,20 | 4402,70 | 291,90 | 0,00 | 3,90 |
| | **26** | 1803,88 | 3133,50 | 228,50 | 0,00 | 0,00 | – | 1803,75 | 3377,00 | 260,80 | 0,00 | 1,65 |
| | **6** | 1845,45 | 2353,53 | 49,45 | 0,00 | 0,00 | – | 1812,18 | 2617,40 | 61,50 | 0,00 | 22,26 |
| **20** | **16** | 1624,55 | 3009,30 | 188,90 | 20,00 | 20,00 | 0,00 | 1804,92 | 3657,80 | 207,50 | 0,00 | 4,83 |
| | **26** | 1810,58 | 2550,40 | 172,00 | 0,00 | 0,00 | – | 1808,69 | 2752,10 | 169,10 | 0,00 | 1,56 |

Como pode ser observado na tabela, os algoritmos **ToZArtR** e **TiZArtR** não tiveram um bom desempenho. O primeiro só conseguiu encontrar solução melhor que a solução inicial *sol0* para 21,33% das instâncias e a otimalidade foi provada para somente 20% delas. O algoritmo **TiZArtR** teve um *gap* médio de 8,56% e a otimalidade foi provada para 18,00% das instâncias. Nenhuma solução ótima foi provada para classes com $L = 12$ lotes ou mais.

A Tabela 6.12 exibe resultados para os algoritmos **HoZArtR** e **HiZArtR**. A tabela mostra os valores médios do *tempo*, *#it*, *#nós*, *%opt* e do *gap* de cada algoritmo. O algoritmo **HoZArtR** obtém solução melhor que *sol0* para todas as instâncias.

Tabela 6.12: Resultados dos algoritmos **HoZArtR** e **HiZArtR**.

| | | **HoZArtR** | | | | | **HiZArtR** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1451,80 | 6099,40 | 135,40 | 20,00 | 1,97 | 1482,30 | 6689,50 | 152,30 | 20,00 | 1,97 |
| **4** | **16** | 4,00 | 48,30 | 1,70 | 100,00 | 0,00 | 3,65 | 41,30 | 1,40 | 100,00 | 0,00 |
| | **26** | 3,43 | 30,70 | 1,00 | 100,00 | 0,00 | 2,52 | 21,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1833,77 | 4152,20 | 99,10 | 10,00 | 2,99 | 1817,93 | 4124,90 | 95,70 | 0,00 | 3,12 |
| **8** | **16** | 14,63 | 119,00 | 5,20 | 100,00 | 0,00 | 8,91 | 79,60 | 2,60 | 100,00 | 0,00 |
| | **26** | 6,99 | 61,90 | 1,00 | 100,00 | 0,00 | 5,44 | 40,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1667,22 | 3713,10 | 92,90 | 10,00 | 3,35 | 1718,15 | 3824,90 | 101,60 | 10,00 | 3,37 |
| **12** | **16** | 8,24 | 75,80 | 1,00 | 100,00 | 0,00 | 12,23 | 97,60 | 2,20 | 100,00 | 0,00 |
| | **26** | 12,97 | 84,00 | 1,40 | 100,00 | 0,00 | 9,96 | 64,90 | 1,30 | 100,00 | 0,00 |
| | **6** | 1832,42 | 3195,10 | 75,70 | 0,00 | 3,24 | 1834,29 | 3161,50 | 73,90 | 0,00 | 3,09 |
| **16** | **16** | 210,91 | 611,20 | 22,10 | 90,00 | 0,01 | 29,02 | 163,70 | 4,30 | 100,00 | 0,00 |
| | **26** | 15,33 | 99,60 | 1,00 | 100,00 | 0,00 | 15,75 | 86,50 | 1,20 | 100,00 | 0,00 |
| | **6** | 1827,31 | 2747,20 | 58,90 | 0,00 | 5,11 | 1884,43 | 2739,00 | 51,90 | 0,00 | 6,37 |
| **20** | **16** | 108,30 | 364,80 | 10,80 | 100,00 | 0,00 | 68,50 | 277,00 | 8,60 | 100,00 | 0,00 |
| | **26** | 25,86 | 149,20 | 1,40 | 100,00 | 0,00 | 19,40 | 114,20 | 1,20 | 100,00 | 0,00 |

O desempenho dos algoritmos **HoZArtR** e **HiZArtR** foi similar em relação aos valores médios dos medidas consideradas: tempo computacional (601,55 e 594,16 segundos, respectivamente), número de iterações (1436,77 e 1435,12), número de nós (33,91 e 33,35), porcentagem de instâncias em que a otimalidade foi provada (68,67% para ambos) e *gap* (1,11% e 1,20%). Para as classes com $N = 6$ culturas, **HoZArtR** provou a otimalidade somente para 8% das instâncias com um *gap* médio de 3,33%. O segundo, para 6% das instâncias, com *gap* médio de 3,59%. Para as classes com $N = 16$ e $N = 26$ culturas, com o algoritmo **HoZArtR**, a otimalidade foi provada para 99% das instâncias em uma média de 41,07 segundos e, com **HiZArtR**, para 100% das instâncias em uma média de 17,54 segundos. Em geral, o tempo computacional cresce um pouco com o número de lotes nos dois algoritmos.

Em um segundo conjunto de testes alguns parâmetros do algoritmo **HiZArtR** foram alterados. Em alternativa às colunas artificiais disponíveis em cada nó da árvore de enumeração, usamos a heurística *lote–a–lote–completa* para gerar um PML 0–1 inicial factível. Para o algoritmo de geração de colunas **HGC**, geramos uma versão alternativa (nomeada **HlGC**) em que a função objetivo do subproblema $H_k$ foi substituída pela função do problema PRC–A para o lote $k$, ou seja,

$$\max \sum_{i \in C} \sum_{j=1}^{M} t_i x_{ij} \tag{6.1}$$

Além desses parâmetros, a regra de ramificação foi substituída por uma ramificação padrão, em que a variável original com valor mais próximo de ½ é a escolhida. Para a ordenação dos nós, também foram implementadas a busca em profundidade e em largura. Por fim, desigualdades válidas como descritas na Subseção 5.2.5 são inseridas. Desta forma, foram definidos seis algoritmos.

A Tabela 6.13 resume as características de cada um e as Tabelas 6.14 a 6.17 exibem os resultados obtidos.

Tabela 6.13: Algoritmos *branch–and–price*.

| **Nome** | ***Geração de colunas*** | ***Colunas Iniciais*** | ***Ramificação*** | ***Ordem dos nós*** | ***Colunas no PML do nó*** | ***Corte*** |
|---|---|---|---|---|---|---|
| **HiZHeuR** | **HGC (H)** | *solini* (**i**) | *alternativa* (***R***) | **Z** | *heurística* (**Heu**) | |
| **HliZArtR** | **HlGC (Hl)** | *solini* (**i**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) | |
| **HiLargArtR** | **HGC (H)** | *solini* (**i**) | *alternativa* (***R***) | **Larg** | *artificiais* (**Art**) | |
| **HiProfArtR** | **HGC (H)** | *solini* (**i**) | *alternativa* (***R***) | **Prof** | *artificiais* (**Art**) | |
| **HiZArtP** | **HGC (H)** | *solini* (**i**) | *padrão* (**P**) | **Z** | *artificiais* (**Art**) | |
| **HiZArtRc** | **HGC (H)** | *solini* (**i**) | *alternativa* (***R***) | **Z** | *artificiais* (**Art**) | (**c**) |

Tabela 6.14: Resultados dos algoritmos **HiZHeuR** e **HliZArtR**.

| | | **HiZHeuR** | | | | | **HliZArtR** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1498,41 | 4550,70 | 156,60 | 20,00 | 1,98 | 1474,52 | 5285,90 | 116,00 | 20,00 | 2,98 |
| **4** | **16** | 3,90 | 39,60 | 1,50 | 100,00 | 0,00 | 5,22 | 47,90 | 1,70 | 100,00 | 0,00 |
| | **26** | 2,52 | 17,60 | 1,00 | 100,00 | 0,00 | 3,80 | 28,00 | 1,00 | 100,00 | 0,00 |
| | **6** | 1844,03 | 3103,00 | 62,80 | 0,00 | 4,36 | 1673,07 | 3406,10 | 78,90 | 10,00 | 3,17 |
| **8** | **16** | 39,80 | 243,60 | 12,40 | 100,00 | 0,00 | 7,31 | 63,60 | 1,90 | 100,00 | 0,00 |
| | **26** | 5,25 | 32,60 | 1,00 | 100,00 | 0,00 | 6,29 | 48,20 | 1,10 | 100,00 | 0,00 |
| | **6** | 1852,17 | 4202,70 | 96,80 | 0,00 | 3,48 | 1691,25 | 4017,10 | 115,10 | 10,00 | 4,68 |
| **12** | **16** | 16,32 | 104,80 | 2,80 | 100,00 | 0,00 | 13,84 | 104,70 | 2,40 | 100,00 | 0,00 |
| | **26** | 9,68 | 52,30 | 1,10 | 100,00 | 0,00 | 9,49 | 64,50 | 1,00 | 100,00 | 0,00 |
| | **6** | 1835,99 | 3209,00 | 58,20 | 0,00 | 3,13 | 1703,12 | 2657,60 | 59,30 | 10,00 | 3,31 |
| **16** | **16** | 20,07 | 114,20 | 1,70 | 100,00 | 0,00 | 106,26 | 345,40 | 13,70 | 100,00 | 0,00 |
| | **26** | 19,76 | 80,50 | 1,50 | 100,00 | 0,00 | 22,00 | 112,10 | 1,70 | 100,00 | 0,00 |
| | **6** | 1864,93 | 2483,60 | 31,50 | 0,00 | 6,70 | 1882,34 | 2653,10 | 54,00 | 0,00 | 6,06 |
| **20** | **16** | 431,78 | 802,10 | 22,70 | 80,00 | 0,02 | 22,01 | 147,40 | 1,40 | 100,00 | 0,00 |
| | **26** | 65,12 | 178,20 | 4,40 | 100,00 | 0,00 | 19,04 | 111,00 | 1,00 | 100,00 | 0,00 |

Em comparação com **HiZArtR**, o algoritmo **HiZHeuR**, no qual colunas artificiais são substituídas por colunas construídas com soluções da heurística *lote–a–lote–completa*, foi um pouco menos eficiente. Ainda que esta estratégia tenha diminuído em 8,84% o valor médio do número de nós e em 10,74%, o valor médio do número de iterações do algoritmo, os valores médios do tempo computacional e do *gap* aumentaram, respectivamente, em 6,7% e em 9,61%, e o número de instâncias para as quais a otimalidade foi provada diminuiu 2,91%.

Um pouco mais eficaz foi a substituição da função objetivo do subproblema $H_k$ por (6.1). Em **HliZArtR**, diminuíram os valores médios do tempo computacional (0,46%), do número de nós (10%) e do número de iterações (11,31%) e a porcentagem de instâncias em que a otimalidade foi provada aumentou (1,94%). No entanto, o valor médio do *gap* do algoritmo aumentou (12,68%).

Tabela 6.15: Resultados dos algoritmos **HiLargArtR** e **HiProfArtR**.

| | | **HiLargArtR** | | | | | **HiProfArtR** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1466,69 | 6213,00 | 143,60 | 20,00 | 2,02 | 1698,35 | 8312,90 | 237,90 | 10,00 | 2,23 |
| **4** | **16** | 4,39 | 47,90 | 1,90 | 100,00 | 0,00 | 4,40 | 42,60 | 3,70 | 100,00 | 0,00 |
| | **26** | 2,53 | 21,60 | 1,00 | 100,00 | 0,00 | 2,49 | 21,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1699,85 | 3650,00 | 77,60 | 10,00 | 3,08 | 1631,99 | 2390,50 | 125,20 | 10,00 | 3,42 |
| **8** | **16** | 8,96 | 82,10 | 2,60 | 100,00 | 0,00 | 217,96 | 611,80 | 66,30 | 90,00 | 0,02 |
| | **26** | 5,41 | 40,60 | 1,00 | 100,00 | 0,00 | 5,34 | 40,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1831,42 | 4503,90 | 121,00 | 0,00 | 2,85 | 1652,81 | 1770,70 | 154,90 | 10,00 | 3,49 |
| **12** | **16** | 12,89 | 101,10 | 2,80 | 100,00 | 0,00 | 19,76 | 94,00 | 11,30 | 100,00 | 0,00 |
| | **26** | 11,39 | 71,00 | 1,80 | 100,00 | 0,00 | 10,67 | 65,00 | 2,30 | 100,00 | 0,00 |
| | **6** | 1666,28 | 2958,50 | 56,50 | 10,00 | 3,25 | 1838,52 | 1572,30 | 118,00 | 0,00 | 3,33 |
| **16** | **16** | 48,17 | 257,20 | 6,70 | 100,00 | 0,00 | 16,45 | 110,10 | 4,30 | 100,00 | 0,00 |
| | **26** | 20,09 | 100,80 | 2,00 | 100,00 | 0,00 | 23,29 | 96,50 | 5,40 | 100,00 | 0,00 |
| | **6** | 1869,19 | 2986,80 | 57,40 | 0,00 | 5,70 | 1852,62 | 1192,20 | 118,90 | 0,00 | 9,25 |
| **20** | **16** | 191,84 | 590,60 | 24,00 | 100,00 | 0,00 | 386,46 | 331,20 | 50,00 | 80,00 | 0,03 |
| | **26** | 22,32 | 126,60 | 1,80 | 100,00 | 0,00 | 23,01 | 115,80 | 3,20 | 100,00 | 0,00 |

Em geral, a escolha do próximo nó pelo melhor limitante teve um desempenho superior, fixando os outros parâmetros em **HiZArtR**, em relação à busca em largura ou profundidade. Para os algoritmos **HiLargArtR** e **HiProfArtR** aumentaram os valores médios do tempo computacional (8,89% e 12,03%, respectivamente), número de nós (7,64% e 79,97%) e do *gap* (2,78% e 30,88%). Além disto, o número de

instâncias em que a otimalidade foi provada diminuiu (3,4% e 6,31%). Quanto ao valor médio do número de iterações, em **HiLargArtR** aumentou 9,24% e, em **HiProfArtR**, diminuiu 19,30%.

Tabela 6.16: Resultados dos algoritmos **HiZArtP** e **HiZArtRc**.

| | | **HiZArtP** | | | | | **HiZArtRc** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***L*** | ***N*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** | ***tempo*** | ***#it*** | ***#nós*** | ***%opt*** | ***gap*** |
| | **6** | 1387,94 | 5463,30 | 130,70 | 30,00 | 1,44 | 1503,85 | 6587,50 | 149,10 | 20,00 | 1,97 |
| **4** | **16** | 3,68 | 40,60 | 1,40 | 100,00 | 0,00 | 3,70 | 41,30 | 1,40 | 100,00 | 0,00 |
| | **26** | 2,56 | 21,60 | 1,00 | 100,00 | 0,00 | 2,51 | 21,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1699,92 | 3456,70 | 68,40 | 10,00 | 3,04 | 1677,74 | 3359,50 | 72,60 | 10,00 | 3,04 |
| **8** | **16** | 27,87 | 188,60 | 8,50 | 100,00 | 0,00 | 14,90 | 110,20 | 4,00 | 100,00 | 0,00 |
| | **26** | 5,45 | 40,60 | 1,00 | 100,00 | 0,00 | 5,47 | 40,60 | 1,00 | 100,00 | 0,00 |
| | **6** | 1836,95 | 4202,30 | 109,20 | 0,00 | 3,27 | 1840,03 | 4271,00 | 121,60 | 0,00 | 3,40 |
| **12** | **16** | 14,23 | 110,70 | 2,50 | 100,00 | 0,00 | 19,04 | 123,50 | 3,90 | 100,00 | 0,00 |
| | **26** | 10,33 | 64,20 | 1,40 | 100,00 | 0,00 | 9,65 | 62,70 | 1,20 | 100,00 | 0,00 |
| | **6** | 1673,09 | 2705,90 | 46,50 | 10,00 | 2,98 | 1830,41 | 3202,40 | 71,40 | 0,00 | 3,31 |
| **16** | **16** | 50,86 | 254,90 | 7,30 | 100,00 | 0,00 | 30,40 | 177,90 | 4,10 | 100,00 | 0,00 |
| | **26** | 20,68 | 101,90 | 1,90 | 100,00 | 0,00 | 31,42 | 132,70 | 3,30 | 100,00 | 0,00 |
| | **6** | 1855,23 | 2895,10 | 54,40 | 0,00 | 6,09 | 1828,46 | 2693,60 | 52,70 | 0,00 | 7,06 |
| **20** | **16** | 125,67 | 382,20 | 13,80 | 100,00 | 0,00 | 49,22 | 235,20 | 6,20 | 100,00 | 0,00 |
| | **26** | 21,37 | 120,20 | 1,40 | 100,00 | 0,00 | 24,63 | 126,80 | 2,00 | 100,00 | 0,00 |

Alterar no algoritmo **HiZArtR** a regra de ramificação para a regra padrão (algoritmo **HiZArtP**) diminuiu 4,92% do valor médio do número de nós. No entanto, aumentou 7,27% no valor médio do tempo de execução, 0,38% no número de iterações e 2,27% do *gap* do algoritmo. Também diminuiu 2,43% da porcentagem de instâncias em que a otimalidade foi provada. Um pouco mais eficiente foi a inserção das desigualdades válidas propostas (**HiZArtRc**). Esta estratégia diminuiu o valor médio do tempo computacional (0,46%) e do número de nós (0,94%) e de iterações (1,58%). A porcentagem de instâncias em que otimalidade foi provada permaneceu a mesma. No entanto, o valor médio do *gap* aumentou 4,77%.

Em resumo, as mudanças nos parâmetros propostos para o algoritmo **HiZArtR** quando trouxeram benefícios, estes não foram muito significativos. De fato, de maneira geral, os algoritmos *branch–and–price* implementados com os procedimentos de geração de colunas combinados com a heurística *lote–a–lote* tiveram

comportamento similar e foram bem mais eficientes que os algoritmos implementados sem esta combinação (**ToZArtR** e **TiZArtR**). Além disto, diferentemente do que ocorreu com os algoritmos *branch–and–cut*, o desempenho dos algoritmos *branch–and–price* com **HGC** ou **HlGC** não foram muito afetados pelo número de lotes. O número de culturas, no entanto, interferiu bastante na eficiência desses algoritmos. Para as classes com $N = 6$ a porcentagem das instâncias para as quais a otimalidade foi provada foi baixa, ainda que o valor médio do *gap* tenha sido bem pequeno. Por outro lado, para classes com $N = 16$ e $N = 26$ culturas, a otimalidade só não foi provada para 100% das instâncias pelo algoritmo **HiProfArtR**.

A influência do número de culturas no desempenho dos algoritmos *branch–and–price* com **HGC** e **HlGC** pode ser visto na Tabela 6.17 que exibe os valores médios das medidas observadas para as classes com $N = 6$ e com $N = 16$–$26$ culturas.

Tabela 6.17: Resultados dos algoritmos *branch–and–price*.

| Nome | *N* | *tempo* | *#it* | *#nós* | *%opt* | *gap* |
|---|---|---|---|---|---|---|
| | **6** | 1722,50 | 3981,40 | 92,40 | 8,00 | 3,33 |
| **HoZArtR** | **16–26** | 41,07 | 164,45 | 4,66 | 99,00 | 0,00 |
| | **média** | **601,55** | **1436,77** | **33,91** | **68,67** | **1,11** |
| | **6** | 1747,42 | 4107,96 | 95,08 | 6,00 | 3,59 |
| **HiZArtR** | **16–26** | 17,54 | 98,70 | 2,48 | 100,00 | 0,00 |
| | **média** | **594,16** | **1435,12** | **33,35** | **68,67** | **1,20** |
| | **6** | 1739,30 | 3312,86 | 75,38 | 6,00 | 4,72 |
| **HiZHeuR** | **16–26** | 45,41 | 138,13 | 3,67 | 99,00 | 0,00 |
| | **média** | **610,04** | **1196,37** | **27,57** | **68,00** | **1,57** |
| | **6** | 1684,86 | 3603,96 | 84,66 | 10,00 | 4,04 |
| **HliZArtR** | **16–26** | 21,53 | 107,28 | 2,69 | 100,00 | 0,00 |
| | **média** | **575,97** | **1272,84** | **30,01** | **70,00** | **1,35** |
| | **6** | 1706,69 | 4062,44 | 91,22 | 8,00 | 3,38 |
| **HiLargArtR** | **16–26** | 32,80 | 143,95 | 4,56 | 100,00 | 0,00 |
| | **média** | **647,01** | **1567,66** | **35,89** | **66,33** | **1,23** |
| | **6** | 1734,86 | 3047,72 | 150,98 | 6,00 | 4,34 |
| **HiProfArtR** | **16–26** | 70,98 | 152,92 | 14,85 | 97,00 | 0,01 |
| | **média** | **665,65** | **1158,10** | **60,01** | **64,33** | **1,56** |
| | **6** | 1690,63 | 3744,66 | 81,84 | 10,00 | 3,36 |
| **HiZArtP** | **16–26** | 28,27 | 132,55 | 4,02 | 100,00 | 0,00 |
| | **média** | **637,38** | **1440,59** | **31,71** | **67,00** | **1,22** |
| | **6** | 1736,10 | 4022,80 | 93,48 | 6,00 | 3,76 |
| **HiZArtRc** | **16–26** | 19,09 | 107,25 | 2,81 | 100,00 | 0,00 |
| | **média** | **591,43** | **1412,43** | **33,03** | **68,67** | **1,25** |

Ainda na expectativa de encontrar melhores resultados, principalmente para classes com $N = 6$ culturas, além dos algoritmos *branch–and–price* apresentados nesta subseção, outros foram implementados. Dentre eles, grande parte das possíveis combinações com as mudanças de parâmetros mencionadas, a substituição do conjunto de restrições de adjacências (3.8) pelo conjunto (3.6) e a inserção desses dois conjuntos de restrições (3.6) e (3.8). Os resultados foram similares aos já discutidos e, por isto, omitidos. Assim, os algoritmos *branch–and–price* com a geração de colunas **HGC** tiveram desempenho bem superior aos algoritmos *branch–and–price* com a geração de colunas **TGC** e aos algoritmos *branch–and–cut* implementados.

## 6.2 Resultados computacionais para o problema PRC-D

Na Seção 5.3 foram propostos um algoritmo de geração de colunas para resolver o problema PRC–D e os procedimentos *α–lote* e *redução–lote* para a redução do número de lotes na solução. Para análise de desempenho destes algoritmos, testes computacionais foram realizados para um horizonte de planejamento de 2 anos dividido em semanas. O período obrigatório de pousio em cada programação de produção foi fixado em 4 semanas. Em relação ao número de culturas, as instâncias dividem–se em 4 grupos com $N = 12$, 16, 20 e 24. Os dados das culturas estão exibidos nas tabelas B.1 e B.2 do Apêndice B e foram obtidos em uma horta de base ecológica situada em Barbacena–MG.

Para cada classe de instâncias, o desempenho do algoritmo foi testado para uma área de plantio com 10.000 m$^2$, dividida em $L = 1$, 3 e 5 áreas. O caso em que $L = 1$ corresponde ao problema PRC–D para áreas homogêneas detalhado na subseção 4.2.1. As classes com $L = 3$ e $L = 5$ correspondem a áreas heterogêneas e foram simulados da seguinte forma: um conjunto de culturas foi selecionado para cada área de plantio variando a produtividade de uma cultura, no máximo, $\pm 30\%$ em relação aos valores coletados em Barbacena. A demanda também foi gerada aleatoriamente respeitando a época de colheita de cada cultura. A eficiência dos métodos foi testada para a situação em que não existe limitante para a produção e para uma produção extra limitada a 200% do valor da demanda. Estes casos são apresentamos nas tabelas como d$\infty$ e d200, respectivamente.

Com a combinação dos diferentes parâmetros do número de culturas ($N = 12$, 16, 20 e 24), áreas de plantio ($L = 1$, 3 e 5) e limitantes da produção (d$\infty$ e d200), foram geradas 24 classes, cada uma com 60 instâncias, perfazendo um total de 1440 instâncias. Em todos os casos foram incluídas variáveis artificiais para permitir o não–atendimento da demanda, como discutido na subseção 4.2.2, assumindo uma penalidade igual para todas as demandas não atendidas. A função objetivo considerada foi a maximização da produção na área de plantio. Para todos os experimentos, o tempo computacional foi limitado em 1 hora.

Nas Tabelas 6.18 a 6.20 são apresentados os resultados obtidos com o algoritmo de geração de colunas. Para cada classe de instâncias, as tabelas mostram os valores médios para o tempo de execução, em segundos, do procedimento (*tempo*), número de lotes na solução ótima ($\#lotes$), porcentagem da demanda não atendida ($\%nad$) e porcentagem da área total ocupada na solução ótima ($\%oarea$).

Tabela 6.18: Resultados para instâncias com $L = 1$

| $L = 1$ | | | | | |
|---|---|---|---|---|---|
| ***cultura*** | ***demanda*** | ***tempo*** | ***# lotes*** | ***%nad*** | ***%oarea*** |
| **12** | *d200* | 250,20 | 224,30 | 0,73 | 72,85 |
| | *d*$\infty$ | 757,25 | 119,80 | 0,00 | 100,00 |
| **16** | *d200* | 910,35 | 395,40 | 0,47 | 86,88 |
| | *d*$\infty$ | 2331,47 | 237,00 | 0,00 | 100,00 |
| **20** | *d200* | 3283,76 | 494,70 | 0,50 | 100,00 |
| | *d*$\infty$ | 3075,58 | 426,30 | 0,00 | 100,00 |
| **24** | *d200* | 3230,28 | 549,40 | 0,60 | 100,00 |
| | *d*$\infty$ | 3601,36 | 528,22 | 0,00 | 100,00 |

Tabela 6.19: Resultados para instâncias com $L = 3$

| $L = 3$ | | | | | |
|---|---|---|---|---|---|
| ***cultura*** | ***demanda*** | ***tempo*** | ***# lotes*** | ***%nad*** | ***%oarea*** |
| **12** | *d200* | 257,37 | 266,10 | 0,51 | 84,53 |
| | *d*$\infty$ | 646,62 | 116,89 | 0,00 | 100,00 |
| **16** | *d200* | 811,76 | 335,40 | 0,63 | 91,82 |
| | *d*$\infty$ | 1283,94 | 244,50 | 0,00 | 100,00 |
| **20** | *d200* | 2630,80 | 497,90 | 0,49 | 100,00 |
| | *d*$\infty$ | 3430,26 | 457,56 | 0,00 | 100,00 |
| **24** | *d200* | 3230,28 | 549,40 | 0,60 | 100,00 |
| | *d*$\infty$ | 3596,96 | 594,50 | 0,00 | 100,00 |

Tabela 6.20: Resultados para instâncias com $L = 5$

| $L = 5$ | | | | | |
|---|---|---|---|---|---|
| ***cultura*** | ***demanda*** | ***tempo*** | ***# lotes*** | ***%nad*** | ***%oarea*** |
| **12** | *d200* | 129,83 | 249,60 | 0,68 | 64,81 |
| | *d∞* | 708,97 | 135,60 | 0,00 | 100,00 |
| **16** | *d200* | 766,70 | 394,50 | 0,48 | 94,30 |
| | *d∞* | 2722,03 | 241,50 | 0,00 | 100,00 |
| **20** | *d200* | 2062,26 | 502,40 | 0,67 | 100,00 |
| | *d∞* | 2947,01 | 424,80 | 0,00 | 100,00 |
| **24** | *d200* | 3168,13 | 546,67 | 0,60 | 100,00 |
| | *d∞* | 3612,39 | 635,89 | 0,00 | 100,00 |

Como exibido nas Tabelas 6.18 a 6.20, a demanda foi atendida em quase sua totalidade (acima de 99% dos casos) no tempo limite permitido. O tempo de execução do algoritmo aumentou com o número de culturas, mas não necessariamente com o número de áreas de plantio. Em geral, as classes para as quais a produção extra foi limitada foram resolvidas mais rapidamente. A área de plantio foi utilizada completamente quando não houve limitação da produção, como esperado, ao passo que, com a produção extra limitada a 200% e poucas culturas, isto não ocorreu.

O número de áreas de plantio não tem clara influência no número de lotes na solução ótima ao contrário do número de culturas, em que esta dependência é bastante significativa. De fato, quanto maior o número de culturas, maior o número de diferentes programações de produção na solução ótima alcançando, em média, 600 programações para instâncias com $N = 24$. Na tentativa de diminuir o número de lotes, os procedimentos *α–lote* e *redução–lote* foram usados.

A aplicação da estratégia *α–lote* com $\alpha = 1\text{m}^2$, produziu os resultados exibidos na Tabela 6.21. Para cada classe, a tabela mostra o número de lotes na solução ótima ($\#lotes$), a porcentagem da área total descartada pelo procedimento ($\%darea$) e a porcentagem da demanda não atendida em função das áreas descartadas ($\%nad$).

Tabela 6.21: Resultados para o procedimento $\alpha$–*lote* com $\alpha = 1\text{m}^2$

| | | $L = 1$ | | | $L = 3$ | | | $L = 5$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ***cultura*** | ***dmax*** | ***#lotes*** | ***%darea*** | ***%nad*** | ***#lotes*** | ***%darea*** | ***%nad*** | ***#lotes*** | ***%darea*** | ***%nad*** |
| **12** | *d200* | 201,40 | 0,11 | 0,86 | 233,10 | 0,15 | 1,18 | 224,80 | 0,12 | 1,54 |
| | *d∞* | 114,70 | 0,03 | 0,15 | 110,00 | 0,03 | 0,50 | 128,30 | 0,04 | 0,38 |
| **16** | *d200* | 336,60 | 0,26 | 0,68 | 303,90 | 0,14 | 1,19 | 351,10 | 0,19 | 0,62 |
| | *d∞* | 221,50 | 0,07 | 0,42 | 232,00 | 0,06 | 0,42 | 230,10 | 0,06 | 0,38 |
| **20** | *d200* | 432,00 | 0,30 | 1,30 | 442,30 | 0,25 | 0,62 | 443,50 | 0,27 | 1,06 |
| | *d∞* | 378,80 | 0,23 | 1,22 | 420,33 | 0,18 | 0,93 | 390,60 | 0,16 | 0,70 |
| **24** | *d200* | 478,20 | 0,33 | 1,32 | 478,20 | 0,33 | 1,32 | 487,89 | 0,28 | 1,40 |
| | *d∞* | 469,44 | 0,26 | 0,93 | 529,80 | 0,31 | 1,49 | 561,67 | 0,35 | 1,68 |

A utilização dessa estratégia simples permitiu a redução do número de lotes em torno de 10%. Além disto, como as áreas descartadas eram pequenas, em média, o aumento da demanda não atendida foi pouco significativo (em torno de 1%).

A Tabela 6.22 exibe os seguintes resultados obtidos com a heurística *redução–lote*: tempo de execução do algoritmo (*tempo*), número de lotes na solução ótima (*#lotes*) e a porcentagem da área total ocupada (%o*area*).

Tabela 6.22: Resultados para o procedimento *redução–lote*

| | | $L = 1$ | | | $L = 3$ | | | $L = 5$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ***cultura*** | ***dmax*** | ***tempo*** | ***#lotes*** | ***%oarea*** | ***tempo*** | ***#lotes*** | ***%oarea*** | ***tempo*** | ***#lotes*** | ***%oarea*** |
| **12** | *d200* | 259,99 | 73,70 | 38,03 | 591,09 | 86,70 | 46,97 | 333,66 | 75,40 | 35,65 |
| | *d∞* | 409,33 | 33,80 | 83,06 | 1026,59 | 36,22 | 81,89 | 1442,74 | 37,00 | 69,48 |
| **16** | *d200* | 463,94 | 137,00 | 53,56 | 844,12 | 120,70 | 52,07 | 764,38 | 138,50 | 58,76 |
| | *d∞* | 1340,40 | 55,30 | 80,84 | 1800,03 | 61,40 | 88,63 | 1800,06 | 69,90 | 95,23 |
| **20** | *d200* | 1551,62 | 181,20 | 73,48 | 1323,63 | 192,40 | 82,38 | 1638,11 | 190,90 | 79,36 |
| | *d∞* | 1800,09 | 169,60 | 100,00 | 1800,10 | 195,33 | 95,34 | 1800,22 | 189,50 | 90,63 |
| **24** | *d200* | 1540,84 | 247,80 | 89,49 | 1540,84 | 247,80 | 89,49 | 1800,18 | 301,78 | 97,50 |
| | *d∞* | 1800,41 | 278,00 | 100,00 | 1800,56 | 272,90 | 100,00 | 1800,46 | 302,89 | 91,11 |

A redução no número de lotes neste caso alcançou 60%, em média, em comparação com a solução inicial fornecida ao procedimento. Nesta heurística não foi permitido aumento do não atendimento da demanda (as variáveis artificiais foram eliminadas) e não houve controle do tamanho dos lotes. Assim, lotes considerados muito pequenos podem ainda ocorrer na solução ótima e a estratégia $\alpha$–*lote* poderia ser aplicada, se desejável.

# Capítulo 7

# Conclusões e Perspectivas

Nesta tese caracterizamos dois problemas de planejamento agrícola dirigidos à produção de base ecológica de hortaliças e propomos modelos matemáticos e procedimentos heurísticos e exatos para resolvê–los. O planejamento da produção de hortaliças é um problema de otimização combinatória complexo pois envolve, em geral, um grande número de culturas com limitações específicas quanto à época de plantio e com períodos de cultivo e produtividades muito variáveis.

A rotação de culturas desempenha um papel central nos dois problemas nomeados *Programação de Rotação de Culturas com restrições de Adjacências* (PRC–A) e *Programação de Rotação de Culturas com atendimento da Demanda* (PRC–D). No Capítulo 2, um modelo de programação linear 0–1 é apresentado para determinar as programações de rotações de culturas factíveis para cada área de plantio. Como o planejamento é dirigido à produção de base ecológica de hortaliças, uma rotação de cultura é considerada factível quando em sua programação, além de restrições técnicas, tais como, época apropriada de plantio e ciclo de cultivo da cultura, são respeitadas as seguintes restrições de base ecológica: culturas de mesma família botânica não são plantadas em sequência e, em cada programação de uma rotação de culturas, são incluídos o plantio de uma cultura para adubação verde e um pousio.

No problema PRC–A, apresentado no Capítulo 3, determina–se uma programação de rotação de culturas factível (como definido no Capítulo 2) para cada um dos lotes de uma área de plantio de modo que, para lotes adjacentes, não sejam cultivadas, em uma mesma época, culturas de mesma família botânica. Um modelo de programação linear 0–1 para o problema é proposto com o objetivo de maximizar a ocupação dos lotes por rotações de culturas as quais têm um tamanho pré–definido e igual para todos os lotes. Demonstramos que, com este objetivo, para lotes nos formatos tabuleiro ou paralelo uma solução ótima para o problema é construída a

partir de uma solução particular para apenas dois lotes adjacentes. Para outras configurações espaciais, mostramos que a existência de soluções para grafos completos com 2, 3 e 4 vértices é suficiente para garantir a obtenção de uma solução factível para qualquer área de plantio.

A formulação matemática para o problema PRC-A tem um número muito grande de restrições e variáveis, o que dificulta sua resolução usando um algoritmo *branch–and–bound* padrão. Por outro lado, o problema pode ser decomposto em subproblemas, cada um associado a uma programação de rotação para um lote de plantio, ligados pelas restrições de vizinhança espacial para lotes adjacentes. Com esta estratégia de decomposição, apresentamos duas heurísticas construtivas gulosas que foram utilizadas tanto para determinar rapidamente boas soluções factíveis para o problema quanto acelerar algoritmos exatos. Além disto, o modelo foi reformulado usando o princípio da Decomposição de Dantzig–Wolfe, o que permitiu sua resolução usando a técnica de geração de colunas em procedimentos heurísticos e em algoritmos do tipo *branch–and–price*.

Para avaliar o modelo proposto para o problema PRC–A, bem como os procedimentos desenvolvidos para resolvê–lo, experimentos computacionais foram realizados usando dados reais com rotações de 2 anos e envolvendo até 28 culturas. Os resultados indicam que, em geral, os procedimentos baseados em geração de colunas, quando combinados com as heurísticas construtivas, produzem boas soluções e em tempo de processamento bastante razoável. Além disto, a reformulação do problema permitiu a obtenção de solução, muitas vezes ótima, para instâncias para as quais não se tinha nenhuma solução factível pelo método *branch–and–cut* padrão, no limite de tempo estabelecido.

Para todas as instâncias com mais de seis culturas, a maioria dos algoritmos *branch–and–price* associados com as heurísticas construtivas provou a otimalidade da solução para o PRC–A muito rapidamente. Para classes com seis culturas, situação pouco realista na prática, os algoritmos não foram eficientes na prova de otimalidade da solução, ainda que o *gap* tenha sido pequeno.

Alguns algoritmos tipo *branch–and–cut* foram implementados, mas tiveram um desempenho bem inferior aos *branch–and–price* combinados com a heurística gulosa. Os mais eficientes foram aqueles para os quais limitantes inferiores e/ou superiores obtidos com outros procedimentos foram inseridos no algoritmo. Um trabalho futuro

poderia ser a investigação de desigualdades válidas mais efetivas e combinação com meta–heurísticas tanto para algoritmos *branch–and–cut* quanto *branch–and–price*.

No Capítulo 4 apresentamos o problema PRC–D que consiste em suprir a demanda de um conjunto de hortaliças, tendo-se disponível para o plantio das culturas, áreas heterogêneas nas quais as culturas passíveis de plantio, bem como as suas produtividades, dependem da área considerada. Para suprir a demanda das culturas, as áreas são divididas em lotes e, para cada lote, é obtida uma programação de rotação de culturas como definido no Capítulo 2. O problema é formulado como um modelo de otimização linear, em que cada variável indica o tamanho de área utilizada com uma programação de produção associada a uma programação de rotação. Como o modelo contém possivelmente um número muito grande de programações de produção, no Capítulo 5 foi proposto um algoritmo de geração de colunas para resolvê–lo. Além disto, metodologias de pós-otimização para minimizar o número de lotes na solução foram discutidas. Testes computacionais em instâncias adaptadas de dados reais com até 24 culturas demonstram a eficácia dos métodos propostos.

Para os problemas PRC–A e PRC–D foram otimizados a ocupação das áreas de plantio e produtividade das culturas, respectivamente. Sem alterar a complexidade do modelo matemático e métodos de solução outros diferentes objetivos podem ser otimizados para os problemas. Além disto, certas questões econômicas estão implicitamente consideradas nos modelos mas difíceis de serem mensuradas, como uma possível redução nos custos de produção com a redução ou eliminação de fertilizantes sintéticos e pesticidas, além da produção de alimentos mais saudáveis.

O trabalho apresentado nesta tese tem, certamente, um viés acadêmico. Contudo, as perspectivas de sua aplicabilidade são reais. Com interface de fácil manuseio, a metodologia poderia ser empregada em propriedades agrícolas de base sustentável ou ainda utilizada por centros de pesquisa agropecuária para gerar propostas de ocupação da área de plantio de produtores de regiões com mesmas características climáticas.

Como perspectiva de trabalhos futuros, almejamos aplicar a combinação de heurísticas construtivas e geração de colunas em outras classes de problemas de otimização combinatória. Esta estratégia aumenta a possibilidade de se encontrar uma solução factível para problemas cuja relaxação linear é resolvida por geração de

colunas. Uma meta–heurística também pode ser usada em adição à heurística construtiva a fim de gerar soluções de melhor qualidade. Em uma experiência prévia (Rodrigues et al., 2007) com o problema de coloração de grafos foram obtidos resultados promissores utilizando uma heurística *tabu*. Assim como ocorreu para os problemas PRC–A e de coloração de grafos, temos a expectativa de que esta combinação também possa reduzir o número de colunas geradas e o tempo computacional do algoritmo de geração de colunas para outros problemas. Além disto, se utilizada em alguns (ou todos) nós da árvore de enumeração de algoritmos *branch–and–price*, a estratégia pode reduzir o tamanho da árvore com a possível obtenção de soluções factíveis de boa qualidade mais rapidamente.

# Referências Bibliográficas


[1] Achterberg T, Koch T, Martin A. branching rules revisited. Operations Research Letters, 33, 42-54; 2005.

[2] Altieri MA. Agroecologia: bases científicas para uma agricultura sustentável. Guaíba: Editora Agropecuária; 2002.

[3] Amahuda O, Villalobos JR. Application of planning models in the agri-food supply chain: A review. European Journal of Operational Research 196(1), 1-20; 2009.

[4] Anetts JE, Audsley E. Multiple objective linear programming for environmental farm planning. Journal of the Operational Research Society, 53, 933–943; 2002.

[5] Bachinger J, Zander P. ROTOR, a tool for generating and evaluating crop rotations for organic farming systems. European Journal Agronomy, 26, 130–143; 2006.

[6] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, and Vance PH, Branch and price: Column generation for solving huge integer programs. Operations Research, 46(3), 316–329; 1998.

[7] Campanhola C, Valarini PJ. A agricultura orgânica e seu potencial para o pequeno agricultor. Cadernos de Ciência & Tecnologia, 18(3), 69-101; 2001.

[8] Castellazzi MS, Wood GA, Burgess PJ, Morris J, Conrad KF, Perry JN. A systematic representation of crop rotations. Agricultural Systems, 97, 26–33; 2008.

[9] Clarke, HR. Combinatorial aspects of cropping pattern selection in agriculture. European Journal of Operational Research, 40, 70-77; 1989.

[10] Dantzig GB, Wolfe P. Decomposition principle for linear programs. Operations Research 8, 101-111, 1960.

[11] Desrosiers J, Soumis F, Desrochers M. Routing with time windows by column generation. Networks, 14, 545-565; 1984.

[12] Detlefsen N, Jensen, AL. Modelling optimal crop sequences using network flows. Agricultural Systems 94, 566–572; 2007.

[13] Diestel R. Graph Theory. Graduate Texts in Mathematics, Springer-Verlag; 2000.

[14] Dogliotti S, Rossing WAH, Van Ittersum MK. ROTAT, a tool for systematically generating crop rotations. European Journal of Agronomy, 19, 239–250; 2003.

[15] Dogliotti S, Rossing WAH, Van Ittersum MK. Systematic design and evaluation of crop rotations enhancing soil conservation, soil fertility and farm income: a case study for vegetable farms in South Uruguay. Agricultural Systems, 80, 277–302; 2004.

[16] Ehlers E. Agricultura sustentável: origens e perspectivas de um novo paradigma. Livros da Terra; 1996.

[17] El-Nazer T, McCarl BA. The choice of crop rotation: A modeling approach and case study. American Journal of Agricultural Economics, 68(1), 127–136; 1986.

[18] Farley AA. A note on bounding a class of linear programming problems, including cutting stock problems. Operations Research, 38, 992–993; 1990.

[19] Gilmory PC, Gomory RE. A linear programming approach to the cutting stock problem. Operations Research 9, 848-859; 1961.

[20] Gliessman SR. Agroecologia: processos ecológicos em agricultura sustentável. Porto Alegre: Editora da Universidade – UFRGS; 2000.

[21] Haneveld WK, Stegeman AW. Crop succession requirements in agricultural production planning. European Journal of Operations Research, 166, 406–429; 2005.

[22] Hildreth CG, Reiter S. On the choice of a crop rotation plan. In TC Koopmans (ed.), Proceedings of the Conference on Linear Programming held in Chicago - 1949, 177-188; 1951.

[23] ILOG CPLEX 11.1 Reference Manual. 10.1. Copyright by ILOG, France, 2006.

[24] Johnson EL, Nemhauser GL, Savelsbergh MWP. Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition. INFORMS Journal on Computing, 12(1), 2-23; 2000.

[25] Jones J, Hoogenboom G, Porter C, Boote K, Batchelor W, Hunt L, Wilkens P, Singh U, Gijsman A, Ritchie J. The DSSAT cropping system model. European Journal of Agronomy, 18(3), 235-265; 2003.

[26] Kantorovich L. Mathematical methods of organizing and planning production (translated from a report in Russian, dated 1939). Management Science, 6, 366–422; 1960.

[27] Land AH, Doig, AG. An Automatic Method for Solving Discrete Programming Problems. Econometrica, 28, 497-520; 1960.

[28] Lasdon LS. Optimization Theory for Large Systems. MacMillan, 1970.

[29] Lucas MT Chhajed D. Applications of location analysis in agriculture: a survey. Journal of the Operational Research Society, 55, 561–578; 2004.

[30] Lyon S. Evaluating fair trade consumption: politics, defetishization and producer participation. International Journal of Consumers Studies, 30, 452-464; 2006.

[31] Maculan NF, Passini MM, de Moura Brito JA, Loiseau I. Column-generation in integer linear programming, RAIRO-Operations, Research, 37(2), 67-83; 2003.

[32] Makatouni A. What motivates consumers to buy organic food in the UK ? Results from a qualitative study. British Food Journal, 104, 345-352; 2002.

[33] Marchand H, Martin A, Weismantel R, Wolsey L. Cutting planes in integer and mixed integer programming. Discrete Applied Mathematics, 123, 397-446; 2002.

[34] Nemhauser GL, Wolsey LA. Integer and Combinatorial Optimization, Wiley-Interscience Publication; 1988.

[35] Raidl JR, Puchinger, J. Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization. Studies in Computational Intelligence (SCI) 114, 31–62; 2008.

[36] Rodrigues CD; Santos LMR; Campelo M; Michelon, P. Heuristicaly Generated Columns for the Graph Coloring Problem. Actes de la 9ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision; 2008.

[37] Santos, LMR, Santos, RHS, Arenales, MN, Raggi, LA. Um modelo para a programação de rotações de culturas. Pesquisa Operacional, 27(3), 535-547; 2007.

[38] Santos LMR., Michelon P, Arenales MN, Santos, RHS. Crop rotation scheduling with adjacency constraints. Annals of Operations Research (DOI: 10.1007/s10479-008-0478-z); 2008.

[39] Seyfang G. Ecological citizenship and sustainable consumption: Examining local organic food networks. Journal of Rural Studies, 22, 383–395; 2006.

[40] Stöckle CO, Donatelli M, Nelson R. CropSyst, a cropping systems simulation model. European Journal of Operations Research, 18, 289–307; 2003.

[41] Tillman D, Cassman KG, Matson PA, Naylor R, Polask S. Agricultural sustainability and intensive production practices. Nature, 418, 671–677; 2002.

[42] Vanderbeck F. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research, 48, 111-128; 2000.

[43] Vanderbeck F. Implementing mixed integer column generation. In Column Generation. Ed. Desaulniers G, Desrosiers J and Solomon MM., Kluwer; 2005.

[44] Vanderbeck, F. Branching in branch-and-price: a generic scheme. Research Report Inria-00311274, University Bordeaux 1, France; 2008.

[45] Vandermeer JH. The Ecology of Intercropping. Cambridge University Press; 1989.

[46] Wilhelm W. A Technical Review of Column Generation in Integer Programming. Optimization and Engineering, 2, 159–200; 2001.

[47] Wolsey LA. Integer Programming. Wiley-Interscience; 1998.

# Apêndice A

Para os experimentos computacionais para o problema PRC–A foram selecionadas 28 culturas cultivadas em uma horta de base ecológica em Viçosa – MG. As culturas 25 a 28 são leguminosas para adubação verde. A unidade de tempo é 10 dias. O ciclo de cultivo de cada cultura inclui o tempo estimado para plantio e colheita. Os dados das culturas são exibidos na tabela abaixo.

Tabela A.1. Dados das culturas: nome, família botânica, época apropriada de plantio e ciclo de cultivo.

| **CULTURA** | | **FAMILIA** | | **ÉPOCA PLANTIO** | | **CICLO** |
|---|---|---|---|---|---|---|
| | | | | *Início* | *Fim* | |
| **1** | **Alface** | 1 | *Compositae* | ano todo | | 5 |
| **2** | **Almeirão** | 1 | *Compositae* | ano todo | | 12 |
| **3** | **Couve** | 2 | *Brassicaceae* | março | junho | 10 |
| **4** | **Brócolis** | 2 | *Brassicaceae* | fevereiro | junho | 23 |
| **5** | **Repolho** | 2 | *Brassicaceae* | fevereiro | julho | 13 |
| **6** | **Couve-Flor** | 2 | *Brassicaceae* | março | junho | 14 |
| **7** | **Beterraba** | 3 | *Chenopodiaceae* | março | julho | 7 |
| **8** | **Espinafre** | 3 | *Chenopodiaceae* | fevereiro | julho | 5 |
| **9** | **Abobrinha** | 4 | *Cucurbitaceae* | agosto | março | 14 |
| **10** | **Moranga** | 4 | *Cucurbitaceae* | setembro | janeiro | 15 |
| **11** | **Pepino** | 4 | *Cucurbitaceae* | ano todo | | 14 |
| **12** | **Alho** | 5 | *Liliaceae* | março | abril | 18 |
| **13** | **Cebola** | 5 | *Liliaceae* | março | junho | 12 |
| **14** | **Quiabo** | 6 | *Malvaceae* | agosto | março | 23 |
| **15** | **Milho** | 7 | *Gramineae* | agosto | abril | 10 |
| **16** | **Aveia** | 7 | *Gramineae* | março | maio | 18 |
| **17** | **Tomate** | 8 | *Solanaceae* | ano todo | | 15 |
| **18** | **Pimentão** | 8 | *Solanaceae* | ano todo | | 16 |
| **19** | **Jiló** | 8 | *Solanaceae* | setembro | fevereiro | 19 |
| **20** | **Cenoura** | 9 | *Umbelliferae* | março | julho | 14 |
| **21** | **Salsinha** | 9 | *Umbelliferae* | setembro | março | 19 |
| **22** | **Feijão Vagem** | 10 | *Leguminosae* | agosto | abril | 11 |
| **23** | **Ervilha** | 10 | *Leguminosae* | março | abril | 9 |
| **24** | **Feijão** | 10 | *Leguminosae* | agosto | setembro | 9 |
| **25** | **Crotalária** | 10 | *Leguminosae* | setembro | dezembro | 9 |
| **26** | **Feijão-de-Porco** | 10 | *Leguminosae* | setembro | dezembro | 8 |
| **27** | **Mucuna** | 10 | *Leguminosae* | setembro | janeiro | 11 |
| **28** | **Ervilha Peluda** | 10 | *Leguminosae* | março | junho | 14 |

# Apêndice B

Para os experimentos computacionais para o modelo PRC–D foram selecionadas 26 culturas cultivadas em uma horta orgânica em Barbacena – MG. As culturas 22 a 26 são leguminosas para adubação verde. Os dados das culturas são apresentados nas Tabelas B.1 e B.2 e a unidade de tempo é uma semana. O ciclo de cultivo de cada cultura inclui o tempo estimado para plantio e colheita.

Tabela B.1. Dados das 24 culturas: nome, família botânica, época apropriada de plantio e ciclo de cultivo.

| | CULTURA | | FAMILIA | ÉPOCA PLANTIO *Início* | *Fim* | CICLO |
|---|---|---|---|---|---|---|
| **1** | **Alface americana** | 1 | *Compositae* | ano todo | | 7 |
| **2** | **Alface mimosa** | 1 | *Compositae* | ano todo | | 7 |
| **3** | **Alface lisa** | 1 | *Compositae* | ano todo | | 7 |
| **4** | **Almeirão** | 1 | *Compositae* | ano todo | | 7 |
| **5** | **Agrião** | 2 | *Brassicaceae* | fevereiro | julho | 32 |
| **6** | **Couve** | 2 | *Brassicaceae* | fevereiro | setembro | 32 |
| **7** | **Brócolis** | 2 | *Brassicaceae* | fevereiro | outubro | 20 |
| **8** | **Couve-Flor** | 2 | *Brassicaceae* | março | outubro | 18 |
| **9** | **Beterraba** | 3 | *Chenopodiaceae* | fevereiro | setembro | 11 |
| **10** | **Espinafre** | 3 | *Chenopodiaceae* | fevereiro | setembro | 20 |
| **11** | **Abobrinha** | 4 | *Cucurbitaceae* | outubro | fevereiro | 14 |
| **12** | **Moranga** | 4 | *Cucurbitaceae* | novembro | janeiro | 19 |
| **13** | **Pepino** | 4 | *Cucurbitaceae* | setembro | março | 13 |
| **14** | **Alho** | 5 | *Liliaceae* | março | abril | 24 |
| **15** | **Cebola** | 5 | *Liliaceae* | março | julho | 24 |
| **16** | **Alho porró** | 5 | *Liliaceae* | abril | abril | 12 |
| **17** | **Quiabo** | 6 | *Malvaceae* | novembro | janeiro | 27 |
| **18** | **Tomate** | 7 | *Solanaceae* | ano todo | | 24 |
| **19** | **Cenoura** | 8 | *Umbelliferae* | ano todo | | 16 |
| **20** | **Salsinha** | 8 | *Umbelliferae* | outubro | fevereiro | 21 |
| **21** | **Feijão** | 9 | *Leguminosae* | outubro | fevereiro | 12 |
| **22** | **Mucuna preta** | 9 | *Leguminosae* | outubro | janeiro | 16 |
| **23** | **Feijão-de-porco** | 9 | *Leguminosae* | outubro | fevereiro | 12 |
| **24** | **Tremoço** | 9 | *Leguminosae* | março | julho | 18 |
| **25** | **Ervilha Peluda** | 9 | *Leguminosae* | março | julho | 20 |
| **26** | **Crotalária** | 9 | *Leguminosae* | outubro | fevereiro | 16 |

Tabela B.2. Dados sobre 21 culturas: nome, unidade de medida, número de períodos (semanas) entre o plantio e a primeira colheita ($o_i$) e a produção esperada nos sucessivos períodos de colheita.

| | **CULTURA** | ***unidade de medida*** | | **COLHEITA** | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $o_i$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **1** | **Alface americano** | *unidade* | 5 | 9 | 3 | | | | | | | | | | | | | | | | | | |
| **2** | **Alface mimosa** | *unidade* | 5 | 9 | 3 | | | | | | | | | | | | | | | | | | |
| **3** | **Alface lisa** | *unidade* | 5 | 9 | 3 | | | | | | | | | | | | | | | | | | |
| **4** | **Almeirão** | *unidade* | 5 | 9 | 3 | | | | | | | | | | | | | | | | | | |
| **5** | **Agrião** | *maço* | 12 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| **6** | **Couve** | *maço* | 12 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| **7** | **Brócolis** | *maço* | 10 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 2 | 2 | 1 | | | | | | | | | | |
| **8** | **Couve-Flor** | *unidade* | 16 | 1 | 3 | | | | | | | | | | | | | | | | | | |
| **9** | **Beterraba** | *kg* | 8 | 1 | 2 | 1 | | | | | | | | | | | | | | | | | |
| **10** | **Espinafre** | *maço* | 5 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 2 | | | | | |
| **11** | **Abobrinha** | *kg* | 9 | 0,2 | 0,3 | 0,3 | 0,3 | 0,2 | | | | | | | | | | | | | | | |
| **12** | **Moranga** | *kg* | 16 | 0,5 | 0,5 | 0,5 | | | | | | | | | | | | | | | | | |
| **13** | **Pepino** | *kg* | 8 | 1 | 2 | 2 | 1 | 1 | | | | | | | | | | | | | | | |
| **14** | **Alho** | *kg* | 23 | 0,3 | | | | | | | | | | | | | | | | | | | |
| **15** | **Cebola** | *kg* | 23 | 3 | | | | | | | | | | | | | | | | | | | |
| **16** | **Alho porró** | *kg* | 8 | 3 | 3 | 3 | 3 | | | | | | | | | | | | | | | | |
| **17** | **Quiabo** | *kg* | 14 | 0,2 | 0,3 | 0,3 | 0,4 | 0,4 | 0,4 | 0,4 | 0,4 | 0,4 | 0,4 | 0,2 | 0,2 | 0,2 | | | | | | | |
| **18** | **Tomate** | *kg* | 15 | 0,8 | 0,8 | 1 | 1 | 1,2 | 1,2 | 1 | 0,8 | 0,2 | | | | | | | | | | | |
| **19** | **Cenoura** | *kg* | 13 | 1,5 | 2 | 1,5 | | | | | | | | | | | | | | | | | |
| **20** | **Salsinha** | *maço* | 8 | 14 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 14 | 10 | | | | | | | |
| **21** | **Feijão** | *kg* | 11 | 0,3 | | | | | | | | | | | | | | | | | | | |