UNIVERSIDADE FEDERAL DO CEARÁ
UNIVERSITÉ D'AVIGNON

TESE DE DOUTORADO

Carlos Diego RODRIGUES

Abordagens híbridas na solução de problemas de programação inteira da teoria e prática

29 de Junho de 2010

UNIVERSIDADE FEDERAL DO CEARÁ
UNIVERSITÉ D'AVIGNON

TESE DE DOUTORADO

para a obtenção do título de

Doutor em Ciências da Computação

da Universidade Federal do Ceará

e

Université d'Avignon

Especialidade: CIÊNCIAS DA COMPUTAÇÃO

Defendida por

Carlos Diego RODRIGUES

Abordagens híbridas na solução de problemas de programação inteira da teoria e prática

Orientadores:

Philippe MICHELON

Manoel BEZERRA CAMPÊLO NETO

defendida em 29 de Junho de 2010 em Fortaleza - Ceará

Banca de juri :

Revisores:	Nelson MACULAN	- COPPE/UFRJ
	Luiz SATORU OCHI	- IC/UFF
Orientadores:	Philippe MICHELON	- LIA/UAPV
	Manoel B. CAMPÊLO NETO	- MDCC/UFC
Examinadores:	Ricardo CORRÊA	- MDCC/UFC
	Plácido Rogério PINHEIRO	- UNIFOR
	Dominique QUADRI	- LIA/UAPV

ABORDAGENS HÍBRIDAS PARA PROBLEMAS DE PROGRAMAÇÃO INTEIRA DA TEORIA E DA PRÁTICA

Carlos Diego Rodrigues

Tese apresentada ao Curso de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como parte dos Requisitos para obtenção do Grau de Doutor em Ciência da Computação.

Composição da Banca Examinadora:

Prof. Dr. Manoel Bezerra Campêlo Neto (Presidente/Orientador)

Prof. Dr. Philippe Michelon (Orientador)

Prof. Dr. Ricardo Cordeiro Corrêa

Prof. Dr. Dominique Quadri

Prof. Dr.Nelson Maculan Filho

Prof. Dr. Luiz Satoru Ochi

Prof. Dr. Plácido Rogério Pinheiro

Fortaleza, 29 de junho de 2010

Agradecimentos

Antes de agradecer expresso aqui a dificuldade de escrever esta seção de minha tese. Essa dificuldade vem sobretudo da certeza de que esqucerei de agradecer algumas pessoas e da incerteza de que essas pessoas me desculparão. Aos omitidos no momento árduo da redação, presto meus primeiros agradecimentos.

Gostaria de prestar os meus mais sinceros agradecimentos a todos os que tornaram este trabalho possível. Primeiramente gostaria de agradecer àqueles que a tornaram possível cientificamente. Aos meus orientadores Philippe Michelon (e, evidentemente, sua família), que sempre me apoiou com idéias e com sua experiência científica, além da riqueza que ele me proporcionou no plano humano e Manoel Campêlo por todos os seus preciosos conselhos e rigor científico invejável. Não poderia deixar de citar também Dominique Quadri que, embora não seja oficialmente, foi de fato minha orientadora na segunda metade desta tese, trazendo sempre a motivação, a crítica, o acompanhamento que essa funcão pede, além de todo o incentivo à capacidade.

Agradeço igualmente à todos os que contribuiram para a minha formação científica, dentre os quais posso nomear os professores Cláudia Linhares e Ricardo Corrêa, criadores do grupo científico ParGO na Universidade Federal do Ceará. Eles foram os semeadores de muitos projetos bem sucedidos, para quais teria a honra de dar minha contribuição. A todo o pessoal da Universidade Federal do Ceará, professores e funcionários, que apesar de todas as dificuldades estruturais fazem proporcionam uma formação de alta qualidade acadêmica e humana.

Os membros exteriores que compõem a banca de defesa desta tese não poderiam deixar de se sentir agradecido. Aos professores Nelson Maculan, Luis Satoru Ochi, Plácido Rogério Pinheiro meu muito obrigado pela disposição e pela crítica científica trazida.

Lembro também daqueles que o fizeram em nível estrutural: ao projeto ALFA e à CAPES através do Colégio Doutoral Franco Brasileiro que me financiaram, assim como todo o pessoal envolvido na gestão desses recursos, em especial à Lorena Pardeñas, coordenadora do projeto ALFA. Igualmente gostaria de agradecer ao pessoal do "*Laboratoire d'Informatique d'Avignon*"que me acolheram e me proporcionaram condições de trabalho de alta qualidade para meu desempenho, em especial às secretárias Simone Mouzac e Jocelyne Gouret. Ao pessoal da "*Université d'Avignon*", simbolizado em minha mente pelo "Service des Relations Internationales"que foram capazes de aprimorar meu nível em francês, assim como o de minha esposa, Simone.

E, certamente, entre os meus próximos, não poderia deixar de agradecer, em primeiro lugar, minha esposa Simone, minha companheira ao longo dos anos, por todo o apoio, paciência e dedicação que só o verdadeiro amor seria capaz de nutrir e por tudo que ela me ensinou. Toda minha família, que apesar de estar longe na maior parte do tempo, me deu tudo o que eu precisava no sentido afetivo que é certamente muito importante nesse percurso. Minha mãe Helena que moldou minha personalidade e meu caráter, meus irmãos Filho, Filipe e o irmãozinho que ganhei durante essa caminhada, Vitor, irmãos com quem eu, prazeirosamente, fui acostumado a dividir todas as minhas conquistas e, por

que não mais essa. Agradeço também meu pai, Carlos, que é certamente minha primeira inspiração científica. Agradeço ao Meneleu, que sempre me mostrou o quanto a educação e a cultura são importantes. Agradeço aos meus sogros, que são grandes exemplos de amor e de ternura. A minha grande amiga Rosália, por toda sua dedicação. Agradeço também uma criaturinha que me foi muito especial e carinhosa e que me deixou uma grande lição, o Gorki.

Agradeço a todos os meus parceiros científicos ao longo de minha tese, mas em especial, meu muito obrigado, também, a todos os colegas que mostraram paciência, compreensão e apoio durante esses anos de tese. Agradeço a Marie-Jean, com quem dividi amigavelmente a sala ao longo desses quase quatro anos de tese, em que vimos passar diversas outras pessoas também passíveis de agradecimento. Ao Boris por todo seu apoio, bom humor cotidianos. Ao Sylvain, com quem tive oportunidade de trabalhar também. Enfim, a todos com quem mantive um bom contato dento do LIA, meus agradecimentos.

Finalmente, mais uma vez na certeza de ter omitido pessoas importantes, agradeço à todos os que foram capazes de sorrir para mim ao me encontrarem ou me ouvirem nesses últimos quatro anos.

Muito obrigado!

Abordagens híbridas na solução de problemas de programação inteira da teoria e prática.

Resumo:

O objetivo principal dessa tese é mostrar como a hibridização de métodos de áreas relacionadas nos permite avançar na resolução de problemas difíceis da Programação Matemática.

Para este fim, nós aplicamos técnicas que combinam a Programação Matemática com outras teorias, especificamente a Programação por Restrições e Teoria dos Jogos, e aplicamos à problemas clássicos da teoria (Problema das Mochilas Múltiplas e Problema da Mochila Quadrática) e a um problema prático (Busca de um alvo inteligente).

Para o estudo dos problemas teóricos partimos do Problema da Mochila, clássico problema NP-difícil da literatura que possui soluções eficazes. No entanto, as variantes que estudamos, o Problema das Mochilas Múltiplas e o Problema da Mochila Quadrático, mostram-se desafiadoras mesmo para instâncias relativamente pequenas. Tais problemas interessam à área por serem a base de muitos outros problemas de Programação Matemática.

Já para o estudo prático, partimos de um problema clássico da Teoria dos Jogos, o problema de busca de um alvo móvel e inteligente. Para resolvê-lo, modelamos tal problema como um problema de programação linear 0-1. Essa abordagem inovadora permite uma resolução do problema por meio de simulação.

Para cada um dos problemas fizemos experimentos computacionais que atestam o desempenho das técnicas propostas e os resultados foram comparados aos melhores resultados conhecidos em cada um dos casos.

Podemos atestar, seguindo diversas referências na área, que as técnicas híbridas proveem um importante conjunto de ferramentas matemáticas para a solução de problemas de diversos domínios.

Palavras-chave: Programação Matemática, Programação por Restrições, Jogos de Busca, Métodos Híbridos

Approches hybrides dans la résolution de problèmes de théorie et de la pratique.

Résumé: Le principal objectif de cette thèse est de montrer l'avantage que présente l'utilisation de techniques multidisciplinaires pour la résolution de problèmes difficiles de la Programmation Mathématique.

Pour cela, nous utilisons des techniques hybrides mêlant la Programmation Mathématique et d'autres théories comme la Programmation par Contraintes et la Théorie des Jeux. Puis nous appliquons ces techniques à des problèmes classiques, disons aussi académiques (Problème du Sac-à-dos Multidimensionnel et Sac-à-dos Quadratique) et, également, à un problème pratique (la recherche d'une cible intelligente).

Les problèmes académiques étudiés sont liés au problème classique du sac-à-dos pour lequel, malgré son caractère NP-difficile, il existe des méthodes exactes de résolution. Ses généralisations, le Problème du Sac-à-dos Multidimensionnel et le Problème du Sac-à-dos Quadratique se montrent en revanche plus hardues à résoudre, même pour des instances de petite taille.

L'étude pratique est basée sur un problème classique de la Théorie des Jeux, le problème de la recherche d'une cible mobile et intelligente. Pour le résoudre nous le modélisons comme une Problème de Programmation Linéaire en Nombres Entiers 0-1. Cette nouvelle approche permet une solution du problème au moyen de la simulation statistique.

Des expérimentations numériques sont menées pour chaque problème traité et démontrent la performance des techniques développées en comparaison avec les résultats de la littérature.

Ces résultats nous amènent à penser que les techniques multidisciplinaires constituent très certainement un outil efficace pour la résolution de problèmes issus d'horizons divers.

Mots-clés: Programmation Mathématique, Programmation par Contraintes, Théorie de la Recherche, Methodes Hybrides.

Conteúdo

Capítulo 1
Introdução

A resolução de problemas de otimização combinatória é o principal objetivo de diversas disciplinas da Pequisa Operacional. Ao longo dos anos, diversas técnicas têm sido desenvolvidas separadamente visando o mesmo objetivo. Tradicionalmente as técnicas aplicadas com sucesso a uma dada classe de problemas pertencem a um mesmo ramo e, logo, essa classe de problemas é absorvida para este ramo.

No entanto, sabemos que a multidisciplinaridade tem se mostrado, sucessivamente, a melhor maneira de resolver diversos problemas como pode ser visto em [Milano 2004, Hooker 1999b, Nisan 2007]. Neste trabalho apresentamos três aplicações multidisciplinares a problemas de otimização combinatória. Mais precisamente, em cada caso, fazemos um estudo de um problema abordado tradicionalmente dentro de um domínio e agregamos técnicas de um outro domínio para uma resolução bem sucedida do problema.

Em primeiro lugar, no Capítulo 2, apresentamos o tradicional o problema das mochilas múltiplas (MKP). Esse problema constitui uma generalização de um dos mais estudados problemas da programação matemática, que apresenta um número arbitrário de restrições de capacidade. Com efeito, o MKP representa para os pesquisadores da área um desafio, pois para este problema não existe uma estrutura particular que possa ser explorada. Além disso, são conhecidas instâncias ([Beasley 1990]) para o problema que, apesar de um número relativamente pequeno de variáveis e restrições, são muito difíceis de se resolver. Como exemplo, podemos dizer que não se conhece até hoje soluções ótimas para instâncias com 250 variáveis e 30 restrições. Para referência, expomos o modelo do MKP a seguir:

$$
\begin{aligned}
\text{(MKP)} \quad \max \quad & \sum_{j=1}^{n} c_j x_j \\
\text{s.a:} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\
& x_j \in \{0, 1\}, \quad j = 1, ..., n
\end{aligned}
$$

onde $b_i \geq 0, c_j \geq 0, a_{ij} \geq 0$ para todo $1 \leq i \leq m$ e todo $1 \leq j \leq n$.

Apesar da dificuldade apresentada por esse problema, desenvolvemos um método cooperativo entre programação matemática e programação por restrições capaz de fornecer os melhores resultados, em termos de tempo computacional, para o problema. Nosso método baseia-se em um esquema cooperativo apresentado por [Oliva 2001]. Tal esquema é baseado na execução simultânea de técnicas advindas das duas disciplinas e na troca de

informações entre essas técnicas. Na ocasião, os autores mostraram essa interação através da filtragem da restrição de custos reduzidos.

Para esse esquema os autores em [Boussier 2009] desenvolveram uma enumeração capaz de explorar ainda mais a cooperação entre os dois paradigmas. Esse esquema enumerativo faz uma primeira divisão do problema através da interseção do conjunto viável com diversos hiperplanos de cardinalidade fixa. Em seguida, para cada um dos hiperplanos, a enumeração ramifica de acordo com os custos reduzidos das variáveis não básicas, de forma a explorar o máximo a filtragem da restrição deduzida através desses coeficientes.

Aqui damos duas novas contribuições. Primeiramente definimos uma nova restrição global, chamada de *Knapsum*, que leva em consideração uma restrição de cardinalidade e uma restrição de capacidade e/ou uma restrição de empacotamento, além da restrição de integralidade. Essa restrição global, bem como a sua filtragem, podem ser aplicadas a quaisquer problemas que possuam essa subestrutura. Mostramos ainda que cortes podem ser derivados a partir dessa filtragem. Finalmente um processo de agregação das filtragens *Knapsum* é capaz de gerar um novo método ligeiramente mais eficiente, segundo os experimentos computacionais. Em segundo lugar, fizemos uma atualização da enumeração apresentada por [Boussier 2009] para levar em conta as restrições *Knapsum*, bem como fazer um melhor percurso da árvore de busca.

O segundo problema considerado neste trabalho é estudado no Capítulo 3. Trata-se do problema essencial de programação quadrática, o problema da mochila quadrático (QKP), introduzido por [Gallo 1980]. Esse problema é uma outra generalização do problema da mochila, utilizado para modelar casos onde a interação binária entre os objetos também constitui um critério de seleção dos mesmos. Isso significa que existe, além do fator linear na função objetivo, um termo quatrático. Dentre as diversas maneiras de tratar este problema encontram-se as técnicas de linearização [Sherali 1999, Chaovalitwongse 2004, Gueye 2009]. O QKP é descrito como:

$$
\begin{aligned}
\text{(QKP)} \quad \max \quad & \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \sum_{i=1}^{j-1} q_{ij} x_i x_j \\
\text{s.a:} \quad & \sum_{j=1}^{n} a_j x_j \leq b, \\
& x_j \in \{0,1\}, \qquad\qquad j = 1, ..., n
\end{aligned}
$$

onde $c_j, a_j, q_{ij}, b \geq 0$, para todo $1 \leq i \leq n$, $1 \leq j \leq n$.

A linearização de uma expressão quadrática em um modelo de Programação Matemática consiste na substituição da mesma por um conjunto de variáveis e restrições relacionadas que fazem o mesmo papel. Por exemplo, a linearização clássica consiste na substituição de cada produto de variáveis $x_i x_j$ que aparece no termo quadrático por uma nova variável z_{ij}. Essa variável está sujeita a três restrições que garantem o comportamento da variável z_{ij} como o produto das variáveis binárias $x_i x_j$. Tal técnica, apesar de

bem difundida e, evidentemente aprimorada, encontrou rapidamente um limite devido ao grande número ($\Omega(n^2)$) de variáveis e restrições adicionais.

Em nosso trabalho retomamos a linearização de uma função quadrática de uma maneira original para o QKP. Com efeito, o termo quadrático da função objetivo é substituído por uma única variável t, que dá o nome da técnica como t-linearização. Essa variável, representante do valor quadrático da solução, deve ser controlada através de restrições que limitam seu valor ao termo quadrático original. Mostramos que o poliedro representando o valor do termo quadrático original é equivalente a um poliedro linear, embora este poliedro contenha um número exponencial de restrições.

A partir dessa caracterização, temos portanto um problema linear equivalente, mas que contém um número muito grande de restrições. Quando esta situação acontece, temos, na programação matemática, o recurso da geração de restrições. No entanto, para que a geração de restrições seja aplicada com sucesso, devemos resolver um problema de separação, ou seja, a partir de uma solução qualquer para o problema determinarmos se existe uma restrição dentro do conjunto de restrições a ser gerado que é violada por aquela solução. Se este for o caso, precisamos determinar qual é esta restrição. Para a caracterização do poliedro apresentada, mostramos que o problema de separação pode ser resolvido em tempo polinomial.

A partir desses elementos, interessamo-nos na aplicação desse esquema para o QKP. Fomos capazes de estabelecer um novo limite superior para o problema através da técnica de linearização apresentada, combinada à relaxação linear do problema. Mostramos que esse limite, embora não seja melhor que o limite apresentado na literatura (devido a [Billionnet 1999]), continua bem próximo do valor ótimo do problema. Além disso mostramos maneiras através das quais o limite pode ser reforçado: seja através do fortalecimento das restrições geradas, tomando em consideração a restrição da mochila, ou através da reintegração da restrição de integralidade sobre as variáveis.

Esse limite superior deu a luz um método enumerativo para a resolução do QKP semelhante ao método apresentado para o MKP, posto que temos em mãos agora um problema que se assemelha àquele apresentado no Capítulo 2. Aplicamos para o QKP, portanto uma camada de programação por restrições, filtrando três restrições: custos reduzidos, limite inferior e restrição da mochila.

O método apresentado para o QKP é bastante eficiente para os problemas esparsos, ou seja, problemas cuja matriz de coeficientes quadráticos é esparsa. Esta é uma característica comum das técnicas de linearização, visto que elas são aproximações do poliedro quadrático. Quando comparado à [Pisinger 2005], que apresenta o melhor método para a resolução do QKP, nosso método é superior para instâncias de pequeno tamanho assim como instâncias de densidade baixa. A técnica apresentada é original e há um grande espaço de aplicações dentro da programação quadrática.

Ambos os problemas descritos acima tratam-se de problemas essenciais de seus domínios. A resolução desses problemas mostra mais que um bom método específico para o problema, mas também um bom método para toda a área.

A última parte da presente tese, encontrada no Capítulo 4, tem por fim o estudo de

um problema prático. Trata-se da busca de um alvo inteligente. Essa classe de problemas consiste num duelo entre duas entidades: um alvo e um pesquisador. O objetivo do alvo é permanecer escondido, enquanto que o do pesquisador consiste na elaboração de um plano de pesquisa, dentro de uma zona pré-estabelecida com e um horizonte de tempo fixo, para encontrar o alvo.

Esse problema nasceu junto com a pesquisa operacional durante a segunda guerra mundial e, especialmente, no pós-guerra, quando os países buscavam maneiras de se defender de ataques de submarinos espiões (ver [Koopman 1956b, Koopman 1956a, Koopman 1957]). Historicamente o problema é tratado como um jogo de soma nula: o pesquisador maximiza a probabilidade de encontrar o alvo, enquanto este a minimiza.

Encontramos na literatura uma sequência de modelos para o problema ([Koopman 1956b, Koopman 1956a, Koopman 1957, Kan 1977, Brown 1980, Thomas 1991, Hohzaki 1999, Hohzaki 2000, Hohzaki 2006]) que evoluiram de acordo com a complexidade do problema, caracterizada por diversos items que constituem a sua definição (mobilidade do alvo, espaço de busca, cooperatividade ou adversidade, objetivo do alvo, inteligência do alvo, etc). Porém, os modelos em Teoria dos Jogos apresentam duas principais limitações. Ou esses modelos tratam apenas de um caso local (para uma decisão instantânea, mas não para todo o plano de busca) ou são modelos com um número muito grande de variáveis e restrições. Além disso, esses modelos não são capazes de dar uma resposta clara ou concreta ao problema – a solução fornecida é a distribuição do esforço de pesquisa e não o plano de busca. Isso se deve ao fato de que a combinatória intríseca ao problema é desconsiderada e apenas uma relaxação linear do modelo é resolvida.

Ao trabalharmos em parceria com o Ministério da Defesa francês, através do órgão DGATn ("Direction Générale de l'Armée - Technique navale"), buscamos, no entanto, modelos que possam ser usados na prática, ou seja, que possam dar uma resposta precisa ao problema e cuja aplicabilidade permita uma resposta em tempo de cálculo razoável.

A resolução do problema passou, portanto, pela elaboração de um modelo original de programação matemática capaz de modelar uma grande parte das variantes do problemas encontradas na literatura. O modelo apresentado é baseado na discretização espaço-temporal e na simulação do comportamento do alvo através de uma amostra determinística das possibilidades para este.

Mostramos que o número de alvos necessários para obter uma confiança estatística comparável a modelos considerados confiáveis é muito grande e que esse número acarreta para nosso modelo uma explosão na quantidade de variáveis e restrições. Assim, partimos para uma decomposição do modelo através do método de janelas deslizantes, que possui um histórico favorável ([Brown 1980]).

Trabalhamos em nossos experimentos com instâncias reais fornecidas pela DGATn. Para essas instâncias verificamos que o modelo foi capaz de fornecer soluções com um número de alvos reduzido. Para verificarmos se o modelo fornecia soluções confiáveis, apesar do reduzido número de alvos, desenvolvemos um simulador, capaz de verificar, com uma confiança arbitrariamente grande e uma margem de erro suficientemente pequena,

qual a real avaliação de uma solução.

Mostramos que, apesar de uma grande variância nos resultados, o modelo que desenvolvemos é capaz de dar, em média, uma solução com uma margem de erro inferior a 10% em tempo razoável. A qualidade dos resultados foi validade pela DGATn e é superior ao método atualmente aplicado pelo órgão.

Capítulo 2

Problema das Mochilas Múltiplas

Sumário do capítulo

RESUMO

Neste capítulo apresentamos um esquema colaborativo entre a Programação Matemática e a Programação por Restrições. Tal esquema consiste na utilização de técnicas de ambos os paradigmas para a resolução de um problema de otimização, possibilitando, desta maneira, a complementaridade entre eles. Para ilustrar o potencial deste esquema, mostramos sua aplicação ao Problema das Mochilas Múltiplas, problema este que não possui uma estrutura particular e que aparece como um subproblema de diversos outros problemas de otimização.

Além disso propomos uma nova restrição global, a qual chamamos de *Knapsum*. Essa restrição, definida por uma restrição de capacidade acoplada a uma restrição de cardinalidade, possui uma filtragem bastante simples e, segundo nossos experimentos computacionais, bastante eficaz. Essa filtragem, em conjunto com a filtragem da restrição de custos reduzidos constitui a camada de Programação por Restrições de nosso método. Mostramos ainda duas extensões na utilização de restrições *Knapsum*: a dedução de cortes para o MKP e a agregação de restrições *Knapsum*.

Em conjunto com um esquema de enumeração concebido especialmente para maximizar a eficiência das filtragens, nosso método se provou o mais eficaz na resolução do MKP, comparado ao programa comercial IBM/ILOG CPLEX e ao melhor método publicado na literatura ([Boussier 2008, Boussier 2009]).

2.1 Introdução

A colaboração entre a Programação Matemática (PM) e a Programação por Restrições (PPR) é um assunto que vem tomando foco nos últimos anos. A impossibilidade de resolver grandes instâncias de problemas difíceis e a aparente complementaridade entre os dois paradigmas estimulam a pesquisa sobre a cooperação entre eles. O presente documento se insere nesse contexto e busca, a partir de um problema recorrente, que aparece em diversos outros como subproblema, estabelecer um método de cooperação entre a PPR e a PM.

2.1.1 O Problema das Mochilas Múltiplas

O problema da mochila é um problema clássico da literatura em otimização combinatória. É amplamente conhecido devido à sua facilidade de caracterização e formulação. As aplicações do "Knapsack problem", como também é conhecido, variam desde a criptografia até problemas práticos de empacotamento de produtos.

Podemos definir tal problema da seguinte forma: dada uma mochila de capacidade b, e um conjunto de n itens $\{1, ..., n\}$, onde cada item possui um peso a_j e um valor c_j, devemos escolher um conjunto de itens de valor máximo de tal forma que não ultrapasse a capacidade da mochila. Mais formalmente, como um problema de programação matemática, a seguinte formulação o caracteriza:

$$\begin{aligned} \text{(KP)} \quad \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.a:} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0,1\},\ j = 1, ..., n \end{aligned}$$

onde $b \geq 0, c_j \geq 0, a_j \geq 0$ para todo $1 \leq j \leq n$. Ao longo dos anos diversas técnicas sobre o problema foram desenvolvidas e aprimoradas, sendo que uma solução por programação dinâmica consegue bons resultados em tempo pseudo-polinomial. No entanto, nenhuma solução em tempo polinomial é conhecida para o problema (lembrando que esse é um problema NP-difícil).

Existem também diversas generalizações para o problema da mochila, que também pertencem à classe dos problemas NP-difíceis. Como exemplo, podemos citar o problema da mochila com repetição, onde é possível escolher um item mais de uma vez, mas um número limitado de vezes; o problema da mochila múltipla escolha, onde cada item deve ser escolhido dentro de uma classe específica de itens, e o problema das mochilas múltiplas, onde ao invés de uma única restrição de capacidade, o problema apresenta diversas. Esse último caso, também chamado de "Multiple Knapsack Problem"ou ainda "Multiple Con-

strained Knapsack Problem"(MKP) é o objeto de estudo deste Capítulo. A formulação do problema aparece a seguir:

$$\begin{aligned} \text{(MKP)} \quad \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.a:} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\ & x_j \in \{0, 1\}, \quad j = 1, ..., n \end{aligned}$$

onde $b \leq 0, c_j \geq 0, a_{ij} \geq 0$ para todo $1 \leq i \leq n$ e todo $1 \leq j \leq n$.

Interessante notar que apesar de o (MKP) ser uma extensão do problema da mochila, não existe nenhuma estrutura a qual se possa explorar e, portanto, mesmo problemas pequenos são de difícil solução. Apesar de serem conhecidas boas heurísticas para o problema, provar que uma solução é ótima ainda pode ser uma tarefa difícil, mesmo para pequenas instâncias.

2.1.2 Esquemas colaborativos

Nos últimos anos, pesquisadores têm buscado fora do âmbito da programação matemática soluções para problemas característicos da área [Hooker 1999a, Osorio 2001, Oliva 2001]. Entre as alternativas complementares se encontra a Programação por Restrições. Isso mostra uma tendência na qual este trabalho se insere. Hooker e Osorio [Hooker 1999a] desenvolveram um paradigma que busca integrar a implicação lógica aos modelos de programação linear ou inteira. Hooker [Hooker 2002] mostrou que em diversos pontos a otimização combinatória poderia se beneficiar da redução de domínio das variáveis, idéia base da programação por restrições.

Mais tarde, e ainda mais aplicado ao problema da mochila, Osorio e Glover [Osorio 2001] aplicaram a idéia de cortes lógicos, ou seja, cortes que eram baseados em implicações do conjunto de restrições, ao invés dos tradicionais métodos da programação matemática. Em [Osorio 2002], os autores aplicaram além dos cortes lógicos uma filtragem na raiz do problema para fixar variáveis de acordo com seus custos reduzidos, técnica que foi aprimorada em [Oliva 2001] e utilizamos neste trabalho.

Recentemente, novas técnicas se mostraram eficientes para o (MKP), como por exemplo em [Vasquez 2001], o autor mostra que saber quantas variáveis são iguais a 1 na solução ótima torna o problema mais fácil de ser tratado. Em [Glover 2008], os autores também usam esse fato para derivar novos cortes do tipo *cover cuts*, usando essa informação extra. Em nosso trabalho, também vamos usar essa informação para ajudar na resolução do método e avançar sobre a combinação entre a programação por restrições e a programação matemática.

Por fim, baseando-se nos últimos trabalhos na área [Vimont 2008] e [Boussier 2009], procuramos integrar novas técnicas de programação por restrições e aprimorar o algoritmo

de enumeração ímplicita proposto para alcançar melhores resultados.

2.1.3 Estrutura deste texto

Iremos apresentar nas seções seguintes os principais aspectos que compõem este Capítulo. Na Seção 2.2 iremos mostrar quais as técnicas de filtragens desenvolvidas e aplicadas ao problema das múltiplas mochilas. Em seguida, na Seção 2.3, mostraremos quais as especificidades que apresenta o processo de *branch-and-bound* quando aplicado ao problema em questão. Na Seção 2.4, alguns pontos adicionais são mostrados na tentativa de acelerar o processo computacional, bem como um aprofundamento da PPR via programação dinâmica. A Seção 2.6 mostra os resultados obtidos quando os métodos das seções anteriores são aplicados, bem como uma comparação entre o método desenvolvido e o principal código comercial aplicado diretamente ao problema (ILOG CPLEX). Por fim, apresentamos uma conclusão e as perspectivas para a continuação do trabalho na Seção 2.7.

2.2 Programação por Restrições e o MKP

A programação por restrições é um paradigma de programação que procura estabelecer relação entre as variáveis do problema sob forma de restrições. É amplamente aplicada na teoria a diversos problemas clássicos da otimização combinatória, mas teve suas raízes em aplicações principalmente derivada da lógica matemática. A resolução de problemas via programação por restrições consiste na redução do domínio das variáveis, devido a aplicações sucessivas de processos de *filtragens*. Em outras palavras, essas rotinas eliminam dos valores possíveis para uma variável (domínio) aqueles que, devido às relações estabelecidas pelas restrições, não são realizáveis.

As *filtragens* podem ser definidas como uma dedução matemática feita a partir de uma (ou mais) restrição(ões) sobre o domínio de uma (ou mais) variável(is). A idéia central sobre o trabalho é usar as informações cedidas por um processo de resolução via programação matemática para alimentar as filtragens sobre as restrições. Estas, por sua vez, vão (pelo menos é o que esperamos) reduzir o dominio das variáveis (binárias) e portanto diminuir o tamanho do problema.

2.2.1 Custos reduzidos

O primeiro tipo de filtragem que vamos apresentar se baseia numa avaliação realizada sobre os custos reduzidos das variáveis do problema obtidos após a resolução da relaxação linear do problema. A partir de dos custos reduzidos de cada variável, é possível avaliar o quanto aquela variável pode contribuir para o valor da função objetivo do problema, caso seu valor seja fixado em 0 ou em 1, assim decidindo fixar o seu valor ou não.

Esse método não é novidade: uma versão mais fraca do que a descrita neste trabalho já foi utilizada antes, e sua eficiência pode ser comprovada em [Osorio 2002]. Com uma versão mais abrangente [Oliva 2001], esperamos fazer melhor.

Para a concepção da filtragem, inicialmente vamos analisar o funcionamento do método simplex aplicado ao seguinte problema com variáveis limitadas:

$$\begin{aligned} \text{(P)} \quad \max \quad & c^T x \\ \text{s.a:} \quad & Ax = b \\ & l_j \leq x_j \leq u_j, j = 1, ..., n \end{aligned}$$

No caso do simplex clássico, teríamos $l_j = 0$ e $u_j = +\infty$. O método simplex então divide as variáveis não-básicas em dois conjuntos: N_0 e N_1, onde N_0 são as variáveis cujo valor na solução básica corrente é igual ao limite inferior e N_1 são aquelas cujo valor é igual ao limite superior. Sendo μ uma solução básica dual, podemos então escrever a função objetivo da seguinte forma:

$$Q(x) = Q + \sum_{j \in N_0} \bar{c}_j (x_j - l_j) + \sum_{j \in N_1} (-\bar{c}_j)(u_j - x_j)$$

onde Q é o valor da função objetivo na solução corrente e $\bar{c}_j$ é o custo reduzido da variável x_j dado pela componente do vetor $\bar{c}^t = c^t - \mu A$ associada a j. O critério de otimalidade então seria:

$$\bar{c}_j \geq 0, \forall j \in N_1$$
$$\bar{c}_j \leq 0, \forall j \in N_0$$

2.2.1.1 Aplicação ao MKP

Considerando o caso do (MKP), como apresentado na Seção 2.1.1, seja s_i a variável de folga associada à restrição $i = 1 \cdots m$. Temos então que $0 \leq x_j \leq 1$ e $0 \leq s_i \leq +\infty$, para todo $i = 1 \cdots m$ e $j = 1 \cdots n$. Sendo $Q^\star$ o valor da solução ótima da relaxação linear e π_i para $i = 1 \cdots m$ o custo reduzido ótimo de s_i da relaxação linear do MKP, a função objetivo do problema pode ser escrita da seguinte forma:

$$Q'(x, s) = Q^\star + \sum_{j \in N_0} \bar{c}_j x_j + \sum_{j \in N_1} (-\bar{c}_j)(1 - x_j) + \sum_{i=1}^{m} \pi_i s_i$$

onde $\bar{c}_j \leq 0$, $\forall j \in N_0$, $\bar{c}_j \geq 0$ $\forall j \in N_1$ e $\pi \leq 0$. Note que $-\pi \geq 0$ é a solução ótima dual da relaxação linear do MKP, ou seja, $\bar{c}^t = c^t + \pi A$.

Suponha agora que seja conhecida uma solução (não necessariamente ótima) do problema (MKP). Essa solução fornece um limite inferior para o problema em questão. Seja $L^\star$ o valor da melhor solução conhecida para o problema. Uma solução melhor que a solução conhecida precisa verificar:

$$Q^\star + \sum_{j \in N_0} \bar{c}_j x_j + \sum_{j \in N_1} (-\bar{c}_j)(1 - x_j) + \sum_{i=1}^{m} \pi_i s_i > L^\star \tag{2.4}$$

2.2.1.2 Filtragem

Observe que em (2.4), os coeficientes de x_j, $(1 - x_j)$ e s_i são todos negativos. Isto nos leva a seguinte inequação:

$$s_i \leq \left\lfloor \frac{L^\star - Q^\star}{\pi_i} \right\rfloor, i = 1 \cdots m$$

Logo, podemos deduzir um limite inferior $\underline{b}_i$ para cada restrição do problema, da seguinte maneira:

$$\underline{b}_i = b_i - \left\lfloor \frac{L^\star - Q^\star}{\pi_i} \right\rfloor, i = 1 \cdots m$$

E podemos também, a partir de (2.4), fazer as seguintes deduções:

$$\forall j \in N_1 \quad L^\star - Q^\star > -\bar{c}_j \Rightarrow x_j = 1$$

$$\forall j \in N_0 \quad L^\star - Q^\star > \bar{c}_j \Rightarrow x_j = 0$$

Essa dedução de tempo linear $O(n)$, feita a partir de um método de programação linear, será usada como uma filtragem e poderá, caso as condições necessárias se façam presentes, ajudar um método de *branch-and-bound* a resolver o (MKP).

2.2.2 Uma restrição global: *Knapsum*

Uma desigualdade linear na forma $\underline{b} \leq \sum_{j=1}^{n} a_j x_j \leq \bar{b}$, como as restrições que aparecem na formulação do (MKP), pode ser filtrada, mas isso raramente irá fornecer alguma conclusão no caso de problemas difíceis. A idéia dessa Seção é combinar restrições para obter uma dedução mais eficiente. Além da restrição acima citada, vamos supor que sabemos quantas variáveis são iguais a 1 na solução do problema, ou seja, vamos fixar um valor k para o qual $\sum_{j=1}^{n} x_j = k$.

2.2.2.1 Definição

A programação por restrições permite que duas ou mais restrições sejam filtradas como uma só. Nesse caso define-se uma restrição global. Vamos chamar de *Knapsum* a restrição global cuja assinatura é: $Knapsum(a, b, k, x)$, onde $a \in R_+^n$, $b = (\underline{b}, \bar{b}) \in R_+^2$, $k \in N$ e x é um vetor de variáveis binárias. Mais formalmente, definimos *Knapsum* como:

$$Knapsum(a,b,k,x) \Leftrightarrow \left\{ \underline{b} \leq \sum_{j=1}^{n} a_j x_j \leq \bar{b},\ \sum_{j=1}^{n} x_j = k\ , x_j \in \{0,1\}, \forall j \in \{1,\cdots n\} \right\}$$

2.2.2.2 Filtragem

Seguindo a norma do processo de filtragem, a filtragem da restrição *Knapsum* possui duas fases: checar a viabilidade da restrição e em seguida verificar a possibilidade de redução de domínio das variáveis. Em nosso caso, qualquer redução de domínio de uma variável significa a fixação desta em um valor 0 ou 1.

Vamos supor, sem perda de generalidade que as variáveis estão ordenadas pelos coeficientes a_j em ordem decrescente, ou seja, $a_j \geq a_{j+1}$. Na prática essa ordenação só será executada uma vez para cada restrição.

1a. Fase

Seja $F = F_0 \cup F_1 \subseteq N = \{1, ..., n\}$ o conjunto de índices das variáveis previamente fixadas, onde F_0 é o conjunto de índices das variáveis fixadas em 0 e F_1 é o conjunto de índices das variáveis fixadas em 1. Seja $L = N \backslash F$ o conjunto dos índices das variáveis livres e defina a função $l(i) : \{1, ..., |L|\} \rightarrow N$ como um reposicionamento dos índices de $\{1, ..., |L|\}$ em N preservando a ordem dos coeficientes, ou seja, $a_{l(i)}$ é o i-ésimo maior coeficiente em $\{a_j : j \in L\}$. A primeira fase consiste no seguinte procedimento, que pode ser realizado em $O(k)$.

1. Calcular $S_F = \sum_{j \in F_1} a_j$.
2. Calcular $\bar{S} = S_F + \sum_{j=1}^{k-|F_1|} a_{l(j)}$.
3. Calcular $\underline{S} = S_F + \sum_{j=|L|-(k-|F_1|)+1}^{|L|} a_{l(j)}$.
4. Se $\bar{S} < \underline{b}$ ou $\underline{S} > \bar{b}$ então a restrição é inviável.

Nesse passo, calculamos os somatorios dos coeficientes das variáveis fixadas S_F e a partir daí calculamos os somatório dos k maiores coeficientes $\bar{S}$ e dos k menores coeficientes $\underline{S}$, dada a fixação F. Então testamos se é possível para tal restrição obedecer os limites $\underline{b}$ e $\bar{b}$. Se não for possível, paramos. Do contrário, podemos passar para a 2a. fase.

2a. Fase

Para fixar variáveis vamos usar o algoritmo da fase anterior, estendendo o conjunto F com uma tentativa de fixação: $F_1 = F_1 \cup \{i\}$ ou $F_0 = F_0 \cup \{i\}$. Caso o algoritmo

retorne inviável como resposta, é possível concluir que a variável deve ser fixada com o valor diferente da tentativa.

No entanto não precisamos testar valores para todas as variáveis indexadas por L: podemos aproveitar a ordem dos coeficientes e testar até onde não for mais possível fazer a fixação. Além disso, os valores de $\bar{S}$ e $\underline{S}$ não precisam ser recalculados, mas sim atualizados a cada iteração do algoritmo, tornando o processo mais rápido.

1. Para $\bar{b}$
   (a) Para $j = 1,...,|L|$
      i. Fazer $F_1 = F_1 \cup \{l(j)\}$.
      ii. Chamar algoritmo anterior.
      iii. Se o algoritmo retornar inviável, fixar $x_{l(j)} = 0$. Caso contrário, sair do loop.
   (b) Para $j = |L|,...,1$
      i. Fazer $F_0 = F_0 \cup \{l(j)\}$.
      ii. Chamar algoritmo anterior.
      iii. Se o algoritmo retornar inviável, fixar $x_{l(j)} = 1$. Caso contrário, sair do loop.
2. Para $\underline{b}$
   (a) Para $j = 1,...,|L|$
      i. Fazer $F_0 = F_0 \cup \{l(j)\}$.
      ii. Chamar algoritmo anterior.
      iii. Se o algoritmo retornar inviável, fixar $x_{l(j)} = 1$. Caso contrário, sair do loop.
   (b) Para $j = |L|,...,1$
      i. Fazer $F_1 = F_1 \cup \{l(j)\}$.
      ii. Chamar algoritmo anterior.
      iii. Se o algoritmo retornar inviável, fixar $x_{l(j)} = 0$. Caso contrário, sair do loop.

Muitos detalhes são omitidos do algoritmo acima por simplicidade: se em algum momento tivermos k variáveis fixadas em 1 ou $n - k + 1$ variáveis fixadas em 0, as variáveis livres são automaticamente fixadas no único valor possível restante; como já foi dito antes, não é preciso realmente chamar o algoritmo da fase 1 e recalcular $\bar{S}$ e $\underline{S}$, basta uma atualização que pode ser feita em tempo constante, guardando os índices das variáveis de maior coeficiente em $\underline{S}$ e menor coeficiente em $\bar{S}$.

Como reaproveitamos o cálculo de $\bar{S}$ e $\underline{S}$, esse algoritmo é $O(n)$ ao invés de $O(nk)$. Se fôssemos fazer uma vez para cada restrição, teríamos uma complexidade total de $O(mn)$

para a filtragem, mas como toda vez que uma variável é fixada temos que fazer uma vez para cada restrição, todo o processo de filtragem é $O(kmn)$, e executa bem rápido na prática.

2.3 Branch-and-Cut para o MKP

O *branch-and-bound* é uma poderosa técnica de resolução de problemas de programação inteira. Não seria exagero dizer que ela e suas derivadas (*branch-and-cut*, *branch-and-price*) são as técnicas mais utilizadas, hoje em dia, para a resolução de programação inteira ou mista. Consiste na enumeração de soluções, onde são descartadas aquelas desinteressantes: que não podem fornecer uma solução de valor melhor que um valor conhecido ou que tornem o problema inviável. Para a avaliação das soluções um método bastante utilizado é a resolução da relaxação linear do problema, onde são descartadas as restrições de integralidade. No caso do (MKP), quando a relaxação linear encontra um valor abaixo do valor da melhor solução viável conhecida, o nó pode ser descartado.

O nosso trabalho é baseado em um *branch-and-cut*, onde o *branch-and-bound* está aliado a um processo de geração de desigualdades válidas para o problema. Essas desigualdades são chamadas *cortes* se elas tornam a solução encontrada pela relaxação linear inviável. Decidir se uma desigualdade corta a solução da relaxação linear do problema é, em nosso caso, um problema NP-difícil, mas heurísticas conseguem um resultado satisfatório no caso de cortes clássicos.

Além da avaliação da relaxação linear e da geração de cortes clássicos para o (MKP), como veremos a seguir, o nosso processo de *branch-and-cut* ainda conta com o possível auxílio dos procedimentos de filtragem descritos no Capítulo 2.2, em cada nó da árvore, que vai guiar ou descartar soluções conforme o processo descrito. Além disso, a filtragem da restrição *Knapsum* nos permite gerar outros cortes que serão detalhados na SubSeção 2.3.2.2.

2.3.1 O esquema de enumeração

O esquema de enumeração adotado em nosso trabalho se diferencia do clássico branch-and-cut, pois trata-se de um processo de enumeração dividido em etapas. São elas: (a) a concepção de um limite inferior para o problema inteiro; (b) a criação de hiperplanos com solução de cardinalidade fixa; e finalmente (c) para cada hiperplano, desempenhamos um branch-and-cut no qual os resultados obtidos pelos outros branch-and-cut de outros hiperplanos podem atualizar o limite inferior. Cada uma dessas fases será discutida nas seções a seguir.

2.3.1.1 Cálculo de um limite inferior

O primeiro passo do esquema de enumeração é a aquisição de um limite inferior para o problema. Através de uma heurística, buscaremos estabelecer um valor mínimo de solução

desejável no problema.

O valor do limite inferior tem grande influência no processo de filtragem dos custos reduzidos e testes experimentais mostram que um pequeno (0.01%) aumento no valor desse limite pode significar a fixação de muitas variáveis. Sabendo disso, em [Vimont 2008], os autores procuraram utilizar uma poderosa heurística tabu para a obtenção desse limite. Em muitos casos, a maior parte do tempo é gasta na busca desse valor.

Nossa estratégia já segue uma outra linha: vamos começar com um valor "fraco"para o limite inferior, obtido com uma heurística gulosa (através da ordenação das variáveis segundo a razão entre o seu coeficiente na função objetivo e o seu coeficiente na restrição agregada de todas as restrições de capacidade do problema) e vamos, ao longo do processo de enumeração escolher os ramos que têm maior chance de aumentar o valor do limite inferior, potencializando a etapa da programação por restrições. Dessa forma, economizamos o tempo gasto a priori em comparação ao método proposto em [Vimont 2008] e aproveitamos o momento de exploração dos nós para aplicar uma "heurística"para o valor do limite inferior.

2.3.1.2 Geração de hiperplanos de cardinalidade fixa

A cardinalidade de uma solução do problema das mochilas múltiplas pode ser definida como o número de variáveis iguais a 1 na solução. Seguindo as observações vistas em [Vimont 2008], o passo seguinte na enumeração é estabelecer valores mínimo e máximo para a cardinalidade da solução ótima.

Dado o valor do limite inferior $L^\star$, calculado por qualquer um dos métodos apresentados na SubSeção anterior, resolvemos os seguintes problemas lineares para obter os limites de cardinalidade:

$$
\begin{aligned}
k_{min} = \min \quad & \sum_{j=1}^{n} x_j \\
\text{s.a:} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\
& \sum_{j=1}^{n} c_j x_j \geq L^\star \\
& x_j \in [0,1], \qquad j = 1, ..., n
\end{aligned}
\qquad\qquad
\begin{aligned}
k_{max} = \max \quad & \sum_{j=1}^{n} x_j \\
\text{s.a:} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\
& \sum_{j=1}^{n} c_j x_j \geq L^\star \\
& x_j \in [0,1], \qquad j = 1, ..., n
\end{aligned}
$$

A seguir, para cada um dos possíveis valores de $k = \lceil k_{min} \rceil, ..., \lfloor k_{max} \rfloor$ iremos resolver o problema inteiro associado adicionando a restrição de cardinalidade. Pode parecer uma idéia muito ruim resolver diversos problemas ao invés de um só, mas experimentos anteriores mostram que isso melhora o tempo de resolução.

2.3.1.3 "*Branch-and-cut*"para o problema com restrição de cardinalidade

A formulação abaixo descreve o problema com restrição de cardinalidade, o qual, nesta Seção, é o nosso maior interesse. Após a geração de hiperplanos de cardinalidade, temos $k_{max} - k_{min} + 1$ problemas distintos para resolver.

$$(MKP_k) \max \quad \sum_{j=1}^{n} c_j x_j \tag{2.7a}$$

$$\text{s.a:} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \tag{2.7b}$$

$$\sum_{j=1}^{n} x_j = k \tag{2.7c}$$

$$\sum_{j=1}^{n} c_j x_j \geq L^\star \tag{2.7d}$$

$$x_j \in \{0, 1\}, \quad j = 1, ..., n \tag{2.7e}$$

O problema com restrição de cardinalidade possui diversos aspectos dos quais podemos tirar proveito. Observe que cada uma das restrições (2.7b) junto com a restrição (2.7c) formam a restrição múltipla *Knapsum* e, portanto podemos realizar a filtragem proposta na Seção 2.2.2. Além disso, a restrição (2.7d) combinada com a restrição (2.7c) também forma uma restrição do tipo *Knapsum* e, portanto, pode ser usada no processo de filtragem. A partir da formação desses problemas MKP_k começa a resolução do MKP por um método de *branch-and-cut*.

Nessa etapa do processo usaremos o nosso esquema cooperativo. Após a resolução da relaxação linear do MKP_k, passaremos a uma etapa de filtragem das restrições. Dependendo do resultado da filtragem, podemos voltar à relaxação linear ou seguir para a avaliação do nó. Caso o nó seja inviável, podemos cortar a árvore neste ponto; caso tenhamos todas as variáveis fixadas a 0 ou 1, guardamos a nova solução e atualizamos o valor de limite inferior do hiperplano e dos demais hiperplanos; caso uma inviabilidade ou uma solução inteira não sejam atingidas, passamos à etapa de ramificação.

Decidimos fazer a ramificação seguir um padrão incomum, conforme visto em [Vimont 2008]. Ao invés dos esquemas tradicionais de ramificar sobre as variáveis básicas, nesse caso, as variáveis não básicas serão o interesse do nosso método. Primeiramente vamos ordenar as variáveis não-básicas pelo valor absoluto do custo reduzido, de forma decrescente. Para o primeiro ramo, à variável com maior valor absoluto de custo reduzido será atribuído o valor contrário àquele que ela recebeu na relaxação linear do nó corrente. Em seguida, no segundo ramo, a primeira variável recebe o valor correspondente ao da relaxação linear e a variável seguinte na ordem recebe o valor contrário ao da relaxação linear. No terceiro ramo, as duas primeiras concordam com a relaxação linear e a terceira variável apresenta um valor diferente e assim continua até que no último ramo todas

as variáveis concordam com o valor dado pela relaxação linear. Nesse último ramo, um processo de enumeração ímplicita (decidimos por uma busca em profundidade) é lançado sobre as variáveis básicas do problema.

Exemplo: Vamos supor que em um determinado nó de nosso processo de enumeração, após resolver a relaxção linear no referido nó, temos em nosso problema quatro variáveis não-básicas (x_1, x_2, x_3, x_4) cujos valores atribuídos na solução da relaxação linear são $(1, 1, 1, 0)$. Sem perda de generalidade, vamos assumir que elas estão ordenadas (não-crescente) pelo valor absoluto de seu custo reduzido. Vamos marcar por $*$ se a variável é livre. Portanto, os nós a serem gerados a partir do nó corrente são: $(0, *, *, *)$, $(1, 0, *, *)$, $(1, 1, 0, *)$, $(1, 1, 1, 1)$, $(1, 1, 1, 0)$.

2.3.2 Cortes para o MKP

Apresentamos aqui os cortes possíveis de serem gerados através da filtragem de restrições *Knapsum*. Começamos por apresentar os *cover cuts* para efeito de contraste com os cortes passíveis de serem gerados pelo método original.

2.3.2.1 Cortes clássicos associados ao MKP: *cover cuts*

Os *cover cuts* estão entre os cortes mais conhecidos da programação matemática. Eles são fáceis de serem encontrados e na prática fornecem bons resultados. Ainda assim são cortes que podem ser facilmente fortalecidos por processos simples.

Considere o conjunto $X = \left\{ x \in \{0,1\} : \sum_{j=1}^{n} a_j x_j \leq b \right\}$ com $a_j \geq 0$. Um conjunto de índices $C \subseteq \{1, ..., n\}$ é uma cobertura se $\sum_{j \in C} a_j > b$. Além disso, C é uma cobertura minimal se, para todo $j \in C$, $C - \{j\}$ não é uma cobertura.

A idéia de um *cover cut* é que se um conjunto de variáveis recebendo valor 1 viola uma restrição do problema, então deve-se excluir uma delas de uma solução viável do problema. Portanto, para uma cobertura C, a seguinte desigualdade:

$$\sum_{j \in C} x_j \leq |C| - 1$$

é válida para o politopo do problema. Uma demonstração formal desse resultado pode ser encontrada em [Wolsey 1998]. Além disso, como pode-se verificar na mesma referência, os *cover cuts* podem ser facilmente fortalecidos. Fortalecer um *cover cut* signifca adicionar variáveis ao lado esquerdo da restrição, sem torná-la inválida. Defina $E(C) = C \cup \{j : a_j \geq a_i, \textit{para todo } i \in C\}$. Logo,

$$\sum_{j \in E(C)} x_j \leq |C| - 1$$

é uma desigualdade válida para o problema.

Heurísticas para encontrar *cover cuts* e para fortalecê-los são temas recorrentes na literatura do problema da mochila [Glover 2008]. Nosso trabalho agrega esse conhecimento ao inserir esses cortes durante o processo de *branch-and-cut.*

2.3.2.2 Cortes da filtragem do *Knapsum*

Dividimos os cortes derivados da programação por restrições em dois tipos. O tipo 1 são cortes que vêm de uma tentativa de fixação de variáveis (baseada na solução da relaxação linear do problema). O segundo tipo são desigualdades do tipo *cover cuts* que podem ser adquiridos na execução de uma filtragem de uma restrição qualquer, observando os coeficientes que estão sendo somados em $\bar{S}$ e $\underline{S}$.

`Tipo 1`

Seja $F = F_0 \cup F_1$ um conjunto de índices de variáveis fixadas, onde $i \in F_q \Leftrightarrow x_i = q$. Suponhamos então que a filtragem da restrição *Knapsum* retornou uma inviabilidade, dada essa fixação. Claramente, uma das variáveis cujo índice está em F tem mudar seu valor para o valor oposto. Isso significa dizer que:

$$\sum_{j \in F_0} x_j + \sum_{j \in F_1} (1 - x_j) \geq 1 \tag{2.8}$$

ou ainda, podemos escrever como:

$$\sum_{j \in F_1} x_j - \sum_{j \in F_0} x_j \leq |F_1| - 1 \tag{2.9}$$

Ao longo do processo de filtragem *Knapsum*, podemos, além de deduzir uma inviabilidade, deduzir novas fixações. Essas novas fixações também podem ser vistas como desigualdades válidas para o problema. Basta que saibamos que a fixação contrária induz uma inviabilidade. Portanto, se encontramos que $F \Rightarrow x_i = 0$, a seguinte desigualdade:

$$\sum_{j \in F_1 \cup \{i\}} x_j - \sum_{j \in F_0} x_j \leq |F_1| \tag{2.10}$$

é válida para o problema. Igualmente $F \Rightarrow x_i = 1$ leva à:

$$\sum_{j \in F_1} x_j - \sum_{j \in F_0 \cup \{i\}} x_j \leq |F_1| - 1 \tag{2.11}$$

Outro ponto importante a notar é que se a filtragem de uma dada equação prova que uma fixação F é inviável ou que induz outra fixação, nem todas as variáveis indicadas por F *participam* da filtragem de forma *ativa.* Quer dizer, a filtragem de uma certa restrição supõe, pela própria definição, que algumas variáveis estão 'fixadas' para calcular o somatório de $\bar{S}$ e $\underline{S}$ como na Seção 2.2. Mais precisamente, as k maiores variáveis fixadas a 1 e as k menores variáveis fixadas a 0 não *ajudam* na dedução do corte. Podemos provar, então, que elas podem ser retiradas do corte, fortalecendo o mesmo.

Portanto, aplicando o processo de filtragem a partir de um conjunto de fixações $F = F^0$, a programação por restrições vai, possivelmente, fazer novas deduções (fixações) e gerar conjuntos F^1, F^2, ..., F^p que acumulam os índices i das variáveis tais que $F^t \Rightarrow x_{i(t+1)} = q$, $q \in \{0, 1\}$. O processo inevitavelmente para, quer seja por uma inviabilidade, quer seja por não conseguir fazer novas deduções. Então para cada uma das deduções, dois cortes podem ser deduzidos: o corte equivalente à dedução $F^t \Rightarrow x_{i(t+1)}$ e o corte equivalente à dedução $\hat{F}^t \Rightarrow x_{i(t+1)}$, onde $\hat{F}^t$ é o conjunto F^t em que foram retiradas as variáveis que não *ajudam* na dedução.

Vamos mostrar um exemplo. Suponha a seguinte restrição *Knapsum* definida pelas expressões abaixo:

$$15x_1 + 13x_2 + 4x_3 + 2x_4 \leq 20$$
$$x_1 + x_2 + x_3 + x_4 = 2$$
$$x_i \in \{0, 1\}$$

e suponhamos que $F^0 = \{x_1 = 1, x_2 = 0\}$. Esse conjunto daria origem ao corte:

$$x_1 - x_2 \leq 1.$$

Porém, poderíamos tentar retirar a variável x_2 do corte. Ora, se o fizermos, teríamos como conjunto $F^1 = \{x_1 = 1\}$. Esse conjunto claramente deduz que $x_2 = 0$. A partir de F^1 poderiamos então escolher heurísticamente uma fixação para constituir F^3 e continuar o método iterativamente.

Esses cortes se assemelham dos *cover cuts*, exceto pela exclusão do somatório de variáveis em F_0. Naturalmente o problema de separação é a maior dificuldade para a geração dos cortes, ou seja, fazer uma boa escolha dos conjunto F_1 e F_0. A possibilidade que adotamos neste trabalho é nos basear na relaxação linear para fazer essa escolha. Vamos definir F_1 como o conjunto de índices de variáveis as quais são iguais a 1 na solução da relaxação linear e F_0 é o conjunto de índices de variáveis iguais a 0. Então, começamos a tentar retirar variáveis desse conjunto, caso encontremos alguma dedução (na tentativa de encontrar um corte mais forte). Caso contrário, adicionamos variáveis a F que podem ajudar a fazer uma dedução. O algoritmo abaixo mostra a geração desses cortes:

Sejam $x^\star$ a solução da relaxação linear e $\delta_j = min\{x_j^\star, 1 - x_j^\star\}$ e faça $F_0 = \{j : x_j^\star = 0\}$ e $F_1 = \{j : x_j^\star = 1\}$.

1. Fazer a filtragem *Knapsum* para o conjunto $F = F_1 \cup F_0$.
2. Se o processo terminou com uma inviabilidade, armazenar a última dedução encontrada, retirar um índice de F segundo algum critério e recomeçar 1. Senão vá para 3.
3. Se a filtragem conseguiu fazer novas deduções, armazenar para cada variável fixada a dedução correspondente, retirar um índice de F segundo algum critério e recomeçar 1. Senão vá para 4.

4. Se a filtragem não foi capaz de encontrar nenhuma dedução, aumentar o conjunto F de uma variável $x_i = q$, $q \in \{0, 1\}$, caso $\sum_{j \in F} \delta_j + \delta_i < 1$, onde $i = arg\ min_j\{\delta_j : j \notin F\}$. Caso contrário, vá para 5.

5. Para cada dedução armazenada, gerar os cortes correspondentes.

Nos passos 2 e 3, retiramos uma variável de F na tentativa de encontrar cortes mais fortes. O critério que usamos em nossos testes foi o de retirar a variável com custo reduzido mínimo, embora o custo reduzido máximo também conduza a resultados interessantes. Ainda não está claro qual critério é o melhor nesse caso.

A idéia de usar o conjunto F baseado na solução da relaxação faz necesserariamente com que as desigualdades advindas de uma dedução de inviabilidade cortem a solução da relaxação linear.

Tipo 2

Ao calcularmos os valores de S_F, $\bar{S}$, $\underline{S}$ na primeira fase do algoritmo *Knapsum*, conforme a Seção 2.2, nós o testamos se algumas variáveis podem ser fixadas a 1: no caso de $\bar{S}$, as variáveis com maiores coeficientes, e no caso de $\underline{S}$, as que têm os menores coeficientes. Para a filtragem, só comparamos $\bar{S}$ com $\underline{b}$ e $\underline{S}$ com $\bar{b}$.

No entanto, caso $\bar{S} > \bar{b}$, podemos inferir que os índices das variáveis cujos coeficientes fazem parte da soma de $\bar{S}$ recaem na definição de uma cobertura dada na Subseção 2.3.2.1. Fácil perceber que acontece algo equivalente para o caso do somatório $\underline{S} < \underline{b}$.

Na tentativa de encontrar uma cobertura minimal, podemos, ao longo do processo de cálculo de $\bar{S}$ e $\underline{S}$, checar a cada iteração se já foi encontrada uma cobertura. Caso positivo, armazenamos o *cover cut* associado à mesma. Claro que ainda podemos fortalecer o corte como descrito na Seção 2.3.2.1.

2.4 Combinação de Restrições

A filtragem *Knapsum* tem como idéia base olhar para duas restrições (knapsack e cardinalidade) ao mesmo tempo. No entanto, existem técnicas que nos permitem considerar várias restrições do tipo knapsack ao mesmo tempo, como uma unica restrição, desde que seus coeficientes sejam inteiros. A motivação desta Seção é mostrar como combinar essas restrições e definir uma restrição *Knapsum* que possa levar em conta múltiplas (possivelmente todas) as restrições do problema de uma vez.

2.4.1 Combinando duas restrições

Combinar duas restrições, em nosso caso, significa encontrar uma terceira restrição que seja funcionalmente equivalente às duas primeiras. Mostraremos que, em caso de uma restrição de igualdade, é possível fazer a substituição de duas restrições por uma combinação delas, se certas condições são satisfeitas.

De um resultado devido à [Garfinkel 1972] temos a seguinte proposição:

Proposição 2.4.1 *Dadas uma variável $x \in Z_M^n$ e as seguintes restrições (satisfeitas em pelo menos um ponto em Z_M^n):*

$$a^t x = b \quad e \quad c^t x = d \tag{2.12}$$

onde $a, c \in Z_+^n$ e $b, d \in Z$, existe um valor α tal que a restrição

$$\alpha(a^t x) + c^t x = \alpha b + d \tag{2.13}$$

tem a seguinte propriedade: $x \in Z_M^n$ satisfaz (2.12) se e somente se $x \in Z_M^n$ satisfaz (2.13), onde $Z_M^n = \{(k_1, \cdots, k_n) : k_i \in \{1, \cdots, M\}, \forall i \in \{1, \cdots, n\}\}$.

Prova Suponha x satisfazendo as duas equações iniciais. Nesse caso é facil ver que (2.13) também é satisfeita. A dificuldade da prova reside no outro lado. Suponha agora $x \in Z_M^n$ satisfazendo (2.13). Reescrevendo (2.13) temos:

$$\alpha(a^t x - b) = d - c^t x \tag{2.14}$$

Claramente, como $x \in Z_M^n$, podemos definir α tal que:

$$\alpha > max_x\{|d - c^t x| : x \in Z_M^n\} \tag{2.15}$$

Suponha, por absurdo, $c^t x \neq d$. Como $\alpha > 0$, então $a^t x - b$ e $d - c^t x$ têm o mesmo sinal. Assim, (2.14) implica que

$$\begin{aligned} \alpha|a^t x - b| &= |d - c^t x| \\ |d - c^t x||a^t x - b| &< |d - c^t x| \qquad \text{pela definição de } \alpha \text{ em (2.15)} \\ |a^t x - b| &< 1 \end{aligned}$$

Como $|a^t x - b|$ é um valor inteiro, então $a^t x - b = 0 \Leftrightarrow a^t x = b$. A partir de (2.14), temos que $c^t x = d$. Uma contradição. Portanto temos $c^t x = d$ e, por (2.14), $a^t x = b$.

2.4.2 Aplicando a filtragem sobre a restrição combinada

Em nosso caso não temos uma restrição de igualdade a priori, mas com a inserção de uma variável de folga inteira completaria as condições necessárias para a aplicação da técnica vista na Seção anterior. Basta agora adaptar a filtragem apresentada na Seção 2.2.2 para o caso com a variável de folga.

O interesse dessa agregação para a filtragem *Knapsum* é diverso: primeiro ela pode ser usada para reduzir o número de restrições no laço de propagação das restrições. Em segundo lugar ela pode fornecer reduções disjuntas daquelas fornecidas pelas restrições filtradas individualmente, podendo assim serem adicionadas ao conjunto de restrições na camada de programação por restrições (e, evidentemente, omitidas da camada de programação linear).

Verificamos que a parte mais custosa da filtragem *Knapsum* é a ordenação dos coeficientes da restrição. A agregação pode ser usada para a combinação de todas as restrições com uma restrição de referência de forma que todas as restrições agregadas possuam a mesma ordem de coeficientes (desde que todos os coeficientes sejam não nulos na restrição de referência. Dessa forma basta escolher um α grande suficiente para obter o resultado desejado.

No método RMC + COMBO visto na Seção 2.6 escolhemos de agregar todas as restrições de mochila com a restrição de limite inferior $\sum_{j=1}^{n} c_j x_j \geq L$, onde L é o valor da melhor solução conhecida.

2.5 Resumo do método

Para recapitular todo o procedimento, fazemos aqui um resumo do método proposto.

O primeiro passo para a resolução do MKP é a obtenção de uma solução heurística. Nossa heurística consiste em, primeiramente, resolver a relaxação linear do problema. Em seguida agregamos as restrições segundo seus coeficientes duais para a obtenção de uma restrição agregada. A razão entre o coeficiente na função objetivo e o coeficiente na função agregada constitui uma ordem de prioridade entre as variáveis. A partir desta ordem utilizamos uma heurística gulosa para a obtenção de uma solução viável para o problema inteiro.

Em seguida, tendo essa solução, fazemos a divisão do problema segundo os planos de cardinalidade. Para tanto, determinamos qual a cardinalidade mínima k_min e a cardinalidade máxima k_max que uma solução melhor ou igual a solução heurística deve apresentar (ver Seção 2.3.1.2). Dividimos o problema original em $k_max - k_min + 1$ problemas MKP com cardinalidade fixa. Esses problemas constituem as raízes para o método de enumeração cooperativo.

Esse método de enumeração consiste em três principais etapas: resolução da relaxação linear, deduções através da propagação das restrições e, por fim, a ramificação. O processo de propagação se divide em duas principais filtragens: a filtragem de custos reduzidos e a filtragem das restrições *Knapsum*. A filtragem de custos reduzidos também vai influenciar no processo de ramificação.

A filtragem da restrição *Knapsum* começa pela seleção do conjunto de restrições a serem filtradas. No caso do método sem agregação fazemos a filtragem sobre as restrições originais combinadas com a restrição de cardinalidade. No caso agregado, fazemos primeiro a agregação das restrições com a restrição de limite inferior e filtramos a restrição agregada combinada com a restrição de cardinalidade (ver Seção 2.4). Em ambos os casos fazemos ainda a filtragem da restrição de limite inferior combinada com a restrição de cardinalidade. É importante notar que a propagação das restrições *Knapsum*, contrariamente à propagação da filtragem de custos reduzidos, pode cortar a solução da relaxação linear. Se este for o caso, iteramos o processo à partir da resolução da relaxação linear.

Quando o processo colaborativo acaba, para um dado nó, começa o processo de ramificação. As variáveis são ordenadas segundo seus custos reduzidos e a ramificação é realizada conforme visto na Secção 2.3. O processo de seleção de nós da árvore de enumeração é não trivial e envolve uma tentativa de diversificação das características dos nós selecionados (para mais detalhes, ver Seção 2.3).

Finalmente quando os nós possuem um número suficientemente pequeno de nós em relação ao problema original (menos de 20 variáveis, para as instâncias que testamos), partimos para uma enumeração em profundidade sem o cálculo do limite superior, utilizando apenas a filtragem *Knapsum*.

2.6 Resultados Computacionais

2.6.1 Ambiente computacional

Todos os testes descritos nas tabelas abaixo foram realizados em uma computador Intel Xeon 2.66 GHz, 1GB de memória RAM. Para efeitos de comparação foi utilizado o software comercial CPLEX, versão 10.1. Toda a codificação foi feita em linguagem C++ e compilado com g++ 4.2.

2.6.2 Instâncias consideradas

As instâncias consideradas neste trabalho são aquelas que se encontram na OR-library [Beasley 1990]. Elas foram obtidas a partir de um gerador de instâncias que utiliza diversos parâmetros. Os valores dos coeficientes das restrições são números inteiros gerados entre 0 e 1000. Os coeficientes da função objetivo são correlatos ao somatório dos coeficientes das restrições (o que experimentalmente provou-se mais difícil). O principal parâmetro, além do tamanho (número de variáveis e de restrições), é um valor $\alpha \in [0, 1]$, que mede a razão entre o lado direito e o lado esquerdo de cada restrição, ou seja, $\alpha = \frac{b_i}{\sum_{j=1}^{n} a_{ij} x_j}$. Foi constatado que os problemas ficam mais difíceis de se fazer deduções com relação ao custo reduzido conforme o valor de α cresce, dado o mesmo número de variáveis e de restrições. Naturalmente, quanto maior o número de variáveis e de restrições, mais difícil será o problema também.

Nossa bateria de testes considera seis classes de instâncias, cada uma com 30 exemplos. O tamanho das instâncias varia de 100 à 500 variáveis, e o número de restrições fica entre 5 ou 10 restrições. O parâmetro α supracitado varia entre 0,25 e 0,75.

2.6.3 Tabelas de resultados

As medidas comparativas que tomaremos para cada uma das instâncias serão: tempo de *branch-and-bound* e número de nós na árvore de decisão. Os métodos escolhidos para comparação foram o método que inclui as filtragens apresentadas e o método que faz as filtragens e a combinação de restrições, conforme apresentada. A inclusão de cortes não

proporcionou diferenças significativas nos testes preliminares e a adição das técnicas de programação dinâmica levaram a um tempo total ainda menos interessante.

2.6.3.1 Resultados do nosso método

As tabelas 2.1-2.6 informam o status final da execução do nosso método para cada classe de instâncias CB1-CB6. Existe uma tabela para cada conjunto de 30 instâncias na qual estão descritas o número de nós que foram gerados na árvore e o tempo total, em segundos, que o método levou na resolução do problema no caso das tabelas 2.1-2.5. É possível afirmar que a versão com a agregação de restrições é superior em geral à versão sem a agregação por uma ligeira margem que não permite de diferenciá-las.

A tabela 2.6 mostra os resultados para um conjunto de instâncias para o qual nosso problema não encontrou o valor ótimo para todas as instâncias em menos de 24 horas, tempo limite estabelecido. Nessa tabela mostramos o valor do limite inferior obtido na coluna "L". Esse limite inferior é corresponde ao valor de uma solução realizável. O tempo levado para a obtenção desse valor é apresentado na coluna "Tempo L". Além disso mostramos o valor da solução ótima para cada uma das instâncias, calculado através do método apresentado em [Boussier 2009] sem limitação de tempo (com efeito, certas instâncias levaram diversos meses para serem resolvidas, sendo o tempo CPU total de cálculo de todas as instâncias superior a 1 ano). A coluna "Provou"mostra se o método apresentado conseguiu provar que a solução obtida é a solução ótima.

Vemos que apesar de não ter resolvido todas as instâncias, o método apresentado é capaz de obter o valor ótimo de 26 das 30 instâncias em menos de 24 horas (sendo que 20 instâncias chegaram ao ótimo em menos de 2 horas), provando a otimalidade de 11 delas.

2.6.3.2 Comparação com outros métodos

A tabela 2.7 foi confeccionada pensando em uma comparação entre o método desenvolvido e o pacote comercial ILOG CPLEX. Além disso, comparamos também com o melhor método desenvolvido segundo nosso conhecimento [Boussier 2009]. Todos os métodos foram submetidos aos mesmos parâmetros no mesmo ambiente computacional. Para cada um deles foi medido o tempo de execução em segundos e para cada conjunto de instâncias é apresentado a média do tempo.

A figura 2.1 mostra um comparativo dos dois algoritmos desenvolvidos com o melhor método conhecido para o problema [Boussier 2008, Boussier 2009] em escala logaritmica. Vemos que os métodos apresentados mantém uma grande vantagem apesar do crescimento exponencial do tempo de resolução do problema.

2.7 Conclusão

Apresentamos neste trabalho um método para solução do Problema das Mochilas Múltiplas. Ele se baseia no esquema cooperativo desenvolvido por [Oliva 2001] que promove

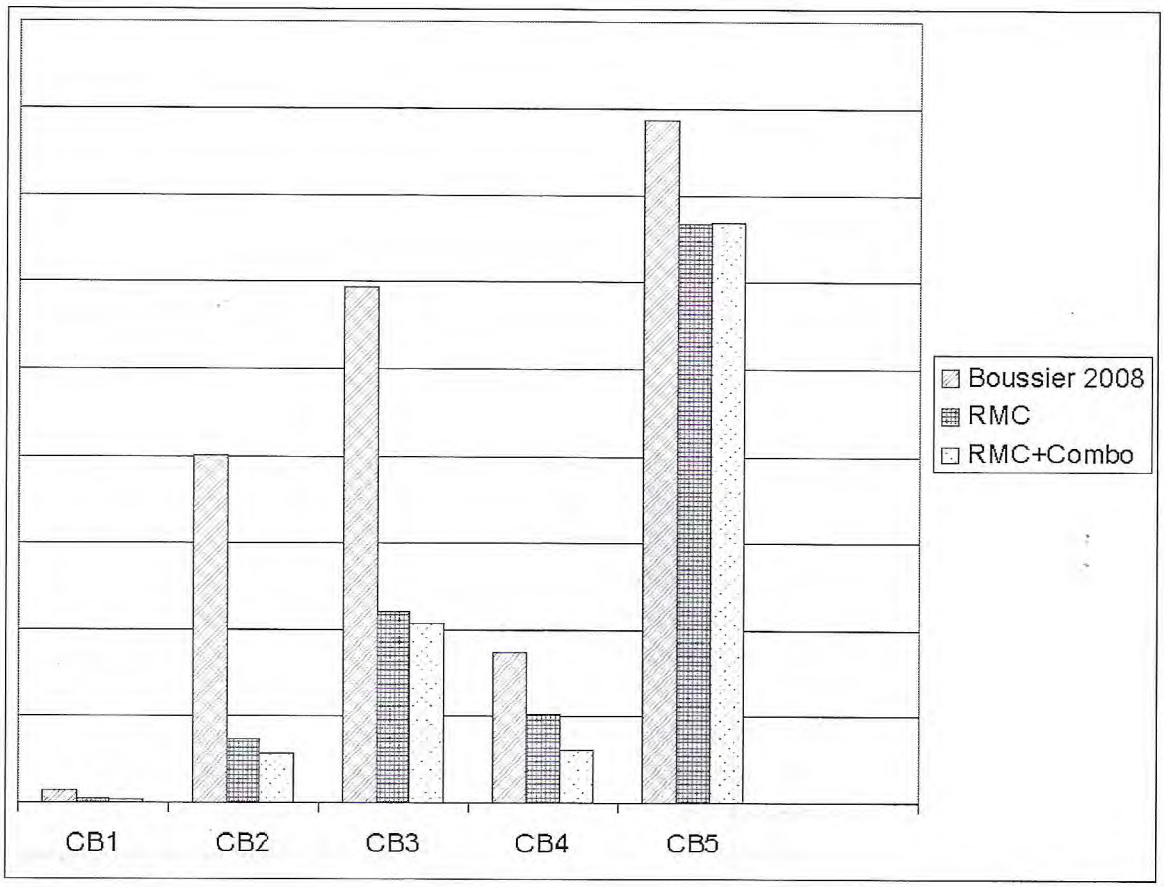

Figura 2.1: Comparação do tempo de computação entre o método proposto e o melhor método na literatura (em escala logarítmica).

a cooperação entre técnicas de Programação Matemática (PM) e Programação por Restrições (PPR). No que diz respeito à PM, aplicamos tal esquema sobre o plano enumerativo desenvolvido por Vimont, Boussier e Vasquez [Vimont 2008]. A árvore de busca gerada dessa forma é altamente desbalanceada quanto ao número de variáveis fi-xadas para cada filho, o que torna essencial uma boa escolha do percurso a ser seguido na exploração da mesma. Propomos uma escolha heurística nesse trabalho que tem por meta visitar inicialmente nós com muitas variáveis fixadas (fáceis) em torno da solução da relaxação linear na raiz.

Quanto à programação por restrições, utilizamos dois tipos de filtragens: sobre os custos reduzidos e sobre uma nova restrição global, a qual chamamos *Knapsum*. A restrição *Knapsum* engloba duas restrições do problema: uma restrição de 'mochila', da qual temos m, por definição do (MKP), e uma restrição de cardinalidade, que aparece quando aplicamos o esquema de enumeração supracitado. Além das restrições de mochila, *Knapsum* também pode ser definida para a restrição de limite inferior. Assim sendo,

temos $m + 1$ restrições globais desse tipo. Além de permitir a fixação de variáveis ou deduzir prematuramente a poda de nós, a filtragem *Knapsum* também permite a geração de cortes semelhantes aos *cover cuts*. Porém, assim como para os outros métodos eficazes para esse problema, a adição de tais cortes adicionaram um 'peso computacional' ao modelo, o que não prova sua importância, ou seja, os cortes não promoveram significativa melhora para o método.

A eficiência das filtragem é um componente essencial para caracterizar um bom método híbrido. Podemos constatar que a filtragem por custos reduzidos tem complexidade $O(n)$, enquanto a filtragem de uma restrição *Knapsum* pode chegar a $O(nlog(n))$, devido ao reordenamento de cada restrição. Além disso, a propagação sobre todas as m restrições pode elevar a complexidade total a $O(mn^2log(n))$. Uma maneira de evitar o reordenamento completo das restrições, ou evitar uma passagem por cada restrição, sem desconsideá-las, foi apresentado: a combinação de restrições, que permitiu a troca de um certo número de nós extras por um melhor tempo total de execução.

Uma segunda extensão à restrição global *Knapsum* foi apresentada: a integração de técnicas de programação dinâmica para o problema da mochila. Esta, apesar de promover a redução do número total de nós, também fez aumentar, incovenientemente, o tempo de execução e foi descartada.

Por fim, mostramos uma comparação entre os dois métodos que se mostraram mais promissores: RMC, o método que implementa o esquema cooperativo apresentado, e RMC+COMBO que faz o mesmo que RMC e implementa a combinação de restrições. Podemos ver que combinar as restrições permite resolver mais rápidamente o problema, apesar da perda de eficiência em termos de número de nós. Também foi apresentada uma comparação entre os métodos desenvolvidos (RMC, RMC+Combo), o pacote comercial ILOG CPLEX 10.1 e o melhor método conhecido para o problema ([Boussier 2009]), que também se baseia em um esquema cooperativo (PM e PPR). Os resultados mostram que os esquemas cooperativos contribuem de maneira essencial para problemas de grande tamanho e que a filtragem da restrição *Knapsum* pode contribuir de maneira eficiente para a solução do problema.

CB1		RMC		RMC+COMBO	
n=100, m=5	Opt	Nós	Tempo	Nós	Tempo
0	24381	2239	0.05	4093	0.04
1	24274	2458	0.04	5847	0.04
2	23551	3284	0.06	7652	0.07
3	23534	13012	0.24	21579	0.23
4	23991	2697	0.04	4627	0.05
5	24613	1471	0.02	3467	0.03
6	25591	481	0.01	1013	0.01
7	23374	4855	0.08	15775	0.08
8	24216	2892	0.04	5324	0.04
9	24411	2525	0.05	4348	0.04
10	42757	1226	0.03	1797	0.02
11	42545	1283	0.02	2413	0.02
12	41968	14642	0.3	24885	0.26
13	45090	4907	0.08	8240	0.07
14	42218	1982	0.04	3264	0.04
15	42927	756	0.02	1312	0.02
16	42009	1102	0.02	1375	0.01
17	45020	6291	0.15	11629	0.14
18	43441	752	0.01	1300	0.01
19	44554	2591	0.05	4962	0.05
20	59822	818	0.02	1755	0.02
21	62081	460	0.01	823	0.01
22	59802	943	0.02	1712	0.02
23	60479	3185	0.06	5431	0.06
24	61091	1163	0.01	2270	0.02
25	58959	2797	0.04	4442	0.04
26	61538	640	0.02	1083	0.02
27	61520	619	0.02	1068	0.02
28	59453	298	0.01	553	0
29	59965	4549	0.08	8026	0.06
Total		86918	1.66	162065	1.52
Média		2897.27	0.06	5402.17	0.05

Tabela 2.1: Resultados para o conjunto CB1.

CB2		RMC		RMC+COMBO	
n=250, m=5	Opt	Nós	Tempo	Nós	Tempo
0	59312	4131	0.1	8154	0.1
1	61472	45845	0.93	66073	0.71
2	62130	17792	0.51	36706	0.48
3	59463	427254	7.58	756061	4.71
4	58951	65491	1.18	105510	1.1
5	60077	81493	2	160081	1.82
6	60414	38931	0.82	63533	0.78
7	61472	116827	2.86	212555	1.39
8	61885	38001	0.72	71383	0.67
9	58959	5904	0.12	8610	0.12
10	109109	68700	1.49	114635	1.34
11	109841	30313	0.63	60570	0.66
12	108508	32242	0.66	57500	0.62
13	109383	111133	1.96	135849	0.41
14	110720	217725	4.02	263737	1.9
15	110256	60983	1.3	94422	1.14
16	109040	39906	0.9	75425	0.83
17	109042	148040	2.82	239309	0.63
18	109971	65442	1.3	129428	1.24
19	107058	69695	1.3	104439	1.19
20	149665	67809	1.48	104410	1.25
21	155944	33979	0.73	52617	0.62
22	149334	70894	1.32	100709	1.12
23	152130	20482	0.39	32140	0.36
24	150353	49986	0.99	71777	0.24
25	150045	2687	0.08	4042	0.06
26	148607	1896	0.04	3102	0.04
27	149782	47743	0.97	78523	0.86
28	155075	12009	0.25	24502	0.22
29	154668	38502	0.68	63446	0.59
Total		2031835	40.15	3299248	27.18
Média		67727.83	1.34	109974.93	0.91

Tabela 2.2: Resultados para o conjunto CB2.

CB3		RMC		RMC+COMBO	
n=500, m=5	Opt	Nós	Tempo	Nós	Tempo
0	120148	2045540	35.8	3779635	37.55
1	117879	273162	5.08	438850	2.9
2	121129	735278	14.65	1235520	16.41
3	120804	665027	12.35	1139052	19.93
4	122319	808408	4.76	1357396	6.58
5	122024	1065197	10.21	1558347	14.47
6	119127	1234183	15.6	2000079	12.4
7	120568	535508	0.57	807554	9.74
8	121586	1339439	35.47	2180327	18.47
9	120717	1149299	18.11	1713407	12.61
10	218428	1220682	3.45	1580956	10.22
11	221202	200656	4.68	257706	3.4
12	217542	2668316	27.29	3915787	40.9
13	223560	1902732	29	2327594	17
14	218966	159856	3.21	193237	2.03
15	220530	903797	17.24	1008362	2.06
16	219989	288596	6.97	409811	5.62
17	218215	386462	9.7	477987	3.06
18	216976	1046934	0.2	1201641	8.14
19	219719	1689949	27.74	2984305	12.56
20	295828	84668	1.54	132809	1.21
21	308086	621192	17.8	702367	1.07
22	299788	196578	3.94	410529	4.95
23	306478	345295	9.3	481944	0.12
24	300342	579462	12.71	737753	1.52
25	302571	267305	5.13	280816	3.16
26	301339	87002	1.56	111800	1.24
27	306454	66903	1.89	94375	1.59
28	302828	505243	10.33	597891	3.17
29	299910	1511467	3.42	1801820	26.24
Total		24584136	349.68	35919657	300.32
Média		819471.2	11.66	1197321.9	10.01

Tabela 2.3: Resultados para o conjunto CB3.

CB4		RMC		RMC+COMBO	
n=100, m=10	Opt	Nós	Tempo	Nós	Tempo
0	23064	219863	6.63	276854	3.04
1	22801	234167	6.82	278942	1.96
2	22131	30387	0.88	36697	0.76
3	22772	285415	9.47	334051	3.32
4	22751	25447	0.73	30302	0.64
5	22777	244182	6.99	290692	0.62
6	21875	59173	1.46	72678	1.26
7	22635	17798	0.57	22379	0.51
8	22511	21227	0.63	26134	0.57
9	22702	88690	2.54	107082	2.24
10	41395	115908	2.84	151223	2.47
11	42344	36467	0.86	46313	0.72
12	42401	72154	1.89	86847	1.66
13	45624	67672	2.04	80112	1.81
14	41884	43428	1.03	50409	0.4
15	42995	143412	3.96	170128	2.22
16	43574	129328	3.59	162741	0.21
17	42970	64662	1.67	82876	1.43
18	42212	130846	3.26	158111	0.64
19	41207	109980	3.18	133344	0.06
20	57375	6231	0.16	7811	0.14
21	58978	63176	1.64	77633	1.45
22	58391	41141	1.06	50207	0.88
23	61966	5584	0.14	7926	0.12
24	60803	8008	0.2	9738	0.15
25	61437	31695	0.96	38293	0.84
26	56377	66761	1.86	78754	0.73
27	59391	6321	0.16	7531	0.14
28	60205	17776	0.46	20798	0.39
29	60633	8806	0.27	11322	0.24
Total		2395705	67.99	2907928	31.61
Média		79856.83	2.27	96930.93	1.05

Tabela 2.4: Resultados para o conjunto CB4.

CB5		RMC		RMC+COMBO	
n=250, m=10	Opt	Nós	Tempo	Nós	Tempo
0	59187	68820062	1906.42	82344112	1568.24
1	58781	26239376	821.12	30876970	648.22
2	58097	14712508	420.31	17419500	379.31
3	61000	95777147	3344.61	109288276	2869.68
4	58092	411124772	1356.47	453387344	10539.89
5	58824	12049322	16.69	13696803	387.17
6	58704	11440471	345.46	13051960	320.55
7	58936	708246353	37614.31	1180410587	33470.08
8	59387	49669846	920.65	57955662	1200.38
9	59208	95271142	2291.56	109773010	2252.11
10	110913	71585416	777.01	47560465	957.27
11	108717	67083437	1537.5	52049690	1228.58
12	108932	45552177	495.49	30879595	668.93
13	110086	195856215	1618.75	121667176	2731.61
14	108485	29950188	871.39	29958507	611.68
15	110845	80498956	1012.67	49593049	1104.78
16	106077	105767405	1922.76	75559550	254.33
17	106686	70835688	31.15	60646045	809.13
18	109829	63528915	1720.3	45123226	663
19	106723	15166970	455.48	11897704	5.36
20	151809	10788107	280.9	9858791	134.71
21	148772	30081219	894.22	24630766	519.07
22	151909	5643235	168.74	4310940	91.33
23	151281	6807788	196.06	49550321	1223.38
24	151948	10016335	325.89	17973553	15.56
25	152109	7046251	203.02	4872382	68.54
26	153131	1101441	28.56	908829	7.56
27	153578	69263867	3133.59	52991753	916.69
28	149160	7704770	193.42	6502489	116.01
29	149704	8451219	255.92	7421253	165.77
Total		2396080598	65160.44	2772160308	65928.92
Média		79869353.27	2172.01	92405343.6	2197.63

Tabela 2.5: Resultados para o conjunto CB5.

n=500, m=10	L	Ótimo	Gap	Tempo L	Provou?
0	117809	117821	-0.0102%	3848	NÃO
1	119229	119249	-0.0168%	217	NÃO
2	119215	119215	0.0000%	198	NÃO
3	118825	118829	-0.0034%	11484	NÃO
4	116530	116530	0.0000%	19762	NÃO
5	119504	119504	0.0000%	2400	NÃO
6	119827	119827	0.0000%	84336	NÃO
7	118344	118344	0.0000%	14582	NÃO
8	117815	117815	0.0000%	3445	NÃO
9	119251	119251	0.0000%	2051	NÃO
10	217377	217377	0.0000%	8	NÃO
11	219077	219077	0.0000%	127	NÃO
12	217847	217847	0.0000%	27	SIM
13	216868	216868	0.0000%	17	NÃO
14	213873	213873	0.0000%	30190	NÃO
15	215086	215086	0.0000%	25	NÃO
16	217940	217940	0.0000%	412	SIM
17	219990	219990	0.0000%	83394	NÃO
18	214382	214382	0.0000%	52	NÃO
19	220899	220899	0.0000%	2094	SIM
20	304387	304387	0.0000%	2431	SIM
21	302379	302379	0.0000%	267	SIM
22	302416	302417	-0.0003%	34	NÃO
23	300784	300784	0.0000%	461	SIM
24	304374	304374	0.0000%	666	SIM
25	301836	301836	0.0000%	901	SIM
26	304952	304952	0.0000%	198	SIM
27	296478	296478	0.0000%	966	SIM
28	301359	301359	0.0000%	3021	NÃO
29	307089	307089	0.0000%	75	SIM

Tabela 2.6: Resultados para o conjunto CB6.

Conjunto	n	m	CPLEX	VBV	RMC	RMC+Combo
CB1	100	5	5.16	0.19	0.06	0.05
CB2	250	5	244.02	99.73	1.34	0.91
CB3	500	5	4219.56	926.7	11.66	10.01
CB4	100	10	39.53	6.37	2.27	1.05
CB5	250	10	N/A	8536.26	2172.01	2197.63

Tabela 2.7: Resumo da média do tempo (em segundos) para cada método sobre cada conjunto de instâncias

CAPÍTULO 3

Problema da Mochila Quadrático

Sumário do capítulo

RESUMO

Apresentamos um estudo feito sobre uma nova técnica de linearização (a qual chamamos t-linearização) para problemas 0-1 com uma função objetivo quadrática e restrições lineares. Esta linearização contrasta com outras técnicas de linearização pelo fato de adicionar uma única variável adicional. Neste trabalho aplicamos a t-linearização ao QKP, Problema da Mochila Quadrático.

Esta linearização permite a solução do QKP de duas maneiras: uma abordagem direta, na qual aplicamos a linearização ao problema 0-1, ou através do cálculo de um limite superior para o problema. No segundo caso, definimos uma relaxação para o problema e calculamos o limite superior a partir desta relaxação. Mostramos, experimentalmente, que o limite superior obtido é comparável ao melhor limite superior conhecido para o problema ([Billionnet 1999]).

O método de enumeração proposto, do tipo branch-and-bound, segue as mesmas características do método apresentado no Capítulo 2, ou seja, um método híbrido entre Programação por Restrições e Programação Matemática. Os experimentos comprovam que nosso método é, até a presente data, o melhor método para a resolução do QKP de baixa densidade.

3.1 Introdução

Uma generalização do problema da mochila, o problema da mochila quadrática (QKP), é o correspondente problema base da programação quadrática. Podemos definir o problema da mochila como o seguinte:

$$(KP) \begin{cases} \max & \sum_{i=1}^{n} c_i x_i \\ s.t. & \sum_{i=1}^{n} a_i x_i \leq b \\ & x_i \in \{0,1\}, i = 1, ..., n \end{cases}$$

onde os coeficientes c_i, a_i ($\forall i = 1, ..., n$) são números reais positivos, assim como o lado direito, b, da única restrição. A inclusão de um objeto na mochila é modelada pelas variáveis de decisão x_i ($i = 1, ..., n$). Para casos onde a combinação entre duas variáveis afeta o processo de decisão com respeito ao valor dos objetos, consideramos o problema da mochila quadrático (QKP). O problema de decisão associado a este problema foi provado NP-completo [Lueker 1975]. Podemos expressar o (QKP) da seguinte forma:

$$(QKP) \begin{cases} \max & \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \\ s.t. & \sum_{i=1}^{n} a_i x_i \leq b \\ & x_i \in \{0,1\} \ i = 1, ..., n \end{cases}$$

onde os coeficientes q_{ij}, c_i, a_i, b ($\forall i = 1, \cdots, n$ e $j = 1, \cdots, n$) são números reais não negativos.

Além de uma extensão do (KP), (QKP) pode ser visto como um caso especial de dois problemas bastante estudados na área da programação quadrática: o problema quadrático com uma única restrição cujos coeficientes são inteiros e o problema quadrático binário com múltiplas restrições [Quadri 2009].

Este problema, introduzido por [Gallo 1980], é a primeira referência na resolução de problemas de Programação Quadrática, o que nos incentiva a utilizá-lo como medida da capacidade do método aqui apresentado.

Existem duas abordagens principais na resolução de um problema de programação quadrática, e, portanto, do (QKP). A primeira consiste na resolução direta do problema quadrático através de técnicas de programação não linear, como pode ser observado em [Billionnet 2004], [Létocart 2009], [Michelon 1996], [Pisinger 2005]. A segunda, na qual nosso trabalho se encaixa, consiste na linearização da expressão quadrática, adaptando o problema para utilização de técnicas de programação inteira, ou seja, o primeiro passo da resolução consiste em transformar o problema quadrático em um equivalente linear sobre o qual trabalharemos em seguida. Em ambos os casos, as técnicas se baseiam no cálculo de limites para o problema (relaxação lagrangeana, relaxação semidefinida, relaxação linear) que serão, em seguida, incorporados a uma pesquisa arborescente do tipo *branch-and-bound*.

Ao contrário da linearização clássica, propomos em nosso trabalho a utilização de uma técnica de linearização que envolve a adição de uma única variável extra, a qual chamamos de t. Esta técnica consiste na conversão do problema original a um problema com uma variável adicional representando o termo quadrático e a adição de restrições lineares para a modelagem do poliedro do termo quadrático.

Essa caracterização é realizada através de um número exponencial de restrições e para sua utilidade prática, sugerimos o uso de uma técnica de geração de restrições. Podemos mostrar que a geração de restrições pode ser feita de forma exata em tempo polinomial para o caso do problema quadrático sem restrições. Porém o acréscimo da restrição da mochila torna o problema de separação a geração de restrições mais difícil e propomos a geração de cortes válidos, cuja separação pode ser feita de maneira heurística. O resultado final deste processo é um modelo linear que será interpretado como uma relaxação do (QKP) e que será usado como base para uma enumeração que visa a exata resolução do problema.

Fazemos aqui uma breve descriçao do algoritmo para resolução do problema proposto. O processo começa com o cálculo de um limite inferior para o problema através de uma heurística. Mais especificamente, a heurística de Billionnet e Calmels [Billionnet 1996], que consiste em um procedimento guloso com reparação. Em seguida, passamos para a fase de branch-and-bound utilizando o limite superior baseado na t-linearização. Similarmente ao que fazemos com o (MKP), acrescentamos a essa enumeração classicamente advinda da Programação Matemática uma camada de Programação por Restrições, ao executarmos a filtragem da Restrição de Custos Reduzidos.

Apresentaremos neste trabalho, além do desenvolvimento teórico do supracitado, uma análise dos resultados, ressaltando a qualidade do limite obtido ao aplicarmos a t-linearização, comparando-a ao melhor limite superior conhecido para o problema (veja Billionnet et al. [Billionnet 1999]); e um comparativo entre o método proposto para resolução ótima do problema com o melhor método apresentado na literatura, devido a Pisinger et al. [Pisinger 2005].

3.2 Técnicas de linearização

A linearização de um problema involve a criação de uma ou diversas novas variáveis e restrições para a modelização da expressão quadrática. Várias técnicas de linearização já foram propostas ao longo dos anos. Como base de comparação utilizamos a linearização clássica, devido à sua popularidade e facilidade de compreensão.

3.2.1 Linearização clássica

A linearização dita clássica consiste na substituição de cada par de variáveis aparecendo como produto na expressão quadrática, digamos $x_i x_j$ por uma nova variável z_{ij}, bem como o seguinte sistema de restrições:

$$\begin{aligned} z_{ij} &\leq x_i, \\ z_{ij} &\leq x_j, \\ z_{ij} &\geq x_i + x_j - 1 \end{aligned}$$

Fica claro que tal esquema de linearização envolve a criação de $\Omega(n^2)$ novas variáveis e restrições. Apesar de ser amplamente estudado e que diversas modificações e melhoras tenham sido propostas ao longo dos anos (exemplos em [Glover 1975], [Chaovalitwongse 2004]), o número de variáveis e restrições final é um empecilho no emprego de tal linearização.

3.2.2 A t-linearização.

A chamada t-linearização é um processo em duas etapas. Primeiro substituimos o termo quadrático da função objetivo por uma variável t e adicionamos uma restrição limitando o valor dessa variável ao valor da expressão quadrática. A etapa seguinte consiste em reescrever tal desigualdade através de um conjunto de desigualdades lineares. Isto pode ser feito através da caracterização do fecho convexo do problema quadrático 0-1 sem restrições (veja [Gueye 2005, Gueye 2009]).

Podemos re-escrever (QKP) da seguinte forma:

$$(LP(Q)) \left\{ \begin{array}{l|l} \max\ t + \sum_{i=1}^{n} c_i x_i & \\ s.t. & \sum_{i=1}^{n} a_i x_i \leq b \\ & t \leq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \\ & t \in \Re,\ x_i \in \{0,1\}\ \ i = 1, ..., n \end{array} \right.$$

sendo c_i, q_{ij}, a_i and b números reais não-negativos.

Os problemas (LP(Q)) e (QKP) são equivalentes. A função objetivo em (LP(Q)) possui uma expressão linear ao passo que uma nova restrição quadrática foi adicionada ao problema. Devemos agora substitui-la por um conjunto de restrições lineares funcionalmente equivalente. Tal resultado se deve ao seguinte teorema, onde definimos $q_{ij} = q_{ji}, \forall 1 \leq i < j \leq n$.

Teorema 3.2.1 *Politopo quadrático.*

Considere os seguintes conjuntos:

$$X = \left\{ (x,t) \in \Re^{n+1} \mid\ t \leq \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j, x \in \{0,1\}^n \right\}$$

e

$$QP_n = \left\{ (x,t) \in \Re^{n+1} \mid\ t \leq \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)},\ x \in [0,1]^n\ \forall\ \pi\ \in\ S_n \right\}$$

onde S_n é o conjuntos de todas as permutações de $\{1, ..., n\}$.

Temos que $QP_n = Conv(X)$, onde $Conv(X)$ representa o fecho convexo do conjunto X.

Prova Primeiramente vamos mostrar que $QP_n \subset Conv(X)$. Seja $(x, t) \in QP_n$. Temos que mostrar que:

$$\exists\, p \in N,\ (x^i, t^i) \in X\ ,\ \beta_i \geq 0\ (0 \leq i \leq p) \mid \sum_{i=0}^{p} \beta_i (x^i, t^i) = (x, t)\ ,\ \sum_{i=0}^{p} \beta_i = 1.$$

Sem perda de generalidade, podemos supor que as componentes de x estão ordenadas de forma decrescente. Sejam $\alpha_1, \alpha_2, ..., \alpha_m$ os valores (também em ordem decrescente) admitidos por cada componente de x, ou seja:

$$x = (\alpha_1, \alpha_1, ..., \alpha_1, \alpha_2, \alpha_2, ..., \alpha_2, ..., \alpha_m, \alpha_m, ..., \alpha_m).$$

Formalmente, denotando k_j o maior índice l tal que $x_l = \alpha_j$ ($l \in \{1, ..., n\}$ e $j \in \{1, ..., m\}$), nós podemos escrever:

$$x_l = \alpha_j \quad , \quad k_{j-1} < l \leq k_j$$

com $k_0 = 0$.

Denotemos x^0 o vetor nulo de R^n e x^i $(1 \leq i \leq m)$ o ponto de R^n definido por:

$$x_l^i = \begin{cases} 1, & \text{se } l \leq k_i \\ 0, & \text{caso contrário} \end{cases}$$

Temos portanto que $x = \sum_{i=1}^{m-1} (\alpha_i - \alpha_{i+1}) x^i + \alpha_m x^m + \alpha_0 x^0$. Logo, considerando $p = m$ e

$$\beta_i = \begin{cases} \alpha_0 = 1 - \alpha_1, & \text{se } i = 0 \\ \alpha_i - \alpha_{i+1}, & \text{se } 1 \leq i \leq p-1 \\ \alpha_m, & \text{se } i = p \end{cases}$$

obtemos que $x = \sum_{i=0}^{p} \beta_i x^i$ com $\beta_i \geq 0$ (pois $1 \geq \alpha_1 > ... > \alpha_m \geq 0$) e $\sum_{i=0}^{p} \beta_i = \beta_0 + (\alpha_1 - \alpha_2) + (\alpha_2 - \alpha_3) + ... + \alpha_m = \beta_0 + \alpha_1 = 1$.

Dado um índice $i = 1, ..., p$ e um escalar $\delta \leq 0$, consideremos o seguinte valor:

$$t_\delta^i = \sum_{l=1}^{n-1} \sum_{u=l+1}^{n} q_{lu} x_l^i x_u^i + \delta = \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{ul} x_l^i x_u^i + \delta = \sum_{l=2}^{k_i} \sum_{u=1}^{l-1} q_{ul} + \delta$$

por construção de t_δ^i e pela definição de X, $(x^i, t_\delta^i) \in X$.

Além disso, por definição $t_\delta^0 = \delta$, temos que $(x^0, t_\delta^0) \in X$. Logo, resta-nos provar que $\exists\ \delta \leq 0$ tal que $t = \sum_{i=0}^{p} \beta_i t_\delta^i$.

Dado que $(x,t) \in QP_n$, temos

$$t \leq \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{\pi(u)\pi(l)} x_{\pi(l)}, \quad \forall\ \pi\ \in\ S_n.$$

Em particular, para a permutação identidade, obtemos a seguinte desigualdade:

$$t \leq \sum_{l=2}^{n} \sum_{u=1}^{l-1} q_{ul} x_l = \sum_{i=1}^{p} \alpha_i \Big(\sum_{l=k_{i-1}+1}^{k_i} \sum_{u=1}^{l-1} q_{ul} \Big).$$

Seja $\delta = t - \sum_{i=1}^{p} \alpha_i \Big(\sum_{l=k_{i-1}+1}^{k_i} \sum_{u=1}^{l-1} q_{ul} \Big)$. Então temos $\delta \leq 0$ e, através de um cálculo longo porém direto, podemos ver que $\sum_{i=0}^{p} \beta_i t_\delta^i = t$. Portanto, (x,t) pode ser escrito como uma combinação convexa de pontos de X.

De forma análoga, temos que mostrar que $Conv(X) \subset QP_n$. Para tanto, é suficiente constatar que qualquer uma das desigualdades que definem QP_n é válida para X, ou seja, temos que mostrar que

$$\forall (x,t) \in X,\ \forall \pi \in S_n \quad t \leq \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}.$$

Como $(x,t) \in X$, temos $t \leq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j$ e, logo, basta estabelecer que

$$\forall (x,t) \in X,\ \forall \pi \in S_n \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \leq \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}.$$

Isso pode ser feito facilmente se percebermos que

$$\begin{aligned} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(i)\pi(j)} x_{\pi(i)} x_{\pi(j)} \\ &= \sum_{i=2}^{n} x_{\pi(i)} \Big(\sum_{j=1}^{i-1} q_{\pi(i)\pi(j)} x_{\pi(j)} \Big) \end{aligned}$$

para qualquer permutação π e levando em consideração que $x_{\pi(j)} \in \{0,1\}$ e $q_{\pi(j)\pi(i)} \geq 0$ (tais que $q_{\pi(j)\pi(i)} x_{\pi(j)} \leq q_{\pi(j)\pi(i)}$).

Considere os seguintes conjuntos:

$$X = \left\{ (x,t) \in \Re^{n+1} \mid t \leq \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j, x \in \{0,1\}^n \right\} \tag{3.1}$$

e

$$Y = \left\{ (x,t) \in \Re^{n+1} \mid t \leq \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j, \sum_{i=1}^{n} a_i x_i \leq b, x \in \{0,1\}^n \right\} \tag{3.2}$$

Podemos definir, à partir de (3.1) e (3.2) os seguintes programas lineares:

$$\begin{aligned} (QKP(X)) \max \quad & \sum_{j=1}^{n} c_j x_j + t \\ \text{s.a:} \quad & \sum_{j=1}^{n} a_j x_j \leq b, \\ & (x,t) \in Conv(X) \\ & x \in \{0,1\}^n, t \in \Re \end{aligned}$$

e

$$\begin{aligned} (QKP(Y)) \max \quad & \sum_{j=1}^{n} c_j x_j + t \\ \text{s.a:} \quad & (x,t) \in Conv(Y) \\ & x \in \{0,1\}^n, t \in \Re \end{aligned}$$

Neste ponto deve estar claro para o leitor que ambos os problemas descritos acima são equivalentes ao (QKP). A partir do resultado demonstrado no teorema 3.2.1, podemos então estabelecer um programa linear inteiro no qual substituimos a parte quadrática pela equivalência linear.

$$(LP) \left\{ \begin{array}{l|l} \max & t + \sum_{i=1}^{n} c_i x_i \\ s.t. & \sum_{i=1}^{n} a_i x_i \leq b \\ & t \leq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(j)\pi(i)} x_{\pi(j)}, \ \forall \pi \in S_n \\ & t \in \Re, \ x_i \in \{0,1\} \ , \ i = 1, ..., n \end{array} \right.$$

Em vista da definição dos conjuntos X e Y e do problema (LP), o resultado a seguir decorre diretamente do Teorema 3.2.1.

Lema 3.2.2 $Y = \left\{ (x,t) \in \Re^{n+1} \mid t \leq \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j, \sum_{i=1}^{n} a_i x_i \leq b, x \in \{0,1\}^n \right\}$ *é o conjunto viável de LP.*

Prova Seja $\bar{Y}$ o conjunto viável de LP e $(\bar{x}, \bar{t}) \in \Re^{n+1}$. Para mostrar que $Y = \bar{Y}$ é suficiente mostrar que:

$$\bar{t} \leq \sum_{1 \leq i < j \leq n} q_{ij} \bar{x}_i \bar{x}_j \Leftrightarrow \bar{t} \leq \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(j)\pi(i)} \bar{x}_{\pi(j)}, \ \forall \pi \in S_n$$

Seja $\bar{\pi}$ tal que ordena as coordenadas de $\bar{x}$ de forma decrescente. Note que:

$$\sum_{1 \leq i < j \leq n} q_{ij} \bar{x}_i \bar{x}_j = \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\bar{\pi}(j)\bar{\pi}(i)} \bar{x}_{\bar{\pi}(i)}$$

e que por outro lado, $\bar{\pi}$ é a permutação que minimiza

$$\sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} \bar{x}_{\pi(i)}.$$

Logo o resultado segue.

Corolário 3.2.3 *$Conv(Y)$ é a envoltória convexa do conjunto viável de (LP).*

O número de restrições em (LP) é $O(n!)$, uma vez que temos uma restrição para cada possível permutação das variáveis. Isso significa que a utilização prática desta formulação deve levar em consideração esta dificuldade. Para tanto, uma maneira conhecida de lidar com um número muito grande de restrições é a técnica de geração de restrições.

A geração de restrições consiste em considerar um subprograma do modelo original, onde somente uma parte pequena das restrições é levada em conta inicialmente. Esse submodelo é resolvido e a partir dessa solução, podemos verificar (de maneira exata ou heurística) se tal solução satisfaz todas as demais restrições. Caso a solução viole uma das restrições originais do problema, devemos exibir um corte válido para o problema tal que a solução corrente não valide. O processo de verificação e cálculo desse corte é chamado de problema de separação. Uma vez resolvido o problema de separação, caso uma restrição violada pela solução corrente tenha sido encontrada, esta pode ser acrescentada ao submodelo e o processo itera. Caso contrário, a solução do submodelo é uma solução de uma relaxação do problema original. Se o problema de separação é resolvido de forma exata, podemos garantir que essa solução é uma relaxação para o problema original.

Dada a solução do subproblema $\bar{x}$, basta ordenar os índices das variáveis de maneira decrescente para encontrar a permutação ótima para o problema de separação de $Conv(X)$. Este resultado é demonstrado conforme o seguinte teorema.

Teorema 3.2.4 *Problema de separação. Dados $(\bar{x}, \bar{t}) \in [0,1]^n \times \mathcal{R}$, achar uma desigualdade que separa $(\bar{x}, \bar{t})$ de $Conv(X)$ ou provar que $(\bar{x}, \bar{t}) \in Conv(X)$ pode ser feito em $O(m + nlogn)$, onde m é o número de coeficientes quadráticos da matriz de coeficientes q.*

Prova Basta notar que para uma dada solução $(\bar{x}, \bar{t}) \in [0,1]^n \times \mathcal{R}$, a permutação $\bar{\pi}$ que minimiza $\sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(i)}$ é tal que $\bar{x}_{\bar{\pi}(1)} \geq \bar{x}_{\bar{\pi}(2)} \geq ... \geq \bar{x}_{\bar{\pi}(n)}$. Para demonstrarmos isto, usamos o fato que todos os coeficientes são positivos e, portanto, $q_{\bar{\pi}(j)\bar{\pi}(i)}\bar{x}_{\pi(i)} \geq q_{\bar{\pi}(j)\bar{\pi}(i)}\bar{x}_{\pi(j)}$ se $\bar{x}_{\bar{\pi}(i)} \geq \bar{x}_{\bar{\pi}(j)}$. Em consequência, uma simples ordenação não crescente dos índices das variáveis segundo os valores de $\bar{x}$ produz uma permutação cuja restrição possui um lado direito de valor mínimo. Esta operação pode ser feita em $O(nlogn)$. Já o cálculo dos coeficientes da restrição passa por no máximo uma vez para cada coeficiente da matriz q, o que prova o resultado do teorema.

Através do Teorema 3.2.1 sabemos caracterizar os pontos em $Conv(X)$ de maneira linear, mas não os pontos em $Conv(Y)$. Ao relaxarmos a restrição de integralidade em LP, obtermos o problema relaxado $\overline{LP}$. Um processo de geração de restrições onde as restrições são geradas conforme o Teorema 3.2.1 constitui, portanto, uma relaxação do problema original, obtendo-se assim um limite superior para o QKP, descrito na Seção 3.3. Porém, mostramos, através do Lema 3.2.2 que o conjunto de pontos que devemos caracterizar para a resolução exata de LP são os pontos em (3.2). Na Seção 3.3 mostraremos como podemos gerar desigualdades válidas mais justas para $Conv(Y)$ do que aquelas que caracterizam $Conv(X)$.

3.2.3 Exemplo

Apresentamos, para efeitos didáticos, um exemplo do funcionamento da t-linearização. Vamos considerar o seguinte problema de base:

$$
\begin{aligned}
(\text{QKP}_{ex}) \quad \max \quad & 91x_1 + 78x_2 + 22x_3 + 4x_4 + 48x_5 \\
& +55x_1x_2 + 23x_1x_3 + 35x_1x_4 + 44x_1x_5 \\
& +92x_2x_3 + 11x_2x_4 + 20x_2x_5 \\
& +7x_3x_4 + 57x_3x_5 \\
& +100x_4x_5 \\
\text{s.a:} \quad & 34x_1 + 33x_2 + 12x_3 + 3x_4 + 43x_5 \leq 50 \\
& x_i \in \{0,1\} \ i = 1, ..., 5
\end{aligned}
$$

Vamos apresentar a geração de uma restrição do tipo:

$$t \leq \alpha_\pi^t x = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{\pi(i)\pi(j)} x_{\pi(j)}. \tag{3.6}$$

para uma permutação π qualquer. Vamos considerar a permutação identidade $\pi = (1,2,3,4,5)$. Portanto, para cada uma das variáveis vamos calcular o coeficiente da variável na restrição.

$$\alpha_{\pi(1)} = \alpha_1 = 0$$
$$\alpha_{\pi(2)} = \alpha_2 = q_{12} = 55$$
$$\alpha_{\pi(3)} = \alpha_3 = q_{13} + q_{23} = 23 + 92 = 115$$
$$\alpha_{\pi(4)} = \alpha_4 = q_{14} + q_{24} + q_{34} = 35 + 11 + 7 = 53$$
$$\alpha_{\pi(5)} = \alpha_5 = q_{15} + q_{25} + q_{35} + q_{45} = 44 + 20 + 57 + 100 = 221$$

Logo a restrição gerada é:

$$t \leq 0x_1 + 55x_2 + 115x_3 + 53x_4 + 221x_5$$

3.3 t-relaxação: um limite superior para o QKP

3.3.1 Aplicação da t-linearização ao QKP

O limite superior que propomos para o QKP é dado pela solução ótima de $\overline{LP}$, que é a relaxação linear de (LP) (i.e. $x_i \in [0,1]$, $i = 1, ..., n$) para a qual somente um conjunto limitado de restrições é considerado. A escolha dessas restrições é um passo fundamental para a resolução dessa relaxação de LP. Em razão disso, propomos um método de geração de restrições. Dada uma permutação π dos índices das variáveis, geramos a restrição (3.6) adequada, de acordo com o Teorema 3.2.1 e testamos se a solução corrente a viola ou não. Caso positivo, adicionamos essa restrição a $\overline{LP}$ e resolvemos ele novamente. Este processo é repetido até encontramos uma permutação para a qual a restrição associada não é violada pela solução corrente de $\overline{LP}$. A Figura 3.1 mostra os principais passos do nosso algoritmo.

O Passo 3 consiste no cálculo de uma permutação π a partir de $\overline{x}$, a solução ótima de $\overline{LP}$. Isso pode ser feito ordenando os índices das variáveis de forma não crescente, de acordo com o valor de $\overline{x}$. Um critério secundário que consideramos é o uso dos custos reduzidos (também em ordem não crescente), visto que a solução da relaxação linear pode ter muitos valores empatados em 0 ou 1.

No Passo 4, checamos se a restrição obtida no Passo 3 invalida a solução $\overline{x}$. Se este for o caso a restrição é adicionada a $\overline{LP}$. Passos 3 e 4 correspondem à resolução do problema de separação feita para $Conv(X)$. Não obstante, nosso objetivo é gerar desigualdades que caracterizam $Conv(Y)$, pois, através do Lema 3.2.2 mostramos que o conjunto viável do problema inteiro é igual a (3.2). Na Seção 3.3.3 mostraremos como utilizar a restrição da mochila para a obtenção de desigualdades válidas mais justas para $Conv(LP)$.

Um esquema do método proposto é apresentado na Figura 1.

1. Encontre uma permutação das variáveis.
   Inicialmente escolhemos uma permutação associada a uma solução viável calculada via heurística de Billionnet e Calmels [Billionnet 1996].
   A restrição correspondente é adicionada a $\overline{LP}$.
2. Resolver $\overline{LP}$. Seja $\bar{x}$ a solução obtida.
3. Resolver o problema de separação. Seja π a permutação encontrada ao ordenarmos as variáveis de maneira decrescente de acordo com $\bar{x}$.
4. Se a restrição associada a π, $t \leq \alpha_\pi x$ é violada por $(\bar{x}, \bar{t})$ então adicionar a restrição a $\overline{LP}$ e voltar para o passo 2; senão parar.

Figura 3.1: Passos para o cálculo do limite superior do QKP.

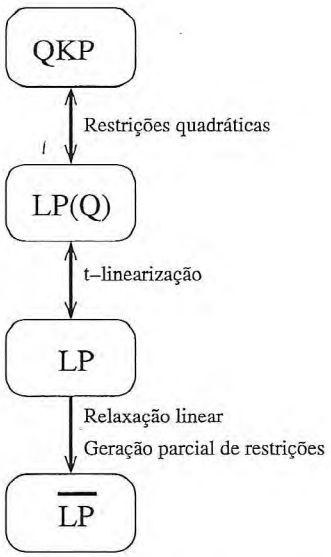

Figura 3.2: Transformações aplicadas ao QKP para o cáculo do limite superior.

3.3.2 Voltando ao exemplo

Inicialmente, dentro de nosso método precisamos de uma solução heurística para o problema. Essa solução é obtida através do método de [Billionnet 1996]. Em nosso exemplo temos como solução o vetor $x = (0, 1, 1, 1, 0)$. Podemos, a patir dessa solução, e seguindo o Teorema 3.2.4, construir uma permutação π_0 ordenando as variáveis segundo os valores da solução heurística. Dentre as diversas permutações equivalentes, escolhemos $\pi_0 = (4, 3, 2, 1, 5)$. Resta agora a construção da restrição $t \leq \sum_{j=1}^{5} \alpha_{\pi_0(j)} x_{\pi_0(j)}$.

Utilizamos então o Teorema 3.2.1 que estabelece a equivalência entre o politopo quadrático e o politopo das permutações. Vamos determiar os coeficientes da restrição da seguinte forma :

$$\begin{aligned}
\alpha_{\pi_0(1)} &= \alpha_4 = 0 \\
\alpha_{\pi_0(2)} &= \alpha_3 = q_{34} = 7 \\
\alpha_{\pi_0(3)} &= \alpha_2 = q_{24} + q_{23} = 11 + 92 = 103 \\
\alpha_{\pi_0(4)} &= \alpha_1 = q_{14} + q_{13} + q_{12} = 35 + 23 + 55 = 113 \\
\alpha_{\pi_0(5)} &= \alpha_5 = q_{45} + q_{35} + q_{25} + q_{15} = 44 + 20 + 57 + 100 = 221
\end{aligned}$$

A restrição obtida é, portanto :

$$t \leq 113x_1 + 103x_2 + 7x_3 + 0x_4 + 221x_5$$

Podemos adicioná-la então ao modelo que vai constituir a primeira iteração do algoritmo de geração de restrições:

$$\begin{aligned}
(\mathrm{QKP}^1_{ex}) \quad \max \quad & 91x_1 + 78x_2 + 22x_3 + 4x_4 + 48x_5 + t \\
\text{s.a:} \quad & 34x_1 + 33x_2 + 12x_3 + 3x_4 + 43x_5 \leq 50 \\
& 113x_1 + 103x_2 + 7x_3 + 0x_4 + 221x_5 \geq t \\
& t \geq 0, x_i \in \{0, 1\} \ \ i = 1, ..., 5
\end{aligned}$$

O modelo acima pode então ser relaxado em suas restrições de integralidade e a relaxação resolvida por um método de resolução de programas lineares. A solução do modelo acima é $\bar{x}^1 = \{0.2058, 0, 0, 0, 1\}, \bar{t}^1 = 264.26$. Podemos agora partir para a segunda permutação, π_1, obtida através da ordenação das variáveis a partir da solução vinda da relaxação linear. No entanto, a relaxação linear nos fornece uma informação complementar: os custos reduzidos. Estes serão usados para desempatar as variáveis que possuem o mesmo valor na solução – por exemplo x_2, x_3, x_4 nesse caso. Os custos reduzidos obtidos são $\bar{c}^1 = \{0, -17, -43, -14, 11\}$. Portanto, temos $\pi_1 = (5, 1, 4, 2, 3)$. A montagem da restrição $t \leq \sum_{j=1}^{5} \alpha_{\pi_1(j)} x_{\pi_1(j)}$ é a seguinte:

$$\alpha_{\pi_0(1)} = \alpha_5 = 0$$
$$\alpha_{\pi_0(2)} = \alpha_1 = q_{15} = 44$$
$$\alpha_{\pi_0(3)} = \alpha_4 = q_{45} + q_{15} = 100 + 35 = 135$$
$$\alpha_{\pi_0(4)} = \alpha_2 = q_{25} + q_{12} + q_{24} = 57 + 55 + 11 = 123$$
$$\alpha_{\pi_0(5)} = \alpha_3 = q_{35} + q_{13} + q_{34} + q_{23} = 20 + 23 + 7 + 92 = 142$$

A restrição a ser adicionada ao modelo é:

$$t \leq 44x_1 + 123x_2 + 142x_3 + 135x_4 + 0x_5$$

O passo seguinte é testar se a solução do modelo que deu origem a essa restrição viola ou não a restrição obtida. Caso a resposta seja positiva, essa restrição é adicionada ao modelo e o algoritmo itera. Caso contrário podemos parar a execução do algoritmo. Ao verificarmos a restrição, constatamos que $\bar{t}^1 = 246.26 > 44\bar{x}_1^1 + 123\bar{x}_2^1 + 142\bar{x}_3^1 + 135\bar{x}_4^1 + 0\bar{x}_5^1 = 9.64$. A restrição corta a solução relaxada $(\bar{x}^1, \bar{t}^1)$ e pode ser adicionada ao modelo que se tornaria:

$$\begin{aligned}
(\text{QKP}^2_{ex}) \quad \max \quad & 91x_1 + 78x_2 + 22x_3 + 4x_4 + 48x_5 + t \\
\text{s.a:} \quad & 34x_1 + 33x_2 + 12x_3 + 3x_4 + 43x_5 \leq 50 \\
& 113x_1 + 103x_2 + 7x_3 + 0x_4 + 221x_5 \geq t \\
& 44x_1 + 123x_2 + 142x_3 + 135x_4 + 0x_5 \geq t \\
& t \geq 0, x_i \in \{0,1\} \ \ i = 1, ..., 5
\end{aligned}$$

a partir do qual podemos iterar o processo de geração de restrições.

3.3.3 Reforçando as restrições.

Dada uma permutação π, a restrição (3.6) não leva em consideração a estrutura do problema para calculá-las. Ou seja, as restrições do tipo (3.6) descrevem o politopo (3.1), mas não o politopo (3.2). Nosso objetivo nesta seção é de gerar cortes válidos para (3.2) à partir das restrições (3.6).

Da maneira apresentada, cada coeficiente $\alpha_{\pi(j)}$ da restrição, na ordem π, pode ser visto como o valor ótimo do seguinte problema:

$$\alpha_{\pi(j)} = \max \left\{ \sum_{i=1}^{j-1} q_{\pi(i)\pi(j)} x_{\pi(i)} \quad : \quad x \in \{0,1\}^{j-1} \right\}$$

o qual pode ser resolvido por uma simples soma de coeficientes precedendo $\pi(j)$ na ordem π. No entanto, a adição da restrição da mochila poderia ser levada em consideração no

cálculo de tais coeficientes, reforçando a restrição. Vamos, portanto, gerar cortes do tipo:

$$t \leq \beta_\pi^t x \tag{3.9}$$

onde cada componente de β_π, $\beta_{\pi(j)}$, pode ser definida da seguinte maneira:

$$\beta_{\pi(j)} = \max \left\{ \sum_{i=1}^{j-1} q_{\pi(i)\pi(j)} x_{\pi(i)} \quad : \quad \sum_{i=1}^{j-1} a_{\pi(i)} x_{\pi(i)} \leq b - a_{\pi(j)}, \quad x \in \{0,1\}^{j-1} \right\}. \tag{3.10}$$

Vamos mostrar agora que (3.9) constituem cortes válidos para $Conv(Y)$.

Proposição 3.3.1 *Seja $\pi \in S_n$ uma permutação e $\beta_\pi \in \Re^n$ tal que:*

$$\beta_{\pi(j)} = \max \left\{ \sum_{i=1}^{j-1} q_{\pi(i)\pi(j)} x_{\pi(i)} \quad : \quad \sum_{i=1}^{j-1} a_{\pi(i)} x_{\pi(i)} \leq b - a_{\pi(j)}, \quad x \in \{0,1\}^{j-1} \right\}.$$

Seja $(\bar{x}, \bar{t}) \in Y$. Então $\bar{t} \leq \beta_\pi^t \bar{x}$.

Prova Seja $(\bar{x}, \bar{t}) \in Y$. Para qualquer subconjunto de índices de $I \subseteq \{1, \cdots, n\}$, nós temos que:

$$\sum_{j \in I} a_j \bar{x}_j \leq b.$$

Portanto, podemos estabelecer a seguinte relação, para todo $i = 1, \cdots, n$:

$$\sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} \bar{x}_{\pi(j)} \leq \max \left\{ \sum_{j=1}^{i-1} q_{\pi(j)\pi(j)} x_{\pi(j)} \ : \ \sum_{j=1}^{i} a_{\pi(j)} x_{\pi(j)} \leq b, \quad x \in \{0,1\}^i \right\}.$$

Particularmente se $\bar{x}_{\pi(i)} = 1$, obtemos:

$$\sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} \bar{x}_{\pi(j)} \leq \max \left\{ \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} x_{\pi(j)} \ : \ \sum_{j=1}^{i-1} a_{\pi(j)} x_{\pi(j)} \leq b - a_{\pi(i)}, \quad x \in \{0,1\}^{i-1} \right\} = \beta_{\pi(i)}$$

Por outro lado, $(\bar{x}, \bar{t}) \in Y$ implica que

$$\bar{t} \leq \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{ji} \bar{x}_i \bar{x}_j = \sum_{i=2}^{n} \sum_{j=1}^{i-1} q_{\pi(j)\pi(i)} \bar{x}_{\pi(j)} \bar{x}_{\pi(i)} \leq \sum_{i=2}^{n} \beta_{\pi(i)} \bar{x}_{\pi(i)}$$

onde a segunda desigualdade sai do fato de que não é necessário considerar os índices i tais que $\bar{x}_{\pi(i)} = 1$.

Experimentos computacionais preliminares mostraram que a geração de cortes do tipo (3.10) pode se tornar muito custosa, em termos de tempo CPU, conforme o tamnho do problema aumenta. Com efeito, para cada coeficiente (3.10), um problema da mochila é resolvido. Trata-se, portanto, da resolução de $O(n)$ problemas NP-difíceis para cada corte gerado.

Propomos então a geração de cortes $\bar{\beta}_\pi$, onde relaxamos a integralidade das variáveis x na definição de cada coeficiente, vistos na equação (3.10). Ao aplicarmos essa relaxação nos deparamos, para cada coeficiente, com um problema que pode ser resolvido em tempo linear [Martello 1990].

$$\bar{\beta}_{\pi(j)} = \max \left\{ \sum_{i=1}^{j-1} q_{\pi(i)\pi(j)} x_{\pi(i)} \quad : \quad \sum_{i=1}^{j-1} a_{\pi(i)} x_{\pi(i)} \leq b - a_{\pi(j)}, \quad x \in [0,1]^{j-1} \right\}. \tag{3.11}$$

Esta técnica provou-se bastante eficiente para instâncias do QKP de alta densidade, ou seja, quando a matriz de coeficientes quadráticos q é densa (ver Seção 3.5).

3.4 Um branch-and-bound baseado na t-linearização.

O método de *branch-and-bound* proposto para a resolução do problema linearizado LP(Q), e portanto do (QKP), começa por encontrar uma solução heurística factível para o problema, x_H. A heurística escolhida é aquela apresentada por Billionnet e Calmels [Billionnet 1996], cuja eficácia foi comprovada em [Caprara 1998]. A solução x_H nos permite a construção de uma primeira permutação de índices π_H dada por uma ordem não crescente dos valores de x_H. Evidente que, como os valores se limitam a 0 ou 1, teremos bastantes empates. O critério de desempate é a razão $\frac{c_i + \sum_{j=1}^{n} q_{ij}}{a_i}$, definida para cada variável x_i $\forall i = 1, ..., n$. Essa razão é uma superexpectativa do valor que cada variável pode contribuir ao valor da função objetivo, caso o objeto correspondente seja escolhido. A restrição (3.9) construída é então adicionada ao nó raiz do problema.

Daí, em cada nó da árvore de busca, um limite superior U é calculado utilizando-se o método de geração de restrições descrito na Seção 3.3. Em seguida, similarmente ao procedimento usado na resolução do (MKP) passamos a uma fase de fixação, ou seja, a camada de programação por restrições. Ao final da geração de restrições, temos, na última iteração, uma solução $\overline{x}$ com custos reduzidos $\overline{c}$. Podemos, portanto, estabelecer uma restrição implícita de custos reduzidos. A definição de tal restrição, bem como a filtragem definida para ela podem ser vistas na Seção 2.2.

Ainda antes de iniciarmos o procedimento de ramificação, tendo em vista que o procedimento de geração de restrições pode ter gerado muitas restrições, é comum se fazer uma seleção das restrições a serem guardadas para os nós descendentes. Em nosso caso, optamos por fazer uma agregação das restrições de forma a passar somente uma restrição aos nós subsequentes, em adição à restrição da mochila. Isto vai diminuir de maneira

drástica o tamanho total do modelo permitindo assim que mais nós da árvore de decisão possam ser guardados na memória ao mesmo tempo. Em contrapartida, algumas restrições podem ter que ser geradas mais de uma vez nos nós filhos.

O procedimento de agregação para um conjunto de índices de restrições K consiste na substituição de tais restrições por uma nova restrição $t \leq \tilde{\alpha}_K x$ de tal forma que:

$$\tilde{\alpha}_K(j) = \sum_{i \in K} \frac{\overline{\mu_i} \cdot \alpha_{\pi_i}(j)}{\sum_{k \in K} \overline{\mu_k}}$$

onde $\overline{\mu_i}$ é o valor dual resultante da resolução do modelo linear relativo à restrição $t \leq \alpha_{\pi_i} x$. Seja $K^\star$ o conjunto de índices de todas as restrições geradas em um dado nó. Podemos mostrar que $\sum_{k \in K^\star} \overline{\mu_k} = 1$ (coeficiente de t na função objetivo). Logo, simplificamos a expressão para cada $\tilde{\alpha}_{K^\star}(j)$:

$$\tilde{\alpha}_{K^\star}(j) = \sum_{i \in K^\star} \overline{\mu_i} \cdot \alpha_{\pi_i}(j).$$

Além disso, podemos mostrar que a relaxação linear do modelo na qual todas as restrições $t \leq \alpha_{\pi_i} x$ for $i \in K^\star$ foram substituidas por $t \leq \tilde{\alpha}_{K^\star} x$ tem o mesmo valor ótimo que o modelo com todas as restrições. Para provar isto considere o problema no qual a variável x está fixada a $\overline{x}$, a solução linear. Então temos que $\overline{t} = \max\{t : t \leq \alpha_{\pi_i} \cdot \overline{x}, \forall i \in K^\star\}$. Pelo Teorema da Dualidade (forte)

$$\sum_{i \in K^\star} (\alpha_{\pi_i} \cdot \overline{x}) \cdot \overline{\mu_i} = \overline{t}.$$

Sabemos que $c^T \overline{x} + \overline{t}$ é o valor ótimo do problema t-relaxado. Portanto, a única restrição passada aos nós filhos é $t \leq \tilde{\alpha}_{K^\star} x$.

A regra de ramificação segue o clássico critério de ramificação sobre a variável com o mais próxima de 0.5.

3.5 Resultados Computacionais

Apresentamos nesta seção diversos experimentos computacionais a fim de atestar tanto a qualidade do limite superior proposto quanto do método para solução exata. Para tanto, utilizamos as mesmas instâncias de Billionnet et al. [Billionnet 1999] e o mesmo gerador de instâncias usado em Pisinger et al [Pisinger 2005].

Os resultados computacionais dizem respeito a: (i) a qualidade do limite superior comparado ao valor ótimo para um conjunto de instâncias e o tempo computacional necessário para obtê-lo face aos resultados publicados por Billionnet et al. [Billionnet 1999]; (ii) a performance computacional de diversas versões do algoritmo branch-and-bound apresentado comparado ao melhor resultado conhecido para o QKP [Pisinger 2005].

Tabela 3.1: Comparação da qualidade dos limites superiores.

Método	Ótimo	[Billionnet 1999]		t-relaxação		t-relaxação integral		
No.		LS	gap(%)	LS	gap(%)	LS	gap(%)	NRG
1	18558	18910.56	1.90	19124.1	3.05	18865	1.65	109
2	56525	56574.63	0.09	56576	0.09	56525	0.00	32
3	3752	3807.68	1.48	3900.53	3.96	3785	0.88	312
4	50382	50448.08	0.13	51064.5	1.35	50589	0.41	580
5	61494	61623.22	0.21	61621	0.21	61494	0.00	6
6	36360	36464.87	0.29	36654.9	0.81	36399	0.11	283
7	14657	14749.58	0.63	14853.5	1.34	14657	0.00	445
8	20452	20525.15	0.36	20528.5	0.37	20452	0.00	46
9	35438	35485.16	0.13	35487	0.14	35438	0.00	361
10	24930	25191.5	1.05	25496.9	2.27	20190	1.04	401
Média	**32255**	**32378.04**	**0.63**	**32530.7**	**0.85**	**32339.6**	**0.26**	**257.5**

Nossos métodos foram programados utilizando linguacem C++, e todos os experimentos foram conduzidos num Intel Pentium 4 2.66 GHz com 1 GB RAM, exceto os resultados relativos a [Billionnet 1999] e [Pisinger 2005].

3.5.1 Comparação entre os limites superiores

A qualidade do limite superior dado pela t-linearização é comparada à qualidade do limite superior sugerido por Billionnet et al [Billionnet 1999]. Além disso comparamos o resultado com o modelo da t-relaxação final com restrições de integralidade, ou seja, após a geração de todas as restrições da t-relaxação, adicionamos as restrições de integralidade. Chamamos esse model de t-relaxação integral. É importante notar que usamos em todos os experimentos os cortes mais fortes, ou seja, aqueles que foram reforçados com as técnicas apresentadas na Seção 3.3.3.

A comparação entre esses três procedimentos pode ser vista na Tabela 3.1. O tempo CPU necessário para obter o limite superior da t-relaxação comparado à versão integral estão expressos na Tabela 3.2. No caso de [Billionnet 1999], os experimentos foram conduzidos em um HP9000. A diferença entre as duas máquinas é muito grande para garantir uma comparação de tempos CPU justa e representativa. Por isso os tempos de [Billionnet 1999] foram omitidos nas tabelas.

Os três métodos foram testados com o mesmo conjunto de instâncias, cada uma com 100 variáveis e 25% de densidade na matriz do termo quadrático. Tais instâncias foram apresentadas em [Billionnet 1999].

A Tabela 3.1 mostra os limites superiores obtidos para cada método e o desvio percentual (denotado pela coluna *gap* em de cada método em relação ao valor ótimo (mostrado na coluna *Ótimo*). Além disso, o número de restrições geradas pelo procedimento de geração da t-linearização pode ser observado na coluna NRG.

O limite dado por [Billionnet 1999] é de melhor qualidade que o limite provido pela t-relaxação. Não obstante, esta é capaz de dar um limite, em média, a menos de 1% do valor ótimo. Quando considerada a integralidade das variáveis, a t-relaxação integral fornece um limite superior claramente melhor. Nesse caso o gap médio fica em apenas 0.26%. Podemos afirmar também que o número de restrições geradas é bem menor que o

Tabela 3.2: Comparação dos tempos CPU requeridos por ambos limites superiores t-relaxação e t-relaxação integral

Instâncias No.	t-relaxação	t-relaxação integral
1	36.42	41.77
2	0.02	0.03
3	115.11	115.55
4	9.16	15727.48
5	0.0	0.01
6	0.54	23.92
7	38.78	39.41
8	1.34	1.43
9	0.97	2.65
10	278.06	4927.69
Média	**48.04**	**2087.94**

número total de permutações possíveis ($O(n!)$). Tais dados nos permitem concluir que o método proposto é capaz de descrever a vizinhaça da solução ótima com poucas restrições, comparado às $O(n^2)$ restrições da linearização clássica.

A Tabela 3.2 mostra que a t-relaxação integral consome bastante tempo. Isso mostra que a utilização prática de tal método fica bastante comprometida. Ao invés disso preferimos utilizar, em nosso branch-and-bound, a versão com variáveis contínuas. A geração de um grande número de restrições se mostra uma dificuldade na resolução de problemas inteiros. Por isso, veremos experimentalmente que ao limitarmos o número de restrições geradas a cada nó teremos uma melhor performance geral para a resolução exata do QKP.

3.5.2 Métodos para solução exata

Comparamos nesta seção o desempenho do algoritmo exato proposto, um *branch-and-bound* baseado na t-relaxação, com o método enumerativo apresentado por Pisinger et al. [Pisinger 2005]. Um limite de tempo de três horas foi imposto para a resolução de cada instancia para efeitos de comparação com o supracitado.

Todas as instâncias foram geradas de acordo com a literatura relativo ao problema (ver [Billionnet 2004], [Michelon 1996], [Caprara 1998] e [Pisinger 2005]). Coeficientes c_i e q_{ij} da função objetivo são números inteiros distribuidos de maneira uniformemente aleatória entre 0 e 100. Os coeficientes da restrição a_i são distribuidos entre 0 e 50. O lado direito da restrição , b, é um número aleatório entre 50 e $max\{50, \sum_{i=1}^{n} a_i\}$. Os tempos CPU apresentados nas tabelas que seguem são uma média do tempo de solução de 10 instâncias em um dado conjunto.

Elegendo uma versão. Conforme apresentado na Secção 3.3, nosso modelo é resolvido através de um processo de geração de restrições no qual as restrições do fecho convexo quadrático (t-linearização) são adicionadas. Dois tipos de restrições podem ser geradas, dependendo se levamos ou não em consideração a restrição da mochila. Antes de proceder a comparações com outros métodos, avaliamos qual das duas alternativas nos dá um melhor equilíbrio entre a qualidade do limite superior e o tempo para calculá-lo. O critério de comparação é o tempo total para resolução da instancia. Além disso, uma outra decisão que temos que tomar é sobre o número de restrições. Visto que esse número

Tabela 3.3: Tempo médio para instâncias de 100 e 200 variáveis (em segundos).

	versão fraca		versão forte	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS100.25	0.37	10	0.78	10
GHS100.50	3.17	10	1.54	10
GHS100.75	0.34	10	0.41	10
GHS100.100	0.25	10	0.2	10
Média	**1.03**	100%	**0.73**	100%
	versão fraca		versão forte	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS200.25	18.92	10	42.32	10
GHS200.50	9.84	10	13.09	10
GHS200.75	61.77	10	27.84	10
GHS200.100	1236.95	10	397.11	10
Média	**331.87**	100%	**120.09**	100%

é teoricamente exponencial, mas que um pequeno número é suficiente para uma boa descrição do poliedro, procederemos a testes numéricos para determinar uma quantidade adequada de restrições a gerar em cada nó.

Primeiramente implementamos duas versões do nosso algoritmo, a qual chamamos versão fraca e verão forte, dependendo de considerarmos ou não a restrição da mochila na geração de restrições. Ou seja, a versão fraca adiciona as restrições do tipo (3.6), enquanto a versão forte utilisa os procedimentos de reforço sugeridos (3.9).

Ambas versões são capazes de chegar aproximadamente aos mesmos resultados, conforme mostrado nas Tabelas 3.3 e 3.4. No entanto, a versão forte consegue resolver mais instâncias otimamente para o conjunto de 300 variáveis, dentro do limite de tempo estabelecido. Nos casos em que ambas as versões resolvem o mesmo número de instâncias, a versão forte tem melhor desempenho que a versão fraca em média. É preciso esclarecer que em ambas versões o número de restrições geradas a cada nó foi fixado a 10.

Em um segundo momento, interessamo-nos em determinar quantas restrições devemos gerar em cada nó. O objetivo é ver como o crescimento do número de restrições geradas poderia ajudar no tempo de execução final do nosso método. Decidimos limitar o número de restrições geradas de acordo com o tamanho da instancia (número de variáveis).

Outros fatores que poderiam ser considerados dizem respeito a densidade da matriz de coeficientes do termo quadrático. Quanto mais denso o problema, mais difícil para que uma técnica de linearização o represente. Logo, possivelmente mais restrições deveriam ser adicionadas para um limite superior de qualidade, quando comparado a uma instancia menos densa.

Um outro estudo que poderia ser feito diz respeito a como esta organizada essa matriz de coeficientes. Podemos definir um grafo de acordo com essa matriz da seguinte forma.

Tabela 3.4: Tempo médio para instâncias de 300 e 400 variáveis (em segundos).

	versão fraca		versão forte	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS300.25	40.7	10	74.92	10
GHS300.50	1926.4	9	1480.18	9
GHS300.75	2143.74	10	2134.21	10
GHS300.100	35.43	8	2163.94	9
Média	**1066.63**	92.5%	**1444.43**	95%
	versão fraca		versão forte	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS400.25	49.18	10	87.97	10
GHS400.50	1526.18	10	1468.31	10
GHS400.75	2049.02	9	2698.21	9
GHS400.100	2000.87	9	516.45	9
Média	**1427.67**	95%	**1170.92**	95%

Cada variável representa um vértice do grafo. A existência (e o peso) de uma aresta entre dois vértices é determinada pelo coeficiente da matriz quadrática. A conectividade desse grafo, o tamanho da maior componente conexa, bem como o tamanho da clique máxima poderiam igualmente fornecer indicações sobre a dificuldade do problema.

Para estes testes, fizemos duas versões idênticas do nosso programa que poderiam ser diferenciadas somente pelo número máximo de restrições geradas a cada nó. Primeiramente calculamos um coeficiente $\gamma = N/100$. A versão *linear* do algoritmo gera no máximo $\gamma \times 10$ restrições, enquato a versão *quadrática* gera no máximo $\gamma^2 \times 10$ restrições. Para referência, usamos a versão forte de nosso método, ou seja, aquela que considera a restrição da mochila no cálculo dos coeficientes.

Na Tabela 3.5 os experimentos mostram que o crescimento do número de restrições geradas não é capaz de melhorar o tempo de computação final. Em geral a versão linear é capaz de resolver mais instâncias e tem melhor desempenho conforme o número de variáveis aumenta. No entanto ambas ficam aquém do desempenho obtida pela versão que limita o número de restrições máximas a cada nó a 10.

Como consequencia, escolhemos como nossa melhor versão aquela que utiliza o reforço proposto para o cálculo dos coeficientes e que limita em apenas 10 restrições o número máximo em cada nó.

Comparando com os resultados da literatura. Contrastamos os resultados do nosso algoritmo com os melhores resultados conhecidos para o QKP. Esses resultados são obra de Pisinger et al. [Pisinger 2005]. É importante notar que o conjunto de instâncias utilizados na comparação não é o mesmo, embora sejam gerados da mesma forma. A média de 10 instâncias ainda nos parece uma boa medida, mesmo se a variância na dificuldade das instâncias possa ser comprovada. Outro ponto diferente é o ambiente

Tabela 3.5: Tempo médio para instâncias de 200, 300 e 400 variáveis (em segundos).

	progressão linear		progressão quadrática	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS200.25	63.32	10	282.86	10
GHS200.50	23.57	10	45.88	10
GHS200.75	71.79	10	158.46	10
GHS200.100	222.07	10	143.36	8
Média	**95.19**	100%	**158.39**	95%
	progressão linear		progressão quadrática	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS300.25	350.17	10	560.63	10
GHS300.50	5527.28	9	411.37	7
GHS300.75	1316.63	9	4945.91	10
GHS300.100	1171.85	9	135.07	8
Média	**2044.42**	92.5%	**1690.3**	87.5%
	progressão linear		progressão quadrática	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS400.25	2441.33	10	2232.77	10
GHS400.50	2728.69	8	6890.34	8
GHS400.75	4345.18	8	12281.94	8
GHS400.100	296.25	9	1601.46	9
Média	**2390.59**	87.5%	**5431.97**	87.5%

Tabela 3.6: Tempo médio para instâncias de 100 e 200 variáveis (em segundos).

	t-linearização		[Pisinger 2005]	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS100.25	0.78	10	210.7	10
GHS100.50	1.54	10	54.2	10
GHS100.75	0.41	10	6.7	10
GHS100.100	0.2	10	2.7	10
Média	**0.73**	100%	**68.56**	100%

	t-linearização		[Pisinger 2005]	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS200.25	42.32	10	860	9
GHS200.50	13.09	10	168.9	10
GHS200.75	27.84	10	23	10
GHS200.100	397.11	10	76.5	10
Média	**120.09**	100.0%	**267.28**	97.5%

computacional. Pisinger et al. [Pisinger 2005] trabalharam com um Pentium IV 2.4GHz com 1GB RAM. Este ambiente é, não obstante, suficientemente próximo do nosso para permitir comparações numéricas.

Nas Tabelas 3.6 e 3.7 podemos ver que nosso branch-and-bound funciona melhor que o método proposto em [Pisinger 2005] para os problemas menores ($N = 100$), para qualquer densidade. Para 200 variáveis, apenas os problemas com 50% ou menos de densidade funcionam melhor com nosso método, porém, em média o tempo de cálculo é 2 vezes menor. Para 300 e 400 variáveis o preprocessamento de redução da instancia, usando o limite superior de Billionnet et al. [Billionnet 1999], torna o problema pequeno o suficiente para que Pisinger et al. possam resolvê-lo de maneira eficaz em média. Ainda assim, para as instâncias de menor densidade, a t-relaxação se mostra o método mais eficiente.

Apresentamos na Figura 3.3 um comparativo de como cada método aqui comparado gasta seu esforço computacional, segundo a densidade da matriz de coeficientes quadráticos. Percebemos que [Pisinger 2005] passa a maior parte do seu tempo em instâncias de baixa densidade, enquanto que o método proposto encontra mais dificuldade com as intâncias densas.

3.6 Conclusão

Neste seção apresentamos um algoritmo do tipo branch-and-bound para a resolução do QKP de forma exata. Para tanto sugerimos um cálculo de limite superior original, baseado na t-linearização.

Tabela 3.7: Tempo médio para instâncias de 300 e 400 variáveis (em segundos).

	t-linearização		[Pisinger 2005]	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS300.25	74.92	10	4031	10
GHS300.50	1480.18	9	556.8	10
GHS300.75	2134.21	10	94.7	10
GHS300.100	2163.94	9	90.3	10
Média	**1444.43**	95%	**1193.2**	100%

	t-linearização		[Pisinger 2005]	
Conjunto	Tempo	Ótimo	Tempo	Ótimo
GHS400.25	87.97	10	1190.1	9
GHS400.50	1468.31	10	978.3	10
GHS400.75	2698.21	9	275.0	10
GHS400.100	516.45	9	173.2	10
Média	**1170.92**	95%	**640.41**	97.5%

Este esquema de linearização adiciona apenas uma única variável, contrariamente à linearização clássica que envolve a adição de $\Omega(n^2)$ variáveis. No entanto a t-linearização implica em um número de restrições exponencial. Tal empecilho é contornado por nosso método através da técnica de geração de restrições, para o qual mostramos um algoritmo polinomial para o problema de separação.

O limite superior fornecido pode ser considerado competitivo em relacção ao melhor limite superior conhecido [Billionnet 1999], ficando a menos de 1% do valor ótimo para o conjunto de instâncias selecionadas.

Nosso branch-and-bound foi comparado ao melhor método conhecido para o QKP [Pisinger 2005] e se mostrou claramente superior para instâncias de baixa densidade (25%) e para instâncias de menor tamanho (100 e 200 variáveis). Para os demais casos, o pré-processamento de redução proposto por Pisinger et al. consegue reduzir o tamanho da instância de maneira eficiente, possibilitando assim melhores resultados.

O valor de nosso trabalho reside na inovação do método proposto, corroborado pelos resultados competitivos obtidos por uma primeira aplicação dessa técnica, o que possibilita futuras explorações nesta mesma direção.

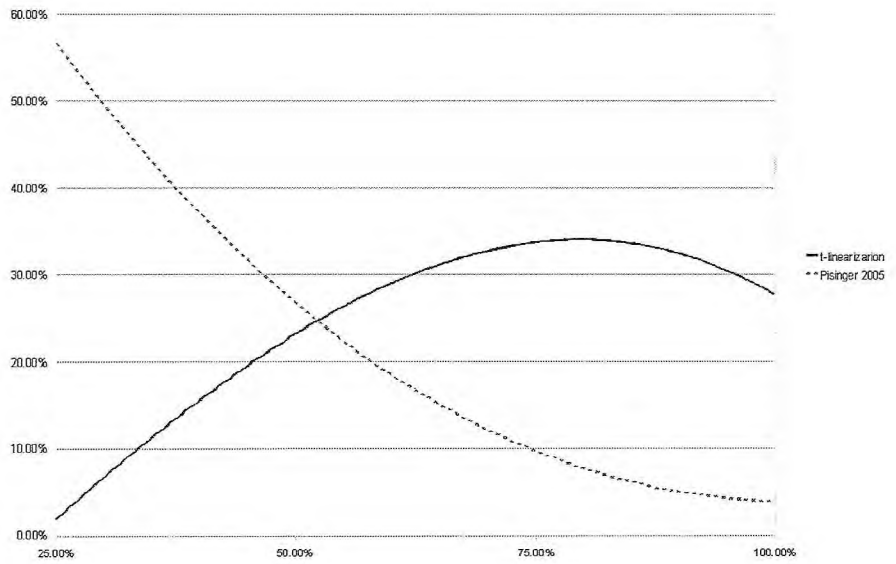

Figura 3.3: Comparação entre a porcentagem de tempo gasto na resolução de instâncias segundo a porcentagem.

CAPÍTULO 4

Problema Prático: à procura de um alvo inteligente

Sumário do capítulo

RESUMO

Um modelo para o problema de busca de um alvo inteligente é o assunto deste capítulo. Primeiramente, apresentamos essa classe de problemas, sua origem e suas variantes. Em seguida, posicionamo-nos em relação à literatura para a identificação do problema que tratamos. De uma maneira geral, o problema consiste em definir um plano de busca com o objetivo de achar (ou maximizar a probabilidade de achar) um alvo que é sensível às pesquisas realizadas.

Propomos um modelo para essa classe de problemas e mostramos a modelagem de restrições para um caso específico, um problema prático realizado com o Ministério da Defesa francês. O modelo desenvolvido tem por base a discretização espaço-temporal e a criação de um conjunto de alvos determinísticos para simular as possibilidades reais do alvo.

Uma análise das características do modelo é estabelecidada, assim como a determinação de um número teórico de alvos determínisticos necessários para simular o comportamento do alvo real, segundo uma confiança estatística e uma margem de erro desejadas. A natureza do problema, assim como o tamanho do modelo não permite que o problema seja resolvido diretamente, sendo uma decomposição necessária para a resolução prática. Em nosso caso escolhemos o método de Janelas Deslizantes como forma de decomposição.

Em razão das capacidades computacionais das máquinas atuais, mostramos experimentalmente que esse número de alvos teórico não pode ser utilizado na prática. No entanto, fomos capazes de mostrar que mesmo com um número pequeno de alvos, a probabilidade dada pelo modelo é próxima da probabilidade do simulador, o qual foi concebido para a validação do modelo.

Essa abordagem constitui, a nosso conhecimento, o primeiro modelo matemático capaz de dar uma resposta precisa ao problema.

4.1 Introdução

A Teoria da Busca nasceu nos anos 50 em um contexto militar particular. Na época, pós segunda guerra mundial, uma grande tensão girava em torno de possíveis ataques marítimos, possivelmente efetuados por submarinos. Após a divisão bipolar do globo, existia, na realidade, uma grande guerra não declarada por informações à respeito do inimigo. Há relatos, por exemplo, de avistamento de submarinos espiões russos ao longo de toda a costa americana [Koopman 1980].

Em vista dessa situação, os governos de ambos os lados procuravam uma maneira de neutralizar o inimigo da melhor forma possível. Uma parte desse problema consistia na busca por aparelhos inimigos. Matemáticos procuravam modelar os diversos problemas de otimização associados a tal contexto. Embora inicialmente essas pesquisas fossem classificadas como secretas, mais tarde, nos anos 60, elas surgem e são absorvidas pela popular (à época) Teoria dos Jogos.

Não se trata, no entanto, de uma simples coincidência que essa nova disciplina tenha sido vista, em princípio, como um jogo. É fácil identificar, de fato, dois atores principais do problema: um pesquisador e um alvo. O pesquisador tem por objetivo, através dos meios que possui, encontrar o alvo em um espaço e tempo definidos. O alvo, por sua vez, utiliza suas capacidades para não ser descoberto. Tal problemática se tornaria popular para as sociedades ocidentais a ponto de ser difundida como jogos para crianças, como o Batalha Naval, por exemplo. Para a sociedade, esse campo encontra diversas aplicações práticas que vão desde a segurança (procura de fugitivos, patrulha), busca de sobreviventes em um ambiente hostil, até a erradicação de um vírus em um ambiente computacional.

Para os matemáticos é um vasto campo de pesquisa, dando espaço a diversas variantes do problema, segundo as capacidades e objetivos de cada um dos atores do problema, bem como das condições nas quais deve ser feita a pesquisa. Inicialmente Koopman [Koopman 1956b, Koopman 1957] estudou o caso do alvo estático, onde uma posição lhe é atribuída inicialmente em um espaço limitado e o pesquisador está sujeito à restrições para encontrá-lo (número de tentativas, combustível, tempo são limitações comumente encontradas na literatura [Washburn 2001]). Em seguida, encontramos o problema conhecido como *Flamming Datum* (ou jogo do Helicóptero contra Submarino). Nesse problema a posição inicial do alvo móvel é conhecida, porém a busca só pode começar um certo tempo depois. O *Flamming Datum* permanece um problema de grande interesse da comunidade científica e militar.

As diversas subcategorias possíveis para o problema podem ser classificadas segundo as características apresentadas:

- **A mobilidade do alvo.**
  Apesar da sua origem militar, que trata de um alvo móvel, o problema do alvo estático é o primeiro a surgir na literatura [Koopman 1956b, Koopman 1957]. Evidentemente o alvo móvel acrescenta um nível de complexidade suplementar ao problema. Além disso, podemos fazer uma distinção de dificuldade significativa caso o alvo siga um movimento markoviano (ver Seção 4.1.1).

- **O espaço e tempo de busca.**
  O espaço definido para o problema pode ser caracterizado por três conjuntos: o espaço do pesquisador K^p, o espaço do alvo K^s e o espaço de busca K^b. Se $K^s \backslash K^b \neq \emptyset$ dizemos que o problema possui zonas de segurança para o alvo. Além disso, existem variantes para o problema que consideram o espaço de busca unidimensional (sobre uma fita) ou um espaço de busca infinito [Baston 1989, Garnaev 1993].

  Em relação ao tempo, existem dois casos distintos: o caso finito e o caso infinito. Não obstante, a convergência de uma determinada estratégia não é garantida, mesmo no caso infinito. [Thomas 1991] mostra que outros fatores podem influenciar nesse resultado. O tempo inicial também é um fator que pode caracterizar o problema. Por exemplo, para o problema *Flamming Datum*, o tempo inicial do alvo é diferente do tempo inicial do pesquisador.

- **Cooperatividade ou adversidade.**
  Distinguimos o caso cooperativo [Bienstock 1991, Fomin 1998], onde o alvo tem interesse de ser encontrado, do caso adversativo, onde o alvo deve permanecer escondido. O primeiro caso é de particular interesse de instituições públicas responsáveis por missões de busca e resgate.

- **O objetivo do alvo.**
  O alvo pode apresentar uma missão a ser cumprida ou um comportamento que influencia suas decisões. Podemos ressaltar nesse caso o problema *Flamming Datum*, onde o alvo vai empregar uma manobra de evasão (ver Seção 4.1.1). Outro caso recorrente na literatura é o caso onde o alvo deve cruzar uma zona do espaço de busca [Bienstock 1991, L. Blin 2008].

- **A inteligência do alvo.**
  A inteligência do alvo é a capacidade que este tem de tomar consciência das ações efetuadas pelo pesquisador. Este conhecimento pode ser total (nesse caso o alvo conhece todas as pesquisas efetuadas) ou parcial, geralmente definido por um raio de alcance ou uma vizinhança de sua posição.

- **Os recursos de pesquisa.**
  Para cada caso na prática, temos um tipo de recurso diferente. Esses recursos definem como é feita a pesquisa e são um fator fundamental na definição do problema. Inicialmente a literatura só considerava casos onde o pequisador possuia um único recurso re-utilizável por uma quantidade pré-definida de vezes [Brown 1980, Washburn 1983]. Nos últimos anos, interessa-se também por recursos alocáveis, ou seja, recursos que podem ser depositados em um certo ponto do espaço de busca e que definem duas vizinhanças: uma subzona de busca, a qual eles são capazes de enxergar; e uma subzona de comunicação, na qual o pesquisador deve estar inserido para obter a informação relativa ao recurso alocado. Recentemente,

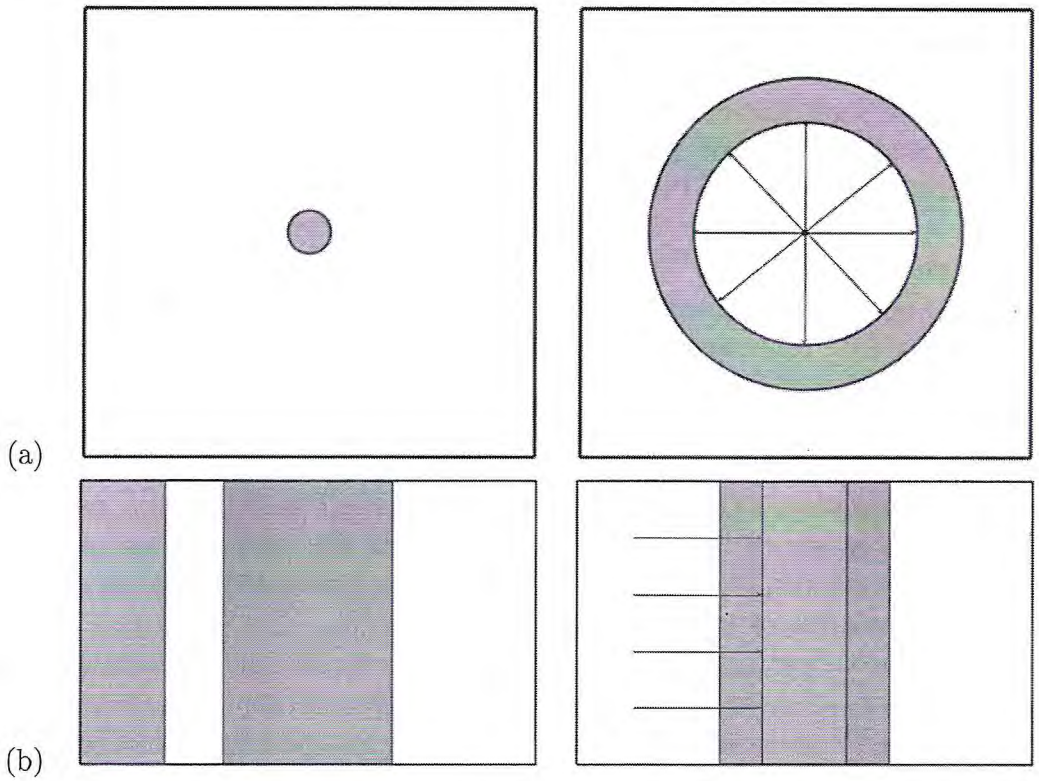

Figura 4.1: Exemplos de missões do alvo. (a) O problema *Flaming Datum*. (b) Um problema onde o alvo deve passar por uma zona onde ele pode ser descoberto.

a literatura também considera casos onde o pesquisador é capaz de utilizar diferentes tipos de recursos, portando diferentes capacidades de busca, comunicação ou alocação.

- **A monotonicidade da pesquisa.**
  Um outro fator que influencia a definição do problema diz respeito à perenidade da busca efetuada. Se uma região do espaço pesquisado não pode ser ocupada novamente pelo alvo, dizemos que a busca é monotônica. Um exemplo prático desse caso é a erradicação de um vírus em uma rede de computadores. Uma vez que o un nó tenha sido desinfectado, supomos que o vírus (o alvo, nesse caso) não pode novamente atacá-lo.

Um resultado interessante em relação à monotonicidade do alvo se encontra na Teoria dos Grafos. Se considerarmos uma busca monotônica, o problema de encontrar o alvo, minimizando o número de buscas efetuadas (ou recursos com tempo de vida unitário) pode ser reduzido ao problema de largura em árvore quando a posição do alvo é revelada. Caso o alvo esteja sempre escondido, teremos uma redução ao problema de largura em caminhos [Bienstock 1991].

Ao longo dos anos essa classe de problemas (jogos de busca) despertou o interesse de outras diversas áreas de pesquisa: a programação estocástica, métodos probabilísticos, a teoria de grafos, etc. Recentemente surgem trabalhos relacionados para o problema em áreas correlatas ([R. Hohzaki 2006], [Soares 2010]. Porém, a nosso conhecimento não existe uma abordagem dentro da Programação Inteira para o problema prático. Portanto, apresentamos um modelo inovador que visa abranger o máximo de casos possíveis dentro do espectro apresentado.

Interessamo-nos particularmente por dois problemas a serem apresentados a seguir: o *Flamming Datum* e o Problema de Busca e Alocação de Recursos.

4.1.1 *Flamming Datum*

Seguramente o problema mais estudado dentro do campo da Teoria da Busca, o problema *Flamming Datum*, nasceu de uma situação militar. Uma vez que o alvo tenha revelado sua posição através de uma fonte externa ao problema, o pesquisador deve reencontrá-lo após um certo tempo decorrido à partir do instante inicial t_0.

Tradicionalmente o pesquisador utiliza um meio de pesquisa que consome uma certa quantidade de tempo a e é capaz de averiguar uma subárea de raio r. Também podemos atribuir-lhe uma velocidade de deslocamento v_P. Já a sua contraparte se locomove a uma velocidade v_S. A relação entre esses parâmetros de velocidade e capacidade de pesquisa, é responsável pela dificuldade do problema. Com efeito, é sabido que, se o tempo consumido por cada pesquisa é suficientemente pequeno e $v_P > v_S$, uma busca em espiral pode resolver o problema de forma ótima [Garnaev 2000].

4.1.1.1 Casos estudados na literatura

Este problema foi modelado pela primeira vez em [Meinardi 1964] sob forma de um *duelo*, ou seja, um jogo de soma nula. A função de pagamento nesse caso é a probabilidade de deteção do alvo, e o espaço e tempo são discretos. Além disso, o autor considera uma lei de deslocamento uniforme (o alvo se locomove de maneira equiprovável para qualquer um dos vizinhos da célula onde se encontra).

Uma variante do problema é tratada por Danskin [Danskin 1968], onde o alvo escolhe uma trajetória à priori e não muda ao longo de sua missão. O autor verifica que o ponto de equilíbrio deste jogo é encontrado de uma maneira muito simples: a estratégia ótima para o alvo consiste em manter uma velocidade constante e uma trajetória uniforme. Para o pesquisador, basta uma distribuição regular de seu *esforço de busca*. O *esforço de busca* é a distribuição da probabilidade com a qual o pesquisador deve efetuar uma busca em determinada subzona. Esses problemas podem ser considerados a base do estudo do *Flamming Datum*.

Ademais, certos autores, como Brown [Brown 1980] ou Washburn [Washburn 1980] propõem o caso em que o alvo pode se comportar de maneira não uniforme. Nesse caso, algoritmos estocásticos propostos por tais autores permitem convergir a uma solucção

ótima relativa ao *esforço de busca.*

Eagle [Eagle 1984] apresenta uma solução através de programação dinâmica no caso do alvo seguir uma lei Markoviana em seu movimento, ou seja, sua posição no instante t só depende de sua posição no instante $t-1$. Com efeito, para o caso de "um alvo markoviano"(considerando o tempo discreto), a solução do problema pode ser dada por sucessivas otimizações locais do problema para cada instante de tempo [Kan 1977]. Esse resultado garante a otimalidade dos métodos de [Brown 1980, Washburn 1983, Eagle 1984].

4.1.1.2 A inteligência do alvo

Todas as soluções apresentadas acima não levam em consideração, no entanto, a inteligência do alvo. Na teoria dos jogos, um modelo matemático proposto por [Thomas 1991] é a primeira aparição de tal problema, chamado na ocasião de "Dynamic Search Game"(**DSG**). A diferença em relação aos seus precedentes é que o alvo toma conhecimento das ações de busca tomadas pelo pesquisador. Ou seja, podemos dizer que o alvo é capaz de memorizar as posições pesquisadas pela sua contraparte e em que momento elas foram realizadas. Os autores também estendem ao pesquisador a possibilidade de pesquisar mais de uma posição ao mesmo tempo. Finalmente esse artigo mostra um modelo para uma solução local (para um instante de tempo fixo) e o autor sugere que um algoritmo de programação dinâmica poderia estender essa solução para uma solução global. Apresentamos a seguir o modelo concebido em [Thomas 1991].

4.1.1.3 Modelo para o DSG

Sejam dados um conjunto de células K e duas funções P e T tais que $P: K \times K \times Z_{H+1} \rightarrow R_{H+1}$ e $T: K \times K \times Z_{H+1} \rightarrow Z_{H+1}$, onde H é o horizonte de tempo do problema. O valor $T(i,j,t)$ representa o menor instante de tempo no qual o pesquisador é capaz de ir da célula i para a célula j. $P(i,j,t)$ é a probabilidade de que o alvo posicionado em j seja detectado pelo pesquisador efetuando a busca na célula i no instante t.

O jogo se desenvolve da seguinte forma:

- A cada instante de tempo, o alvo tem a informação sobre a última posição pesquisada. A partir daí ele escolhe uma nova posição para ocupar. Por sua vez, o pesquisador, sem conhecer a posição do alvo, mas sabendo que todas as suas buscas falharam até o presente escolhe uma nova célula para efetuar sua ação.

- Se o tempo limite H é atingido, o alvo ganha. Senão, existe uma probabilidade $P(i,j,T(k,i,t))$ que o alvo tenha sido detectado. Se ele não é detectado, o jogo recomeça com $k=i, t=T(k,i,t)$.

O jogo está, portanto, definido e avaliado para cada instante de tempo, limitado por um horizonte finito e para cada célula. Seja $Q(i,t)$ o valor do jogo. Denotamos por $x=[x_k]$ a estratégia do pesquisador e $y=[y_k]$ a estratégia do alvo. Trata-se, evidentemente, de dois vetores com as posições dos atores do problema. Essas são as variáveis do jogo.

O modelo proposto por Thomas e Washburn [Thomas 1991] para o pesquisador é o seguinte:

$$Q(i,t) = \min_{y} \max_{x} \sum_{j} y_j \left\{ 1 - \sum_{k} x_k P(j,k,T(i,j,t)) \right\} Q(j,T(i,j,t)) \tag{4.1}$$

onde

$$\sum_{j} y_j = \sum_{k} x_k = 1.$$

Distribuindo o termo $\{1 - \sum_k x_k P(j,k,T(i,j,t))\}$ chegamos a:

$$Q(i,t) = \min_{y} \max_{x} \sum_{j} \left(y_j Q(j,T(i,j,t)) - \sum_{k} x_k P(j,k,T(i,j,t)) Q(j,T(i,j,t)) \right) \tag{4.2}$$

$$Q(i,t) = \min_{y} \left\{ \sum_{j} y_j Q(j,T(i,j,t)) - \min_{x} \left\{ \sum_{k} x_k \sum_{j} y_j P(j,k,T(i,j,t)) Q(j,T(i,j,t)) \right\} \right\} \tag{4.3}$$

Os autores então introduzem uma variável z que corresponde ao mínimo, em k, dos termos $\sum_j y_j P(j,k,T(i,j,t)) Q(j,T(i,j,t)$. O modelo é reduzido a:

$$\begin{aligned} Q(i,t) = \quad & \min_{(y,z)} \quad \sum_{j} y_j Q(j,T(i,j,t)) - z \\ & s.t.: \quad \sum_{j} y_j P(j,k,T(i,j,t)) Q(j,T(i,j,t) - z \geq 0, \quad \forall k \in S \\ & \sum_{j} y_j = 1 \\ & y_j \geq 0, \forall j \end{aligned} \tag{4.4}$$

Além disso, é provado que o modelo, do ponto de vista do alvo (4.5), corresponde ao dual de (4.4), comprovando assim o duelo (jogo de soma nula).

$$\begin{aligned} Q(i,t) = \quad & \max_{(x,u)} \quad u \\ & s.t.: \quad Q(j,T(i,j,t) \left(1 - \sum_{k} x_k P(j,k,T(i,j,t)) \right) - u \geq 0, \quad \forall j \in S \\ & \sum_{k} x_k = 1 \\ & x_k \geq 0, \forall k \end{aligned} \tag{4.5}$$

A concepção desse modelo é a base para uma abordagem do problema de busca de um alvo inteligente através da Programação Matemática. No entanto, a resolução desse

modelo a cada instante de tempo não satisfaz a otimalidade global. É preciso um modelo que seja capaz de considerar o problema global.

Tais modelos foram concebidos por Hohzaki [Hohzaki 2006] e se baseiam na idéia de que as estratégias de cada um dos atores pode ser definida por um caminho no espaço de busca. Veremos um exemplo a seguir.

4.1.2 Busca e alocação de recursos

Originalmente chamado "Search Allocation Game"(**SAG**), este problema também consiste em um duelo onde o pesquisador deve determinar a alocação de seus recursos afim de encontrar um alvo móvel. Por sua vez, o alvo deve evitar ser encontrado. Este problema foi definido em [Garnaev 2000]. Este tipo de problema faz aumentar o número de variáveis de decisão. É evidente que, além de determinar sua própria trajetória, o pesquisador também deve determinar um plano de instalação de seus recursos. Além disso, podemos introduzir restrições relativas aos próprios recursos (duração, número limitado, restrições de comunicação, tempo de instalação, etc.) Em consequência, para o caso de um alvo inteligente, este é uma extensão do jogo de busca dinâmico (DSG).

A literatura sobre este problema é vasta e propõe variantes representando diversas situações que espelham casos reais. A adição de restrições de energia e restrições sobre os recursos estão entre as mais comuns variações do problema. Porém, uma dificuldade significativa para a resolução dos casos propostos consiste em conceber um modelo eficaz.

4.1.2.1 Um modelo para um jogo de busca e alocação de recursos

A modelagem desse problema consiste, primeiramente, em considerar como estratégia para cada um dos atores a escolha da melhor trajetória possível. O problema desta abordagem é que o número de caminhos é exponencial. Além disso, dentro da Teoria dos Jogos, a integralidade das variáveis é comumente ignorada, como é o caso do modelo que vamos apresentar a seguir, proposto por Hohzaki [Hohzaki 2006].

Chamaremos de $K = \{1, \cdots, n\}$ o espaço de busca. Seja $T = \{1, \cdots, H\}$ o espaço de tempo, onde H é o horizonte de tempo. O pesquisador deve então distribuir o esforço de busca entre as células afim de maximizar a probabilidade de encontrar o alvo. O alvo deve se locomover de maneira ótima afim de evitar o pesquisador, ou seja, minimizando a sua probabilidade de detecção. O espaço de estratégias (caminhos) é limitado por n^H e cada caminho p é definido como uma sequência $p_1, \cdots, p_m$ onde $p_i \in K$, $1 \leq i \leq m$.

Um plano de distribuição dos recursos φ é definido como $\varphi = \{\varphi(i,t),\ i \in K,\ t \in T\}$, onde $\varphi(i,t)$ representa um valor que indica a quantidade de esforço de pesquisa gasto na célula i no instante t. Este parâmetro permite ao modelo caracterizar diversas restrições relativas aos recursos (número limitado, zonas proibidas, etc). Fazemos ainda a hipótese que o domínio de φ é convexo.

A estratégia do pesquisador é então definida como a escolha de um par (φ, p) representando, respectivamente, um plano de distribuição do esforço de pesquisa e um caminho.

Atribuimos a cada par (φ, p) um valor $R(\varphi, p)$ que representa a recompensa pela escolha de (φ, p). Para o alvo esse valor representa o perigo de ser encontrado. Supomos R uma função côncava.

Posto que temos uma função de avaliação côncava e uma avaliação convexa para o esforço de busca, sabemos que este jogo possui um ponto de equilíbrio [Hohzaki 1999]. Seja π a estratégia escolhida pelo alvo. Temos então que:

$$\max_{\varphi \in \psi} \min_{\pi \in \prod} R(\varphi, \pi) = R(\varphi^\star, \pi^\star) = \min_{\pi \in \prod} \max_{\varphi \in \psi} R(\varphi, \pi).$$

onde $(\varphi^\star, \pi^\star)$ é a estratégia de equilíbrio para o jogo, $\prod$ é o conjunto de estratégias possíveis para o alvo e ψ é o conjunto de planos possíveis para o pesquisador.

Definimos Ω como o conjunto de caminhos possíveis para o pesquisador. O modelo para este jogo do ponto de vista do pesquisador é portanto:

$$\begin{aligned} \gamma^\star = \quad & \max_{\varphi \in \psi} \quad \gamma && (4.6) \\ & s.t.: \quad R(\varphi, p) \geq \gamma, \forall p \in \Omega && (4.7) \\ & \qquad \varphi \in \psi && (4.8) \end{aligned}$$

Resta ainda a definição da função de recompensa $R(\varphi, p)$. Ela deve representar o ganho em porcentagem pelo esforço de pesquisa alocado. Supondo que cada célula possui um atributo $0 \leq \alpha_i \leq 1, i \in S$ indicando a eficácia da pesquisa naquela célula e seguindo [Hohzaki 2006] temos:

$$R(\varphi, p) = 1 - exp\left(- \sum_{t \in T} \alpha_{p(t)} \varphi(p(t), t) \right)$$

onde $p(t)$ é a posição que se encontra o pesquisador no instante t.

Passando a função exponencial a seu logaritmo $\eta \equiv log\left(\frac{1}{1-\gamma}\right)$, obtemos o seguinte modelo:

$$\begin{aligned} \eta^\star = \quad & \max_{\varphi \in \psi} \quad \eta && (4.9) \\ & s.t.: \quad \sum_{t \in T} \alpha_{p(t)} \varphi(p(t), t) \geq \eta, \forall p \in \Omega && (4.10) \\ & \qquad \varphi(i, t) \geq 0, \forall i \in S, t \in T && (4.11) \end{aligned}$$

sabendo que o valor do jogo é $\gamma^\star = 1 - e^{-\eta^\star}$.

Ainda que linear esse modelo não satisfaz às nossas exigências. Primeiramente ele não é capaz de dar uma solução bem definida para o problema, e sim uma repartição do esforço de busca. Além disso, temos um problema ao tratar com um número exponencial de variáveis, ou seja, todos os caminhos possíveis para o pesquisador.

4.1.3 Definição do Problema de Busca de um Alvo Inteligente

Vamos categorizar o problema que tratamos nesse trabalho segundo os casos vistos na literatura. É importante frisar que o problema (ou a classe de problemas) tratado nesse trabalho compreende os casos clássicos do *Flamming Datum*, do problema de busca e alocação de recursos, do problema de busca dinâmico, entre outros. O modelo que propomos descreve todos eles, diferindo-os somente pela escolha de parâmetros usados na formulação. Situamo-nos na posição do pequisador, e uma solução para nós é caracterizada por um plano de busca, ou seja, uma sequência de ações de deslocamento, instalações e ativações de recursos. Um plano ótimo é um plano que maximiza a probabilidade de detecção do alvo.

Vamos modelar um problema com o alvo móvel e adversativo ao pesquisador, ou seja, o alvo tenta continuar oculto. O espaço do problema é definido por uma zona da qual podemos definir três sub-zonas: a zona autorizada ao pesquisador (zona de busca), a zona inicial do alvo e a zona final do alvo. Essas duas últimas caracterizam que o alvo possui uma missão a executar dentro do horizonte de tempo finito. Quanto a zona autorizada ao pesquisador, esta define o espaço de busca do problema.

A pesquisa é feita de maneira não monotônica através da utilização de recursos idênticos. Estes são caracterizados por diversos parâmetros: um tempo de instalação (período no qual o pesquisador deve ficar no local de instalação), um tempo de habilitação (período que segue a instalação e no qual o pesquisador não pode ativar o recurso), um duração de vida ativa, um raio de detecção, um raio de comunicação (dentro do qual o pesquisador pode ativar o recurso) e um número máximo de ativações. Além disso, um número limitado de recursos está disponível para o pesquisador. Para um bom entendimento do problema é necessária a compreensão do funcionamento dos recursos. Uma vez ativado o recurso fornece ao pesquisador uma visualização momentânea da área definida pelo seu raio de captura. Caso a probabilidade que o alvo esteja nessa área seja igual a 1, o problema foi resolvido à otimalidade e nenhuma outra pesquisa mais é necessária. Caso contrário, o pesquisador procede com o seu plano de busca. A Figura 4.2 mostra um resumo de algumas características aqui descritas.

A inteligência do alvo é caracterizada por sua capacidade de reconhecer as ativações dos recursos feitas pelo pesquisador. Essa capacidade é limitada por um raio de contradetecção. Para nós o alvo utiliza essa informação da seguinte maneira: uma vez contradetectado um recurso, o alvo assume que a região coberta por esse recurso é perigosa e não entrará nessa região durante a vida ativa estimada do recurso. Essa estimativa é feita baseada na mais recente ativação percebida pelo alvo. Caso o objetivo do alvo (definido pelas zonas inicial e terminal) exija uma passagem por uma zona estimada como perigosa, o alvo desviará de tal zona, podendo mesmo continuar parado pelo tempo necessário. Dessa forma, consideramos que o alvo prioriza a sua própria não detecção em detrimento de seu objetivo.

Com excessão da inteligência do alvo, todas as demais restrições do problema não representam um grande desafio de modelagem. Assim sendo, a concepção de um modelo para

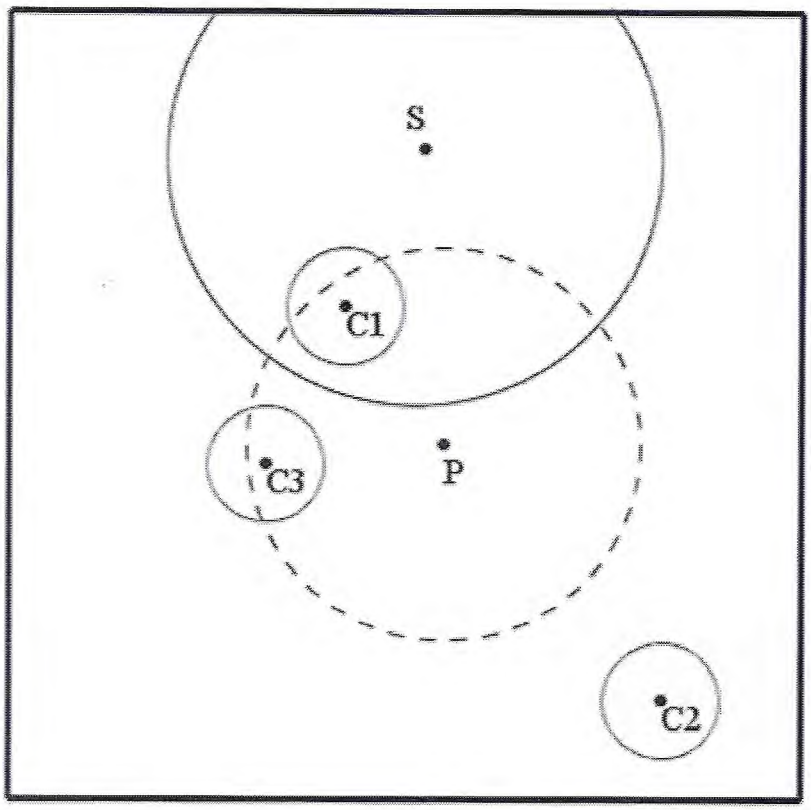

Figura 4.2: Ilustração do problema tratado. Vemos nessa figura um esquema representativo de um instante do problema. O pesquisador instalou 3 sensores representados pelos círculos centrados em C1, C2 e C3. O raio de comunicação do pesquisador, representado nessa figura por um círculo pontilhado permite que este se comunique com os recursos C1 e C3. Caso uma ativação seja efetuada, o alvo, aqui representado por S, poderá contradetectar o sensor C1, pois este encontra-se dentro do raio de contradetecção do alvo.

o problema de busca de um alvo inteligente se orienta na modelagem do comportamento do alvo.

4.2 Um Modelo para o Problema de Busca de um Alvo Inteligente

Detalhamos nessa seção as escolhas realizadas para a modelagem do problema apresentado. Primeiramente mostramos que a modelagem do comportamento reativo do alvo é feita através de um número finito de possibilidades determinísticas. Tão logo apresentada a solução, passamos ao modelo de programação linear 0-1 propriamente dito, descrito através de seus dados iniciais, variáveis de decisão, função objetivo e restrições. Em

seguida fazemos uma análise do modelo apresentado.

4.2.1 Concepção do modelo

Primeiramente, é importan notar partimos do problema de um problema real para a definição de um problema matemático. Essa passagem é feita, entre outros pontos, por uma discretização do espaço e do tempo definidos para o problema. Essa discretização é uma conversão dos parâmetros dados em medidas reais para medidas do modelo – células para as dimensões da zona e passos de tempo para o tempo. Logo, o primeiro passo para tratarmos o problema é a definição de um tamanho de célula e de passo de tempo (doravante chamado simplesmente passo).

O modelo apresentado é válido para quaisquer parâmetros de célula e passo, embora uma escolha cuidadosa se faça necessária devido ao número de variáveis e restrições que ela acarreta. Veremos na Seção 4.2.3 que o modelo possui uma quantidade linear no número de células e passos de variáveis e quadrático de restrições.

Para a resolução prática propomos que o tamanho da célula seja ligado ao raio de detecção associado aos sensores, ou seja, cada célula do problema é um quadrado cuja diagonal é igual ao diâmetro de detecção. Desta maneira, podemos garantir que quando um sensor é depositado em uma célula, toda a área representada pela célula é coberta pelo sensor. Uma ativação do sensor numa determinada célula k em um instante de tempo t permite, portanto, uma resposta à questão: o alvo se encontra na célula k em t? Além disso, tal definição de célula nos permite uma discretização da zona em um número mínimo de células que possam responder à questão precedente. Veremos na Seção 4.3 que alguns ajustes se fizeram necessários para a resolução dos problemas práticos.

A discretização permite-nos definir uma série de dados do problema, bem como algumas variáveis. Primeiramente definimos o conjunto de células do problema K e o horizonte de tempo T, dividido em passos de tempo $\{1, ..., T\}$. Em seguida, definimos a adjacência do pesquisador para uma célula k, $Adj_P(k)$, como o conjunto de células que podem ser alcançadas à partir de k em um instante de tempo. Similarmente, o conjunto de células alcançáveis à partir de uma célula k pelo alvo é chamado $Adj_S(k)$. O mesmo princípio é utilizado para definir as zonas de detecção para um sensor instalado em k, $CH(k)$, bem como o conjunto de células das quais o alvo pode contradetectar um sensor instalado em k, $CD(k)$. Ainda seguindo o raciocínio, o conjunto de células com as quais um pesquisador situado em k pode se comunicar é chamado $Port(k)$.

Para os recursos, consideramos os seguintes dados: um número limite de sensores disponíveis C, uma duração de vida para o sensor D, um número máximo de comunicações que o sensor pode fazer L, um tempo de instalação do recurso (durante o qual o pesquisador deve permanecer imóvel, representando a instalação) I, um tempo de habilitação do recurso a (este tempo representa uma certa quantidade de passos que o pesquisador, após realizada a instalação, deve passar antes de efetuar qualquer ativação do recurso).

Vamos definir agora as variáveis do problema, começando por aquelas que se referem

somente ao pesquisador. A posição do pesquisador é denotada por uma variável binária $x_{k,t}$ que indica que o pesquisador está na posição k no instante t caso igual a 1. Em seguida, definimos $d_{k,t}$ uma variável binária indicando se um sensor é depositado na célula k no instante t. Temos ainda $y_{k,t}$ que indica que um sensor previamente depositado em k é ativado no instante t.

Em vista dos dados apresentados e dessa definição de variáveis, é simples [Hohzaki 1999] construir um conjunto de inequações adequadas para modelar a cinemática, o raio de comunicação, as limitações dos sensores, etc. A maior dificuldade reside agora em modelar o alvo, seu comportamento reativo e as restrições de detecção e contradetecção.

Dentro de nosso contexto, o alvo possui uma missão, que pode variar conforme a instância do problema. Uma missão do alvo é, para o nosso modelo, definida por uma zona de partida, uma zona de chegada e uma zona viável, dentro da qual o alvo tem o direito de se locomover. O comportamento do alvo no modelo deve refletir esta missão, porém, consideramos que para o alvo é sempre mais importante que sua posição não seja revelada. Nesse sentido, a missão do alvo é calculada como um campo magnético que converge a um ponto de interesse dentro da zona de chegada, enquanto que a reatividade às ações do pesquisador é modelada através de restrições no modelo matemático. Em nosso caso, consideramos que uma vez que o alvo tenha contradetectado uma ativação de um sensor, ele considera a zona coberta pelo sensor como zona de risco por uma duração limitada (e igual à duração do recurso).

Para modelar o comportamento incerto e ao mesmo tempo reativo, inteligente do alvo, nós vamos construir um espaço amostral das possibilidades do alvo. A idéia é similar ao modelo de [Hohzaki 2000], mas conforme mostrado pelo autor, não é possível utilizar todo o espaço de caminhos para o alvo. Além disso, é necessário desconsiderar certas trajetórias que não são realistas, ou seja, trajetórias em que o alvo é levado à zona de risco.

Portanto vamos fazer uma geração aleatória prévia de trajetórias que serão atribuídas a alvos determinísticos fictícios, que possuem um comportamento pré-definido. Tal comportamento não impede, no entanto que o alvo possa, conforme o plano de busca calculado, mudar sua trajetória para evitar ser encontrado.

Definimos então, para cada alvo s no conjunto de alvos fictíficios M um conjunto de transições preferenciais que convergem para a missão do alvo:

$$\vec{t}^{s}_{k',k,t} = \begin{cases} 1, & \text{se a preferência do alvo } s \in M \text{ que está na} \\ & \text{célula } k \in S \text{ no instante } t \in [0,T] \text{ é de ir à célula } k' \\ 0, & \text{caso contrário} \end{cases}$$

Portanto a criação de um alvo envolve, inicialmente, a escolha de um ponto de chegada. Em seguida calculamos uma tabela de distâncias pseudo aleatórias entre as células, ou seja, a partir da distância euclidiana, adicionamos um peso aletório capaz de dar a cada alvo uma preferência entre duas células equidistantes. Através do algoritmo de Dijkstra, para caminhos mais curtos, encontramos todos os caminhos que levam até o ponto de destino aleatório escolhido.

É importante notar que esta maneira de criar os caminhos preferenciais do alvo não leva em consideração a estratégia que será desenvolvida pelo pesquisador. No entanto, o fato de estar sendo procurado pode ser levado em consideração. Por isso escolhemos caminhos curtos para o alvo, pois este tem o interesse, na realidade, de executar sua missão o mais rápido possível, de maneira segura. Além disso, seguindo o resultado de [Gal 1979], a estratégia ótima para o alvo que não possui informação sobre o pesquisador estabelece que o destino do alvo deve mudar aleatoriamente após um período de tempo também aleatório. Logo, implementamos na construção de nossas transições, um intervalo aleatório de mudança do ponto de destino e redefinição do campo magnético. A Figura 4.3 mostra como podem ser percebidas as transições.

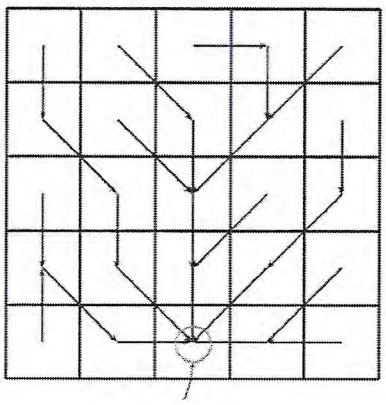

Figura 4.3: Transições são um campo magnético que convergem para um ponto de interesse.

Esse dado é passado para o modelo que deve assegurar que o alvo sempre fará sua transição preferencial a menos que a transição o leve para uma zona de risco. Nesse caso, uma nova posição que não pode ser uma zona de risco para o alvo é determinada como sua próxima posição, caso seja possível. Se não for possível, o alvo permanece imóvel (o que não é uma posição arriscada, a menos que o alvo tenha sido detectado). A figura 4.4 mostra exemplos de desvios.

Podemos ainda, à partir desse espaço amostral de alvos, estimar a probabilidade de encontrarmos o alvo com um dado plano de pesquisa. Essa estimativa pode ser feita com a razão entre o número de alvos detectados pelo plano e o número de alvos do espaço amostral. Assim, definimos a função objetivo do nosso modelo. Seja v^s uma variável binária que indica se o alvo foi detectado ao longo da missão. A função objetivo do nosso modelo é:

$$Max \sum_{s \in M} v^s.$$

Definimos finalmente as variáveis relativas ao alvo: posição, detecção e contra-detecção. Chamamos de $w^s_{k,t}$ a variável binária de posição do alvo que indica se o alvo

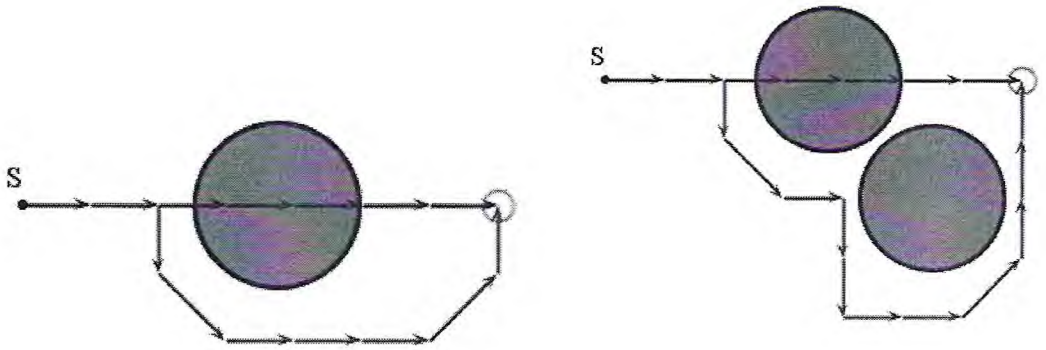

Figura 4.4: Exemplos de desvios.

s está na posição k no instante t. A variável $z^s_{k,t}$ indica se o alvo s é detectado por um sensor instalado em k no instante t. Definimos de maneira análoga a variável $u^s_{k,t}$ para a contra-detecção.

Apresentamos a seguir um recapitulativo de todas as variáveis e dados do modelo. Em seguida descrevemos a função objetivo e todas as restrições que o compõem.

4.2.2 Descrição do modelo

De forma concisa, vamos apresentar o modelo em três partes: dados, variáveis de decisão e restrições.

1. **Dados** :

   (a) Conjunto de células K

   (b) Conjundo de instantes de tempo: $[0, T]$, onde T é o horizonte de tempo.

   (c) Conjunto de alvos determinísticos M.

   (d) Transições (ou estratégias) de cada alvo para cada instante de tempo:

$$\vec{t}^{\,s}_{k',k,t} = \begin{cases} 1, & \text{se a preferência do alvo } s \in M \text{ que está na} \\ & \text{célula } k \in S \text{ no instante } t \in [0,T] \text{ é de ir à célula } k' \\ 0, & \text{caso contrário} \end{cases}$$

   (e) Número de recursos disponíveis C

   (f) Duração D de um recurso.

   (g) Número máximo de comunicações L que cada recurso pode realizar.

   (h) Tempo de instalação I de um recurso.

   (i) Tempo de habilitação de um recurso a.

(j) $Adj_P(k)$: conjunto de células adjacentes para o pesquisador à partir da célula $k \in K$.

(k) $Adj_S(k)$: conjunto de células adjacentes para o alvo à partir da célula $k \in K$.

(l) $Port(k)$: conjunto de células acessíveis a comunicação entre um recurso e o pesquisador posicionado na célula $k \in K$.

(m) $CH(k)$: conjunto de células acessíveis à pesquisa por um recurso posicionado na célula $k \in K$.

(n) $CD(k)$: conjunto de células que podem contradetectar um recurso posicionado na célula $k \in K$.

2. **Variáveis de decisão** :

(a) Posição do pesquisador

$$x_{kt} = \begin{cases} 1 & \text{se o pesquisador está em } k \in K \text{ no instante } t \in [0,T], \\ 0 & \text{caso contrátio.} \end{cases}$$

(b) Instalação de um recurso

$$d_{kt} = \begin{cases} 1 & \text{se um recurso é depositado em } k \in K \text{ no instante } t \in [0,T], \\ 0 & \text{caso contrátio.} \end{cases}$$

(c) Ativação de um recurso

$$y_{kt} = \begin{cases} 1 & \text{se o recurso em } k \in K \text{ é ativado no instante } t \in [0,T], \\ 0 & \text{caso contrátio.} \end{cases}$$

(d) Posição de um alvo

$$w^s_{kt} = \begin{cases} 1, & \text{se o alvo } s \in M \text{ está na posição } k \in K \text{ no instante } t \in [0,T] \\ 0, & \text{caso contrátio.} \end{cases}$$

(e) Detecção do alvo

$$z^s_{kt} = \begin{cases} 1, & \text{se o alvo } s \text{ é detectado por um sensor posicionado na célula} \\ & k \in K \text{ no instante } t \in [0,T] \\ 0, & \text{caso contrátio.} \end{cases}$$

(f) Contradetecção do alvo

$$u^s_{kt} = \begin{cases} 1, & \text{se um recurso é contradetectado na célula } k \in K \text{ no instante} \\ & t \in [0,T] \text{ pelo alvo } s \\ 0, & \text{caso contrátio.} \end{cases}$$

(g) Valor de um alvo s.

$$v^s = \begin{cases} 1, & \text{se o alvo } s \text{ é detectado.} \\ 0, & \text{caso contrátio.} \end{cases}$$

3. **Função objetivo** :

$$Max \sum_{s \in M} v^s \tag{4.12}$$

4. **Restrições** :

(a) Restrições do pesquisador.

i. Unicidade da posição.

$$\sum_{k \in K} x_{kt} = 1, \quad \forall t \in [0, T] \tag{4.13}$$

O pesquisador ocupa uma única célula a cada instante de tempo.

ii. Adjacência (ou cinemática).

$$x_{k,t+1} \leq \sum_{k' : k \in Adj_P(k')} x_{k't}, \quad \forall k \in K, t \in [0, T-1] \tag{4.14}$$

Em uma unidade de tempo o pesquisador se desloca dentro de sua adjacência.

(b) Restrições sobre os recursos.

i. Limite do número total de recursos.

$$\sum_{k \in K} \sum_{t \in [0,T]} d_{kt} \leq C \tag{4.15}$$

Um limite pré-estabelecido de número de sensores é imposto. Para nosso modelo isso é representado por um limite no número de instalações que o pesquisador pode efetuar.

ii. Limite do número de ativações.

$$\sum_{t \in [t', t'+D]} y_{kt} \leq L, \quad \forall k \in K, t' \in [0, T] \tag{4.16}$$

Um recurso é limitado na quantidade de comunicações que ele pode efetuar com o pesquisador. Esse limite é representado no modelo por uma restrição do número de ativações dentro do tempo de vida do sensor.

iii. Ativação deve ser realizada após a instalação e durante a duração do recurso.

$$y_{kt'} \leq \sum_{t \in [t'-D-a, t'-a]} d_{kt} \quad \forall k \in K, t' \in [0, T-1] \tag{4.17}$$

(c) Restrições pesquisador-recursos.

i. Instalação deve ser realizada no local.

$$x_{kt} \geq d_{kt'}, \quad \forall k \in K, t' \in [0,T], t \in [t', t'+I] \tag{4.18}$$

Essa restrição garante que o pesquisador fica no local da instalação por um período de tempo definido pela tempo de instalação do sensor.

ii. Restrição de comunicação.

$$y_{kt} \leq \sum_{k' \in Port(k)} x_{k't}, \quad \forall k \in K, t \in [0,T] \tag{4.19}$$

O pesquisador só pode se comunicar com o sensor se este estiver dentro da adjacência definida pelos conjuntos $Port(k)$ para cada célula k.

(d) Restrições do alvo.

i. Unicidade da posição.

$$\sum_{k \in K} w^s_{kt} = 1, \quad \forall t \in [0,T], \ s \in M. \tag{4.20}$$

Análoga à restrição apresentada para o pesquisador, essa restrição garante que cada alvo ocupa uma única posição.

ii. Adjacência (ou cinemática).

$$w^s_{k,t+1} \leq \sum_{k': k \in Adj_S(k')} w^s_{k't}, \quad \forall k \in K, t \in [0, T-1] \tag{4.21}$$

Idem à restrição apresentada para o pesquisador.

iii. Transições. Cada alvo obedece suas transições preferenciais, salvo contradetecção de um recurso.

$$w^s_{k,t+1} \geq \sum_{k': k \in Adj_S(k')} w^s_{k't} \cdot \vec{t}^s_{k',k,t} - \sum_{k'': k \in CH(k')} \sum_{t' \in [t-D,t]} u^s_{k''t'}, \forall k \in K, t \in [0, T-1] \tag{4.22}$$

Essa restrição tem por objetivo assegurar que cada alvo obedeça as transições pré-definidas $\vec{t}^s$ a menos que uma contra-detecção tenha sido efetuada em uma célula capaz de detectar a célula de destino da transição.

iv. Precaução do alvo.

$$w^s_{k,t+1} \leq 1 - \sum_{k'' \in CH(k')} u^s_{k''t'}, \quad \forall s \in M, k \in K, t' \in [0,T] k' \in CD(k), t \in [t', t'+D \tag{4.23}$$

Essa restrição proíbe o alvo de acessar uma célula que possui um sensor que tenha sido contra-detectado.

5. Restrições alvo-recursos.

   (a) Sistema de detecção. As restrições abaixo garantem, junto com (4.12), a detecção de um alvo localizado dentro do raio de cobertura de um recurso localizado em k no instante t.

$$z^s_{k't} \leq y_{k't}, \qquad \forall s \in M, t \in [0,T], k' \in K \tag{4.24}$$

$$z^s_{k't} \leq \sum_{k:k' \in CH(k)} w^s_{kt} \qquad \forall s \in M, t \in [0,T], k' \in K \tag{4.25}$$

   (b) Sistema de contradetecção. Similar ao sistema de detecção, porém, neste caso adicionamos uma restrição adicional para assegurarmos que toda contradetecção sera levada em conta. Contrariamente ao sistema de detecção, essa restrição é necessária pois as variáveis u não aparecem na função objetivo.

$$u^s_{k't} \geq \sum_{k:k' \in CD(k)} w^s_{kt} + y_{k't} - 1, \qquad \forall s \in M, t \in [0,T], k' \in K \tag{4.26}$$

$$u^s_{k't} \leq y_{k't}, \qquad \forall s \in M, t \in [0,T], k' \in K \tag{4.27}$$

$$u^s_{k't} \leq \sum_{k:k' \in CD(k)} w^s_{kt} \qquad \forall s \in M, t \in [0,T], k' \in K \tag{4.28}$$

   Os sistemas de detecção e contra-deteção tem uma característica quadrática. Na realidade ambas as restrições descritas acima representam a conjunção entre a posição do alvo em uma célula k e uma ativação realizada dentro do raio de detecção (ou, analogamente, contra-detecção). Podemos dizer que utilizamos a linearização clássica para a modelagem dessas restrições.

   (c) Restrição de contagem dos alvos

      i. Cada alvo conta um valor unitário na função objetivo.

$$v^s \leq \sum_{k \in K} \sum_{t \in [0,T]} z^s_{k,t}, \forall s \in M. \tag{4.29}$$

      Essa restrição limita a variável v aos alvos detectados.

4.2.3 Análise do modelo

4.2.3.1 Características do Modelo

Estabelecemos nesta seção algumas características apresentadas pelo modelo. Primeiramente esse modelo caracteriza-se por apresentar somente variáveis binárias e restrições lineares (ou linearizadas no caso das restrições (4.24)-(4.25) e (4.26)-(4.28)). Enquadra-se portanto na classe de Problemas Binários, o que nos dá acesso uma vasta gama de ferramentas matemáticas para sua resolução.

Podemos calcular, utilizando a notação O, o número de variáveis e restrições presentes no modelo. O número de variáveis pode ser calculado diretamente pela expressão $3 \cdot (|K| \cdot (T+1) \cdot (|M|+1)) + |M|$, ou seja $O(|K| \times T \times |M|)$ variáveis. O número de restrições é mais difícil de estimar, pois este depende das características do modelo. Porém podemos garantir que esse número é $O(|K|^2 \times T^2 \times |M|)$. Estabelecemos então que o modelo apresentado é constituído de um número pseudo-polinomial de variáveis e restrições no tempo e no espaço.

Um outro fator que aparece no tamanho do modelo é o número de alvos determinísticos que geramos. Cada alvo constitui um bloco independente da matriz, exceto por uma restrição (4.29). Este fato permite que técnicas de decomposição da matriz baseadas na geração incremental de alvos sejam consideradas.

Determinar o número de alvos necessários para estabelecermos uma solução confiável é portanto uma etapa crucial na resolução do problema.

4.2.3.2 Confiança estatística do modelo

O modelo apresentado varia em função do número de alvos determinísticos usados. Para determinar a eficiência e avaliar o tamanho do modelo precisamos calcular quantos alvos são necessários para obter um modelo estatisticamente confiável. Em nosso caso isto significa que o valor da função objetivo, a probabilidade de detectar o alvo, é representativo a partir de um número suficientemente grande de alvos.

Para determinar esse número vamos considerar algumas premissas. Evidentemente, em um caso real, essas premissas podem não ser obedecidas, mas as utilizamos para efeitos de simplificação e realizabilidade dos cálculos. Seria muito difícil estabelecer um resultado a respeito da confiança estatística do modelo se considerarmos, por exemplo, que o plano de busca do pesquisador afetas as suas escolhas, ou que os alvos possuem, cada um, uma missão diferente.

Primeiramente, supomos que os $|M|$ alvos (que representam os indivíduos de nosso modelo estatístico são todos independentes. Vamos assumir também que para um instante de tempo fixo, cada alvo tem a possibilidade de escolher qualquer uma das $|K|$ células do modelo como sua posição. Cada célula possui à ela associada uma probabilidade $p(k)$ de ser escolhida como posição do alvo.

Seja $N(k)$ o número de alvos que escolhem a célula k como posição. Sabemos que $(N(1), \cdots, N(|K|))$ obedece uma distribuição multinomial de $|K|$ possibilidades $\mathcal{M}(|M|, p(1), \cdots, p(|K|))$. A distribuição multinomial nada mais é que uma generalização da lei binomial para um número arbitrário de escolhas. Em tal distribuição, sabe-se que a densidade (discreta) é dada pela expressão :

$$\mathcal{P}(N(1) = n_1, \cdots, N(|K|) = n_{|K|}) = \frac{|M|!}{n_1! \times \cdots n_p!} p_1^{n_1} \times \cdots \times p_{|K|}^{n_{|K|}} \tag{4.30}$$

para todo $\{n_j \ : \ n_j \in \{0, \cdots, |M|\}$.

Para um caso equiprovável de escolha da célula, a expressão (4.30) pode ser simplificada para:

$$\mathcal{P}(N(1) = n_1, \cdots, N(|K|) = n_{|K|}) = \frac{|M|!}{n_1! \times \cdots n_p!} \left(\frac{1}{|K|}\right)^{|M|}. \tag{4.31}$$

Um resultado conhecido em estatística conhecido como Teorema Central do Limite aplicado a vetores multinomiais mostra que se o número de alvos $|M|$ é suficientemente grande, então a repartição de $(N(1), \cdots, N(|K|)$ segue uma distribuição normal.

Existe uma controvérsia estatística em relação do número suficientemente grande para termos uma distribuição normal. Alguns autores [Krumbein 1965, Dixon 1957] dizem que este número é $|M| \geq |K|^2/(|K|-1)$ enquanto outros falam em $|M| \geq 5|K|$. Em qualquer um dos casos, temos um limite linear para podermos aplicar o Teorema Central do Limite.

Logo que temos (no caso mais restrito) $|M| \geq |K|^2/(|K|-1)$, existe um resultado que relaciona o desvio da densidade de probabilidade para cada componente $j = 1, \cdots, |K|$ em relação à probabilidade estimada ($1/|K|$ no caso equiprovável), o número de células $|K|$ e o tamanho do espaço amostral $|M|$:

$$|M| \cdot |K| \cdot \sum_{j=1}^{|K|} \left(\frac{N(j)}{|M|} - \frac{1}{K}\right)^2 \overset{|M| \to \infty}{\longrightarrow} \chi^2(|K|-1), \tag{4.32}$$

onde χ^2 é a distribuição da soma dos quadrados de $|K|$ variáveis independentes. O teste do χ^2 é amplamente utilizado em estatística para a construção de intervalos de confiança, o que é exatamente o nosso caso. Com efeito, a literatura mostra que a confiança estatística do modelo é aproximadamente:

$$\alpha \approx 1 - \mathcal{P}\left(|M| \cdot |K| \cdot \sum_{j=1}^{|K|} \left(\frac{N(j)}{|M|} - \frac{1}{K}\right)^2 \leq \chi^2_{1-\alpha}(|K|-1)\right) \tag{4.33}$$

onde $\chi^2_{1-\alpha}(|K|-1)$ é o quantificador α na tabela da distribuição χ^2 com $|K|-1$ graus de liberdade [Krumbein 1965].

O desvio em relação ao valor esperado por cada componente é o que chamamos de margem de erro daquela componente ϵ_j. O erro total ϵ é portanto a soma dos erros para cada componente:

$$\epsilon = \sum_{j=1}^{|K|} \left(\frac{N(j)}{|M|} - \frac{1}{K}\right). \tag{4.34}$$

Podemos considerar que o erro total está uniformemente distribuido, portanto:

$$\epsilon_j = \frac{\epsilon}{|K|} \tag{4.35}$$

Logo, a partir das expressões (4.34) e (4.35), podemos relacionar a margem de erro, o tamanho do espaço amostral e o índice de confiança estatístico do modelo (4.33) :

$$\alpha \approx 1 - \mathcal{P}\left(|M| \cdot |K| \cdot \sum_{j=1}^{|K|} (\epsilon_j)^2 \leq \chi^2_{1-\alpha}(|K|-1)\right) \tag{4.36}$$

$$\approx 1 - \mathcal{P}\left(|M| \cdot |K| \cdot \sum_{j=1}^{|K|} \left(\frac{\epsilon}{|K|}\right)^2 \leq \chi^2_{1-\alpha}(|K|-1)\right) \tag{4.37}$$

$$\approx 1 - \mathcal{P}\left(|M| \cdot |K| \cdot \left(\frac{\epsilon^2}{|K|}\right) \leq \chi^2_{1-\alpha}(|K|-1)\right) \tag{4.38}$$

$$\approx 1 - \mathcal{P}\left(|M| \cdot \epsilon^2 \leq \chi^2_{1-\alpha}(|K|-1)\right) \tag{4.39}$$

$$\approx 1 - \mathcal{P}\left(|M| \leq \frac{\chi^2_{1-\alpha}(|K|-1)}{\epsilon^2}\right) \tag{4.40}$$

$$\approx \mathcal{P}\left(|M| \geq \frac{\chi^2_{1-\alpha}(|K|-1)}{\epsilon^2}\right) \tag{4.41}$$

Dessa forma, para uma dada margem de erro e para um índice de confiança pré-definido, podemos estimar o número de alvos necessários, através de uma consulta à tabela da distribuição χ^2, para a obtenção desses parâmetros. Observe que este método é similar aos métodos estatísticos utilizados em questionários de múltipla escolha para a obtenção do número de entrevistados necessários para validar uma pequisa.

Pudemos observar que mesmo para pequenos valores de tamanho do problema em termos de número de células, os valores para a obtenção de um modelo matemático confiável com uma pequena margem de erro são bastante elevados, fazendo com que o tamanho do modelo seja impraticável para uma abordagem direta. Devemos portanto passar por técnicas de decomposição para a resolução do modelo.

4.2.3.3 Decomposição do Modelo: Janelas Deslizantes

Em vista do tamanho do problema e de resultados experimentais preliminares, verificamos que a resolução do modelo é impraticável sem a utilização de uma técnica de decomposição da matriz. Uma primeira abordagem que investigamos, devido à natureza sequencial do problema, foi o método das Janelas Deslizantes, segundo o tempo.

Neste método vamos decompor todo o problema em subpartes de tamanho razoável a serem tratadas por uma resolução exata através de um método do tipo *branch-and-bound*. Cada subparte é definida por um intervalo de passos de tempo $[t, t+f]$ chamadas janelas, onde f é um inteiro positivo que caracteriza o tamanho da janela. Outro parâmetro do algoritmo é a quantidade de passos de tempo p fixados a cada janela.

Cada janela é um submodelo do modelo principal, onde apenas as variáveis e restrições relativas aos passos de tempo inseridos dentro do intervalo que define a janela são considerados. Evidentemente algumas restrições de acoplamento entre janelas se fazem necessárias para garantir uma continuidade da solução. Pois neste método iterativo vamos resolver

as janelas em sequência, aproveitando uma parte da solução das janelas anteriores como base para a próxima janela.

Iniciamos então a resolução do problema a partir da janela $[0, f]$. Se $f < T$, passamos a janela $[p, p+f]$, fixando na nossa solução todas as variáveis relativas ao intervalo $[0, p-1]$. De maneira mais geral, na iteração $k+1$ resolvemos a janela $[k \cdot p, k \cdot p + f]$, estando fixada toda a solução relativa à $[0, k \cdot p - 1]$. Prosseguimos iterativamente até chegarmos à iteração em que $k \cdot p + f > T$, onde o intervalo é reduzido para $[k \cdot p, T]$. Essa técnica é uma generalização do método de Brown [Brown 1980], conhecido na literatura por dar bons resultados para o problema.

No final da execução do método de janelas deslizantes temos um resultado heurístico para o problema global, calculado através da resolução de subpartes exatas. A qualidade da solução pode variar conforme a parametrização do modelo, sendo possível encontrar casos onde, segundo a parametrização, as Janelas Deslizantes possam dar um resultado muito ruim.

Utilizamos em nossos experimentos o passo do algoritmo p sempre igual a 1, o que nos conforta no sentido de procurarmos uma solução mais detalhada. Podemos dizer que a qualidade da solução também é proporcional ao tamanho da janela. É preciso notar que o número de variáveis em cada janela cresce de maneira linear e o número de restrições de maneira quadrática. Não é claro para nós, no entanto, como varia o tempo de resolução segundo o tamanho do modelo e tais dados são relatados na Seção 4.3.

4.2.4 Validação do modelo através de um simulador

Em nosso trabalho de revisão literária, podemos observar que a clsse de problemas tratados carece de resultados experimentais práticos. Por conta desse fator, uma solução apresentada para o problema é dificilmente comparável a outras na literatura. Para cetirficarmos as soluções obtidas através da resolução do modelo apresentado devemos utilizar um número muito grande, para um problema de programação inteira, de alvos.

Como exemplo, podemos tomar o caso de um modelo com 100 células e 30 instantes de tempo. Para aplicarmos os resultados apresentados na Seção 4.2.3.2, devemos estabelecer uma margem de erro e uma confiança desejadas. Para efeitos de exemplo, vamos supor que seja desejada uma confiança de 95% e uma margem de erro de 10% no total. Nesse caso, através de uma consulta na tabela do $\chi^2(99)$ obtemos um resultado que o número de alvos $|M|$ deve ser superior à 12000. Essa quantidade de alvos leva a criação de um problema de base que contém $3 \times (|K| \times T \times (|M|+1)) + |M| = 10812900$ variáveis, ou 372030 variáveis por instante de tempos, no caso de um método de Janelas Deslizantes. O número de restrições varia conforme os parâmetros da instância, mas para as instâncias práticas as quais fomos confrontados, esse número chega a 3 ou 4 vezes o número de restrições, o que na prática torna completamente inviável a utilização do número teórico.

No entanto, gostaríamos de investigar a performance do modelo mesmo com um número pequeno de alvos, e o quão distante este modelo estaria de uma solução com um número estatisticamente confiável de alvos. Devido à impossibilidade prática da uti-

lização de uma quantidade que permite afirmarmos um piso de confiança e uma margem de erro, desenvolvemos um simulador para o alvo.

Esse simulador tem o objetivo de determinar como um alvo aleatório, que reproduz as condições do alvo real, semelhante àquele apresentado na Secção 4.2.1, se comporta face a um plano de pesquisa previamente calculado. A idéia que temos é de gerar uma quantidade suficientemente grande de alvos, conforme calculado na Seção 4.2.3.2, para obter um resultado estatisticamente válido, segundo uma margem de erro e intervalo de confiança fixos.

4.3 Experimentos Computacionais

Descrevemos nesta seção os resultados computacionais realizados para o problema. Primeiramente vamos descrever as instâncias obtidas através da parceria com o Ministério da Defesa francês, através do órgão *"Direction Générale de l'Armée - Techniques Navales"* (DGATn). Em seguida vamos descrever quais testes foram realizados e qual o resultado obtido para cada um deles.

4.3.1 Definição das instâncias

As instâncias fornecidas pela *"Direction Générale de l'Armée - Techniques Navales"* correspondem a casos reais com os quais se deparam os militares franceses. Por motivo de confidência e à pedido da DGATn, não fazemos referências aos aspectos práticos de cada instância, mas descrevemos a parametrização de cada um dos atores do problema, bem como as características do alvo.

Tratamos de nove casos diferentes que podem ser subcategorizados em três tipos de missão para o alvo :

Missão 1 Reconhecimento: o alvo permanece na zona de pesquisa e faz deslocamentos aleatórios, sem revelar sua posição.

Missão 2 *Flaming Datum*, conforme descrito na Seção 4.1.1.

Missão 3 Barreira : o alvo deve atravessar uma zona na qual ele pode ser detectado. Nesse caso a zona de pesquisa é uma subzona da área total do problema. O alvo dispõe de um tempo limitado para atravessá-la.

São definidos igualmente três tipos de pesquisador :

Tipo 1 Lento. Seus recursos tem uma duração de vida instantânea porém possuem um grande raio de detecção.

Tipo 2 Rápido. Os recursos do pesquisador são duráveis e ele é capaz de se movimentar rapidamente em comparação com o alvo. Está limitado no entanto a um raio de comunicação nulo, ou seja, deve estar sobre o seu sensor para poder ativá-lo.

Tipo 3 Multi-sensor. Trata-se de um pesquisador capaz de instalar diversos sensores sobre a zona de pesquisa para ativá-los à distância, possivelmente vários ao mesmo tempo. O raio de pesquisa é, no entanto, pequeno, se comparado aos demais.

Tratamos três instâncias para cada caso, que chamamos A, B e C. A Tabela 4.3.1 mostra, para cada tipo de missão, o pesquisador que lhe foi atribuído em cada caso.

Tabela 4.1: Portador atribuído à cada instância fornecida.

	Missão 1	Missão 2	Missão 3
A	Tipo 1	Tipo 2	Tipo 1
B	Tipo 2	Tipo 2	Tipo 2
C	Tipo 3	Tipo 3	Tipo 3

Podemos a partir desses diferentes casos observar qual o comportamento do modelo quando variamos dois parâmetros: o tamanho da janela deslizante e o tamanho do espaço amostral. Todos os dados sobre cada uma das instâncias, incluindo os dados sobre a discretização são apresentados na Seção 4.3.2

4.3.2 Resultados

Condições dos experimentos

Todos os testes apresentados nesta seção foram realizados em um computador multi-processador (com 8 núcleos) Pentium IV 3.1GHz com 16 GB de memória RAM. O código foi realizado em linguagem C++ compilado com g++ 4.2. Pudemos observar que as instâncias geradas chegaram a consumir um total de até 6GB de memória RAM no caso mais extremo, e que ainda se trata de um modelo decomposto e reduzido com um número de alvos bem inferior ao limite teórico. Isso mostra que o modelo pode chegar a um tamanho muito grande mesmo em casos simplificados. Cada janela foi limitada a um tempo de resolução máximo de 2 horas.

Para efeitos de comparação de eficiência e confiança do modelo, fizemos variar para cada instância fornecida pela DGATn duas características do problema: o tamanho da Janela Deslizante e o tamanho do espaço amostral de alvos.

Para o tamanho da Janela Deslizante, primeiramente estabelecemos um piso fixo que é igual a duas vezes a soma do tempo de instalação mais o tempo de habilitação de um sensor. Em seguida adicionamos um valor suplementar de 1 ou de 3 instantes de tempo. Para o tamanho amostral fizemos testes para 50, 100, 300 alvos.

As características das instâncias, após a passagem pela etapa de discretização do problema, estão descritas nas Tabelas 4.2 a 4.4. Podemos verificar para cada uma delas as definições globais do problema: número de células $|K|$ e quantidade de passos de tempo T. Ainda é possível observar as características relativas aos sensores (mais detalhes na Seção

4.2.1) : duração de vida, tempo de instalação, tempo de habilitação, raio de comunicação e quantidade total de captores. Para o alvo fornecemos também o seu raio de contra-detecção.

As velocidades dos atores são denotadas na Tabela 4.2 como um raio do número de células adjacentes acessível em um instante de tempo. O conceito de raio também se aplica à detecção, comunicação e contra-detecção encontrados nas Tabelas 4.3 e 4.4.

Missão	Caso	$\|K\|$	T	Velocidades	
				Pesquisador	Alvo
1	A	625	12	1	3
1	B	9	33	1	≤ 1
1	C	81	72	2	$<< 1$
2	A	100	16	1	≤ 1
2	B	169	25	1	≤ 1
2	C	441	36	2	$<< 1$
3	A	660	12	1	3
3	B	24	33	1	≤ 1
3	C	150	72	2	$<< 1$

Tabela 4.2: Características das instâncias.

Missão	Caso	Detecção	Contra-detecção
1	A	4	5
1	B	1	2
1	C	1	2
2	A	1	2
2	B	1	2
2	C	1	2
3	A	4	5
3	B	1	2
3	C	1	2

Tabela 4.3: Características dos sensores.

Tabelas de resultados

Apresentamos primeiramente os resultados relativos à qualidade da solução calculada para cada caso considerado, segundo o tamanho da janela deslizante e o número de alvos. Estes resultados aparecem nas tabelas 4.5-4.13 e apresentam, para cada combinação de parâmetros duas características: o valor da probabilidade avaliada por nosso simulador, na coluna "Avaliação"e o tempo computacional expresso em segundos. Em negrito destacamos a melhor execução para cada um dos casos.

Missão	Caso	Duracção	Instalação	Habilitação	Comunicação	Quantidade
1	*A*	0	0	0	0	200
1	*B*	8	3	0	0	200
1	*C*	12	0	2	1	50
2	*A*	4	3	0	0	200
2	*B*	4	3	0	0	200
2	*C*	12	0	2	1	50
3	*A*	0	0	0	0	200
3	*B*	8	3	0	0	200
3	*C*	12	0	2	1	50

Tabela 4.4: Características dos sensores.

O caso A da missão 1 é, para o nosso método, um dos casos mais difíceis de ser resolvido. Os resultados experimentais da tabela 4.5 não são conclusivos sobre os parâmetros, pois apenas uma janela de tempo mínima foi realizada.

Para as combinações de missão e caso 1B, 1C, 2A, 2B e 2C, apresentados nas tabelas 4.6-4.10, o nosso método foi capaz de encontrar soluções para qualquer um dos parâmetros propostos (exceto tamanho de janela igual a 3 e tamanho da amostra igual a 300 para a missão 1, caso C). Em geral, vemos a tendência de uma melhora no valor da solução segundo o aumento no valor dos parâmetros (tamanho amostral ou tamanho da janela).

Além disso, podemos concluir que o aumento no tamanho da janela acarreta num aumento drástico do tempo computacional. Este fato é facilmente explicado ao examinarmos o aumento no número de variáveis e restrições. Além disso, o problema torna-se em si muito mais difícil devido a sua característica exponencial.

Para o caso A da missão 3 (ver Tabela 4.11), vemos precisamente o efeito do aumento do tamanho da janela tornando inviável, por conta do limite de duas horas para cada janela, a resolução do problema. Nesse caso, notamos, porém, que o aumento do tamanho amostral permite um aumento sensível no valor da probabilidade avaliada.

A missão 3, caso B, mostrada na Tabela 4.12, tem o comportamento típico esperado dentro dos experimentos, ao referenciarmos as Tabelas 4.6-4.13, ou seja, em geral o valor da solução aumenta segundo o aumento do tamanho da janela e a quantidade de alvos determinísticos no modelo.

Por fim, o caso C da missão 3 também apresentou-se bastante difícil para a resolução via o método aqui apresentado. Podemos concluir que os casos difíceis apresentam duas características principais: um grande número de possibilidades para o pesquisador (1C e 3C) ou um raio de detecção superior a 1, pós-discretização. Com efeito, pudemos observar em nossos experimentos preliminares que o raio de pesquisa superior a 1 enfraquece a relaxação linear do modelo, ou seja, a relaxação linear se torna mais distante do valor ótimo para qualquer janela do problema. O mesmo efeito é observado, embora com menos intensidade, quando as capacidades de pesquisa são elevadas. Isso acontece sobretudo com o pesquisador do tipo 3, utilizado nos casos 1C, 2C e 3C.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.35	13059.7	-	-
100	**0.43**	8754.84	-	-
300	-	-	-	-

Tabela 4.5: Resultados missão 1, caso A.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.55	1364.27	**0.75**	1410.58
100	0.44	1048.93	0.68	2357.57
300	0.55	1177.06	0.69	21175,2

Tabela 4.6: Resultados missão 1, caso B.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.43	2419.31	0.42	8195.25
100	0.5	2734.05	**0.51**	96941.2
300	**0.51**	2719.77	-	-

Tabela 4.7: Resultados missão 1, caso C.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	**0.22**	1715.39	**0.22**	1718.09
100	0.14	1716.64	0.1	2554.08
300	**0.22**	1757.84	**0.22**	3025.43

Tabela 4.8: Resultados missão 2, caso A.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.14	1583.62	**0.23**	1718.09
100	**0.23**	1604.31	0.16	11671.7
300	**0.23**	1727.01	0.22	3025.43

Tabela 4.9: Resultados missão 2, caso B.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.33	1162.245	0.35	5156.83
100	0.33	1300.45	0.35	15981.16
300	**0.41**	1196.815	0.34	13689.9

Tabela 4.10: Resultados missão 2, caso C.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.53	2342.02	-	-
100	0.6	2335.23	-	-
300	**0.64**	5304.56	-	-

Tabela 4.11: Resultados missão 3, caso A.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.68	1676.69	0.74	2105.25
100	0.61	1652.21	0.75	10462.3
300	0.75	1956.62	**0.79**	21194.9

Tabela 4.12: Resultados missão 3, caso B.

	Tamanho da janela (f)			
	+1		+3	
Amostra	Avaliação	Tempo	Avaliação	Tempo
50	0.66	24982.6	-	-
100	**0.75**	124314	-	-
300	-	-	-	-

Tabela 4.13: Resultados missão 3, caso C.

A Tabela 4.14 mostra para cada missão e caso, qual o melhor valor de probabilidade obtido para o problema. Embora não possamos comparar com outro método presente na literatura, podemos afirmar que nossos resultados ficaram, em média, com o dobro do valor de probabilidade que o método atualmente aplicado pela DGATn, quando avaliados pelo mesmo simulador.

Um outro ponto interessante a ser avaliado é saber, para cada missão e para cada caso, como evolui a diferença entre o valor da probabilidade dada pelo modelo e o valor da probabilidade dada pelo simulador. Conduzimos nessa segunda fase testes que vão até 1000 alvos. A Tabela 4.15 mostra a diferença da média aritmética entre esses valores para cada missão e cada caso. Como esperado, o aumento do tamanho da amostra faz decrescer a disparidade entre o modelo e o simulador.

As Figuras 4.5-4.13 mostram, para cada missão e caso, como evolui essa diferença, enquanto a Figura 4.14 mostra uma tendência geral da média de todas as missões e casos. No eixo das ordenadas observamos o valor da probabilidade dado por cada método. No eixo das abcissas o tamanho amostral. Quando este tamanho alcança 1000 alvos vemos que na maior parte dos casos o erro entre o modelo e o simulador é inferior à 10%, o que também pode ser observado para a média (ver Figura 4.14). O resultado é encorajador, pois mesmo com um número pequeno de alvos em relação ao número teórico calculado na Seção 4.2.3.2, o resultado é suficientemente próximo para garantir uma utilização prática.

Além disso, vemos que mesmo problemas com um pequeno número de céulas (especificamente o caso B da missão 1, conforme Tabelas 4.2 e 4.15) podem apresentar uma margem de erro. Esse erro está presente devido a diversas limitações do modelo. Primeiramente, a discretização pode ser feita de maneira muito grosseira, o que prejudica na precisão das ações do pesquisador. Como no simulador os alvos são gerados em um espaço mais preciso, estes possuem mais possibilidades e portanto conseguem desviar do plano de busca calculado através do modelo.

Em segundo lugar, o modelo é responsável pelo cálculo das ações dos alvos fictícios e, como a função objetivo do modelo é maximizar os alvos capturados, este pode acabar levando os alvos à posições inseguras ou agrupá-los para maximizar um valor. Apesar dessa inconsistência ser contra-balanceada pela restrição que obriga os alvos a seguirem suas transições preferenciais, uma margem de erro é esperada por conta deste fator.

4.4 Conclusão

Apresentamos neste capítulo um modelo original que, a nosso conhecimento, trata-se do único modelo de programação matemática capaz de dar uma resposta precisa para o problema de busca de um alvo inteligente. Fazemos uma recapitulação bibliográfica e mostramos os modelos de Teoria dos Jogos que foram a inspiração para o modelo proposto.

Este modelo, definido por variáveis binárias e expressões (restrições e função objetivo) lineares, é baseado numa discretização espaço-temporal. No entanto o modelo possui uma dificuldade suplementar: modelar o comportamento incerto e ao mesmo tempo reativo do

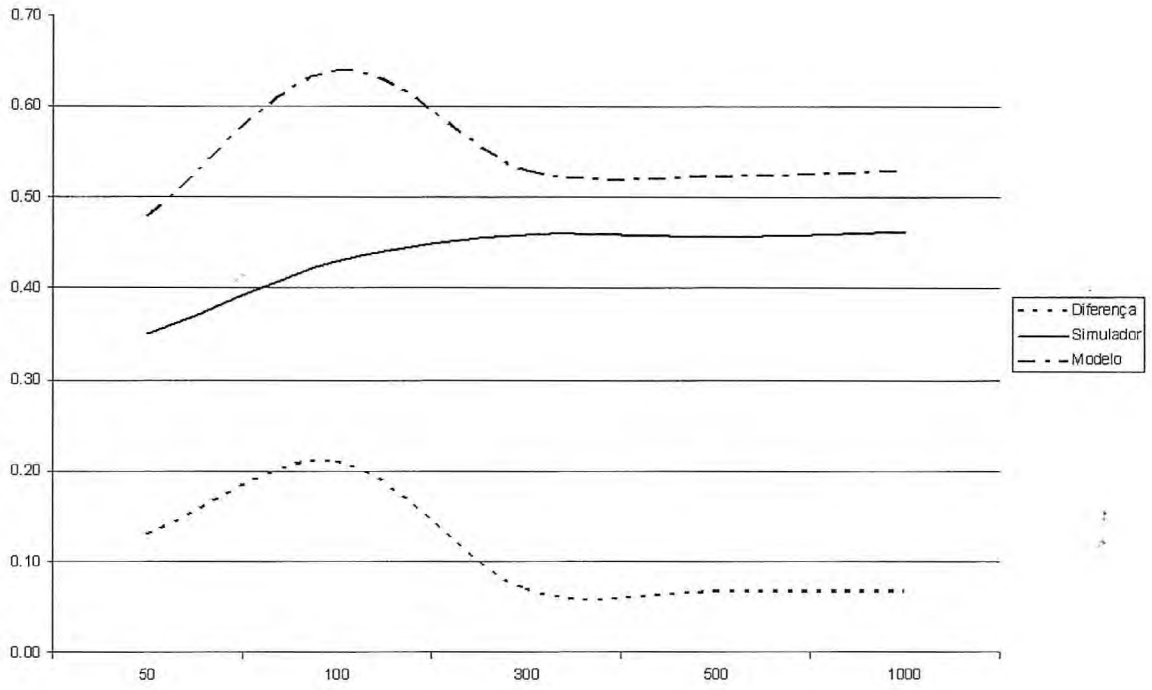

Figura 4.5: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 1, caso A.

alvo. Modelamos tal aspecto através da inserção no modelo da simulação de um certo número de alvos determinísticos cujo comportamento corresponde ao comportamento do alvo real. Esses alvos fazem o papel de amostra estatística do alvo real.

Analisamos o modelo e concluímos que seu tamanho pseudo-polinomial (que é função do tamanho do espaço amostral) não permite uma resolução direta através de um método de enumeração implícita. Com efeito, o número de alvos necessários para estabelecer uma confiança estatística (e margem de erro) suficiente é extremamente elevado.

Mesmo para um número pequeno (em comparação ao valor teórico) de alvos, uma decomposição do modelo se faz necessária para chegar à resolução de problemas reais. Em nosso contexto, o método de Janelas Deslizantes nos pareceu intuitivo e apropriado. Tal método foi utilizado com sucesso na literatura [Brown 1980, Washburn 1983].

Finalmente, desenvolvemos para efeitos de validação do modelo, um simulador capaz de, a partir de uma solução qualquer do problema, verificar qual o valor da probabilidade de encontrar o alvo, segundo as premissas feitas sobre o comportamento do alvo. Este simulador utiliza um número estatisticamente confiável à 95% e margem de erro de 3% para a avaliação das soluções.

Como resultados experimentais, tivemos a oportunidade de trabalhar junto ao Ministério da Defesa francês, que nos forneceu casos reais para os testes. Mostramos que

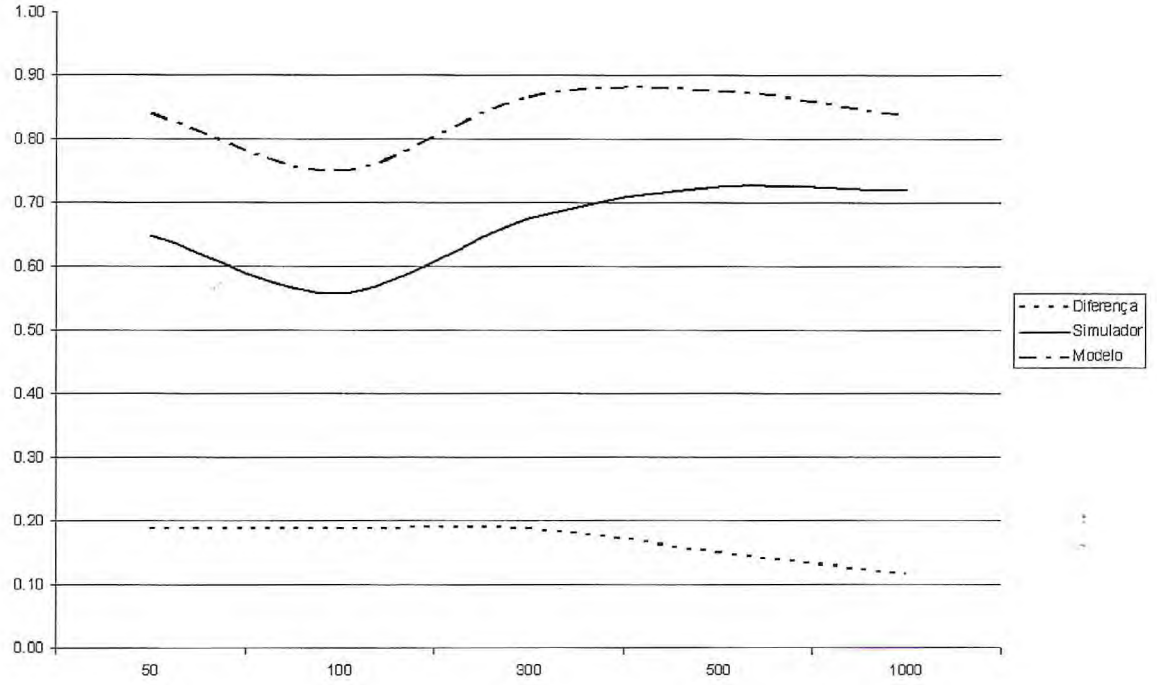

Figura 4.6: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 1, caso B.

mesmo com um número pequeno de alvos fomos capazes de fornecer uma solução de qualidade para o problema. Além disso, o número limitado de alvos permite, de qualquer maneira, que o valor da solução obtida através da resolução do modelo fique em uma margem aceitável do valor dado pelo simulador.

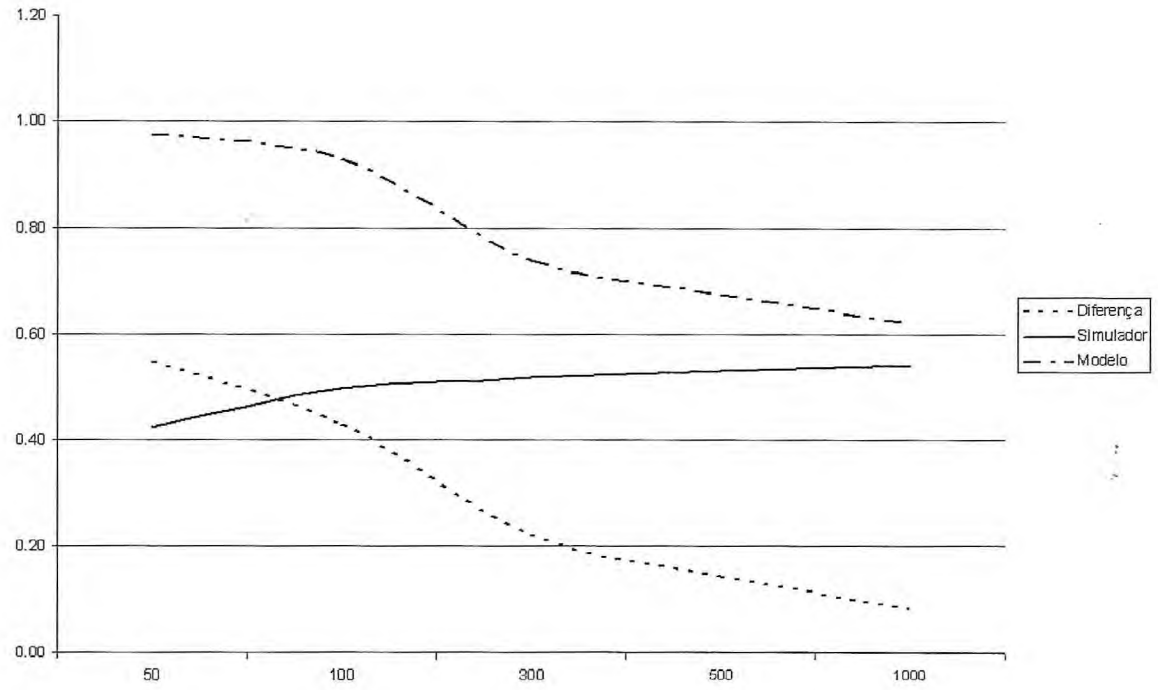

Figura 4.7: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 1, caso C.

		Tamanho da janela (f)	
Missão	Caso	+1	+3
	A	**0.43**	-
1	B	0.55	**0.75**
	C	**0.51**	**0.51**
	A	**0.22**	**0.22**
2	B	**0.23**	**0.23**
	C	**0.41**	0.35
	A	**0.64**	-
3	B	0.75	**0.79**
	C	**0.75**	-

Tabela 4.14: Resumo da avaliação do método.

Missão	Caso		Amostra 50	100	300	500	1000
1	A	Modelo	0.56	0.63	0.53	0.52	0.53
		Simulador	0.39	0.42	0.46	0.46	0.46
		Diferença	0.17	0.21	0.07	0.07	0.07
	B	Modelo	0.92	0.81	0.87	0.88	0.84
		Simulador	0.69	0.65	0.69	0.73	0.72
		Diferença	0.22	0.16	0.18	0.15	0.12
	C	Modelo	0.97	0.89	0.73	0.68	0.62
		Simulador	0.46	0.50	0.52	0.53	0.54
		Diferença	0.51	0.39	0.21	0.14	0.08
2	A	Modelo	0.20	0.17	0.25	0.18	0.19
		Simulador	0.22	0.13	0.22	0.22	0.21
		Diferença	0.02	0.04	0.03	0.04	0.02
	B	Modelo	0.28	0.27	0.26	0.23	0.24
		Simulador	0.18	0.21	0.25	0.24	0.27
		Diferença	0.10	0.06	0.03	0.01	0.03
	C	Modelo	0.52	0.52	0.43	0.43	0.41
		Simulador	0.32	0.34	0.37	0.41	0.41
		Diferença	0.20	0.17	0.06	0.03	0.01
3	A	Modelo	0.89	0.89	0.88	0.91	0.92
		Simulador	0.58	0.61	0.60	0.62	0.63
		Diferença	0.32	0.28	0.28	0.29	0.29
	B	Modelo	0.96	0.87	0.86	0.93	0.87
		Simulador	0.79	0.77	0.83	0.74	0.80
		Diferença	0.17	0.11	0.09	0.19	0.08
	C	Modelo	0.91	0.90	0.86	0.90	0.86
		Simulador	0.68	0.72	0.74	0.76	0.73
		Diferença	0.23	0.19	0.13	0.14	0.14

Tabela 4.15: Resumo da avaliação do método.

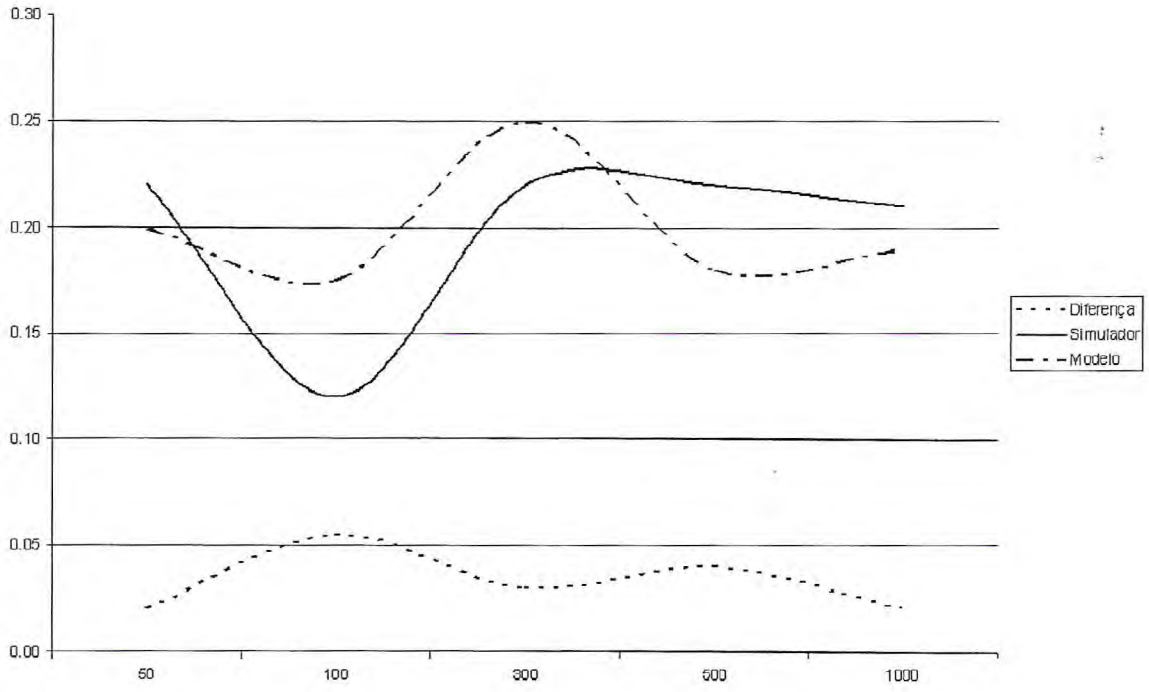

Figura 4.8: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 2, caso A.

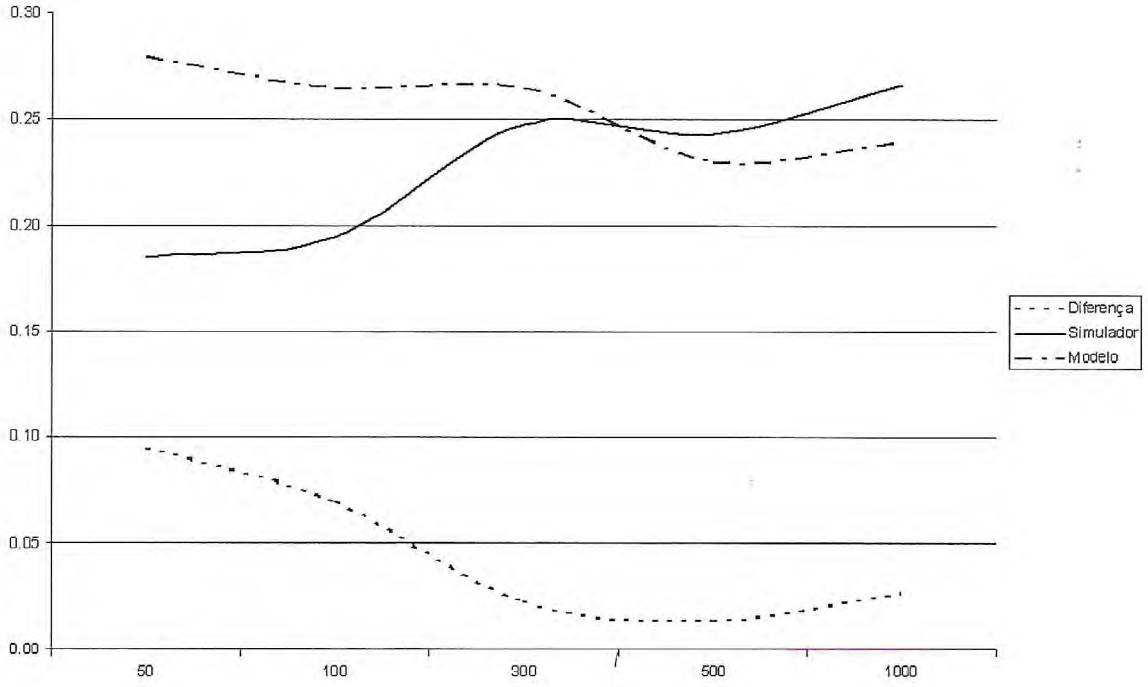

Figura 4.9: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 2, caso B.

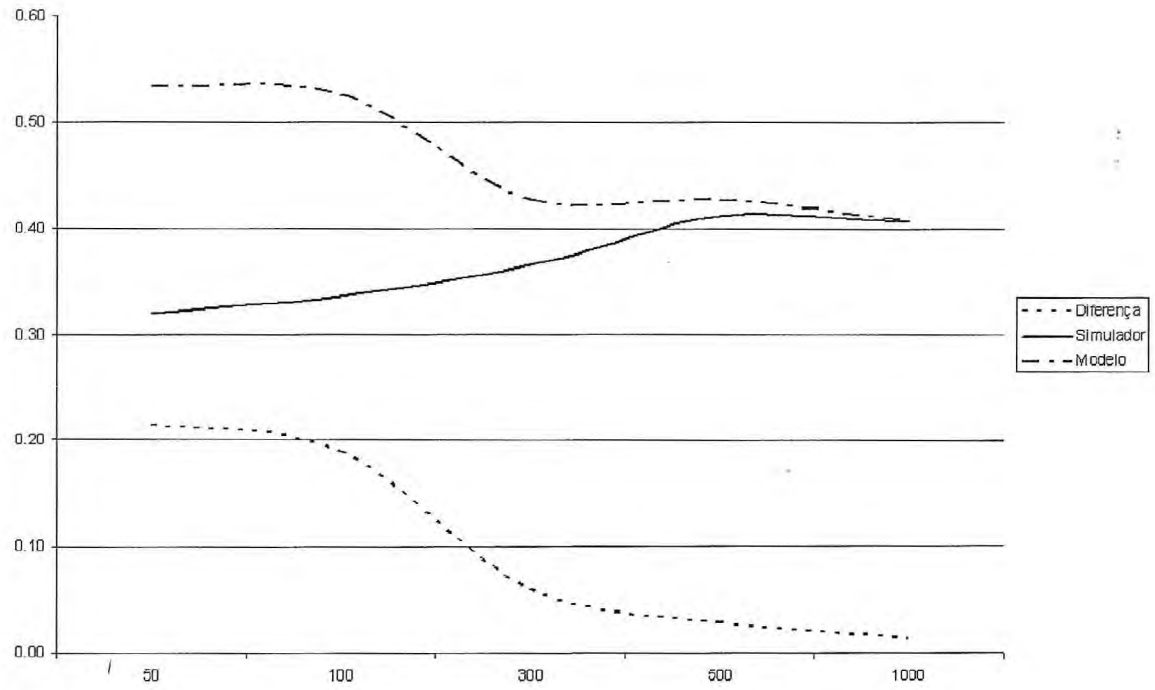

Figura 4.10: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 2, caso C.

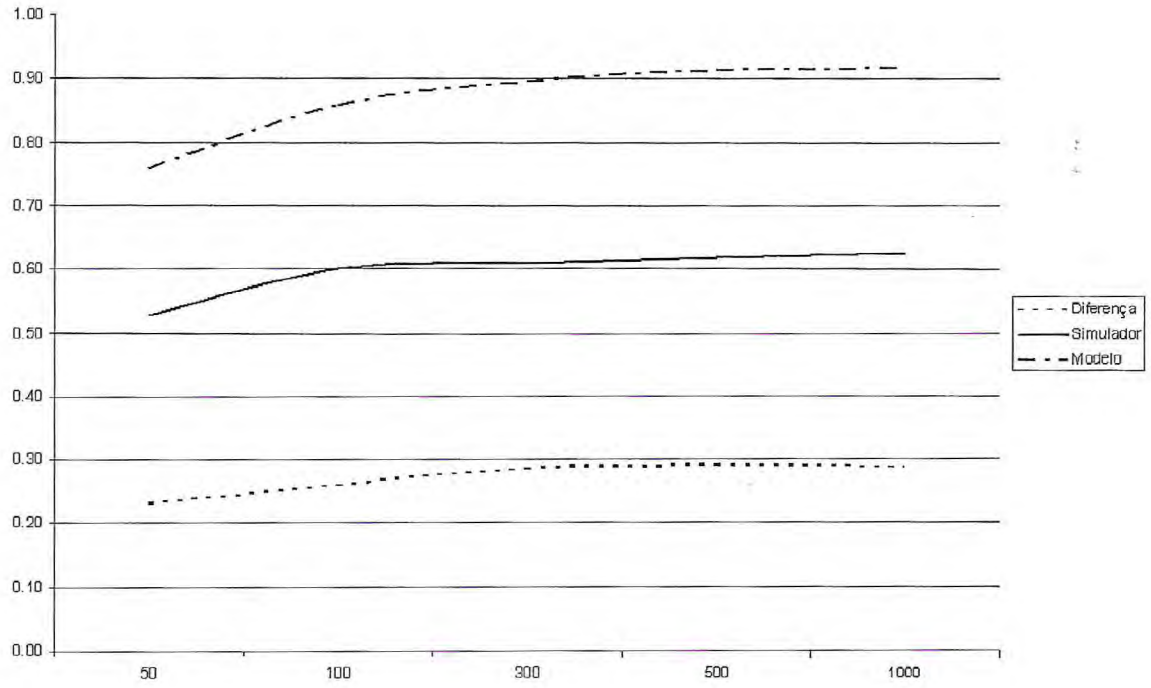

Figura 4.11: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 3, caso A.

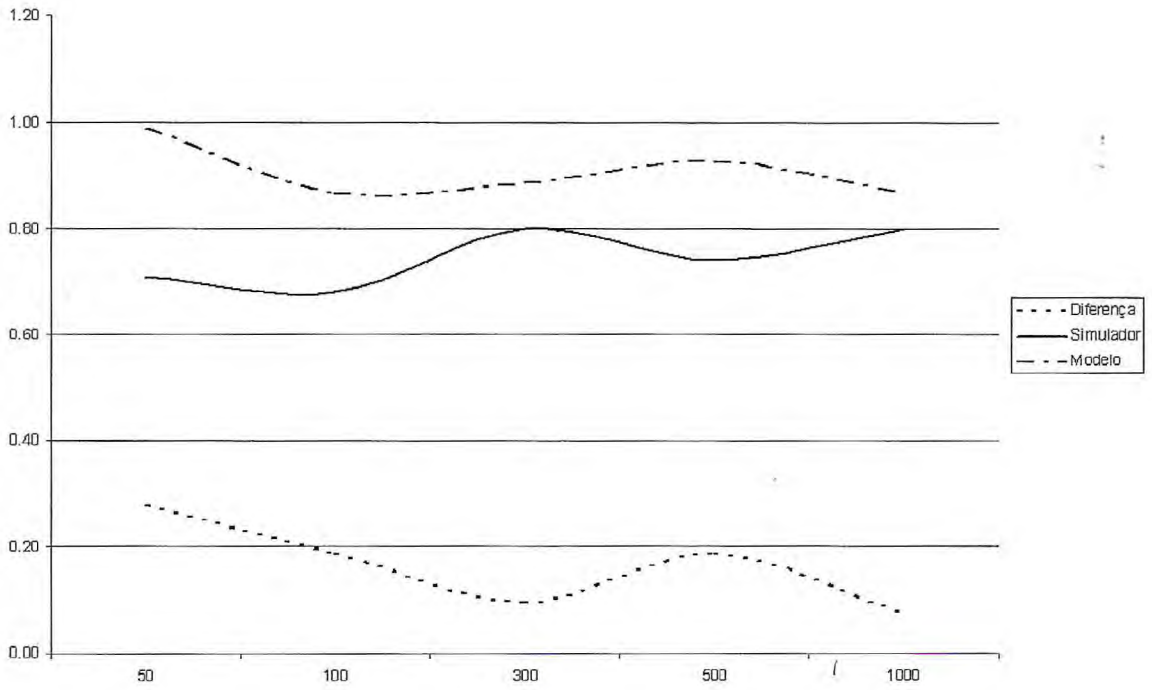

Figura 4.12: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 3, caso B.

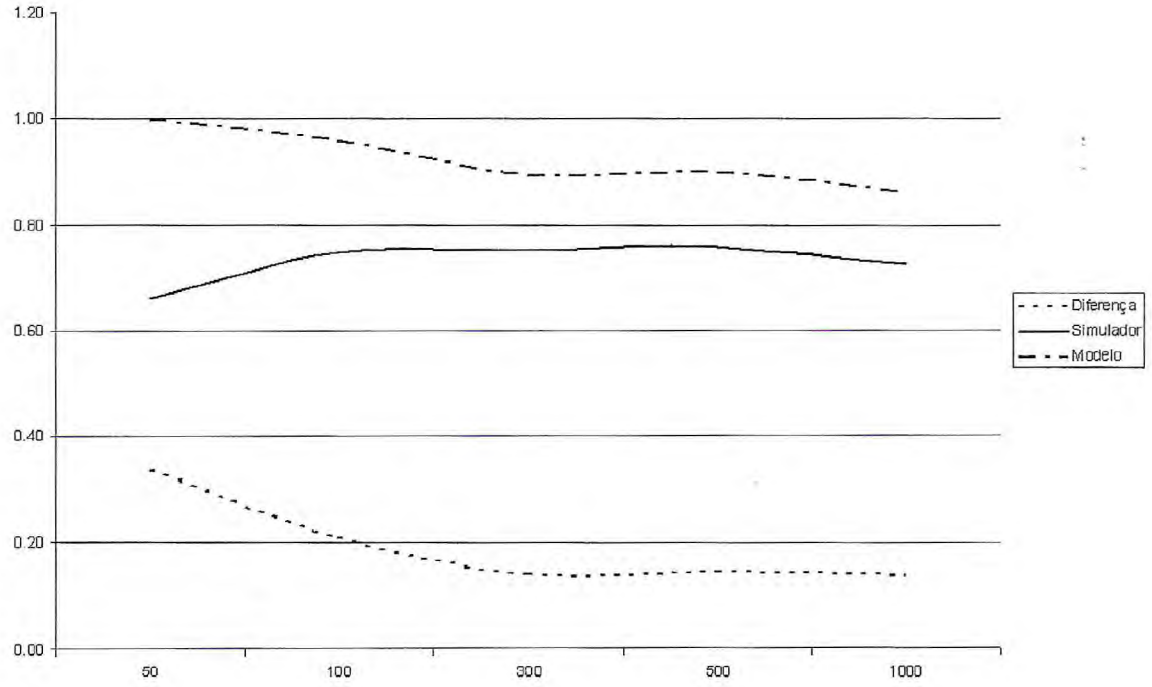

Figura 4.13: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos para a missão 3, caso C.

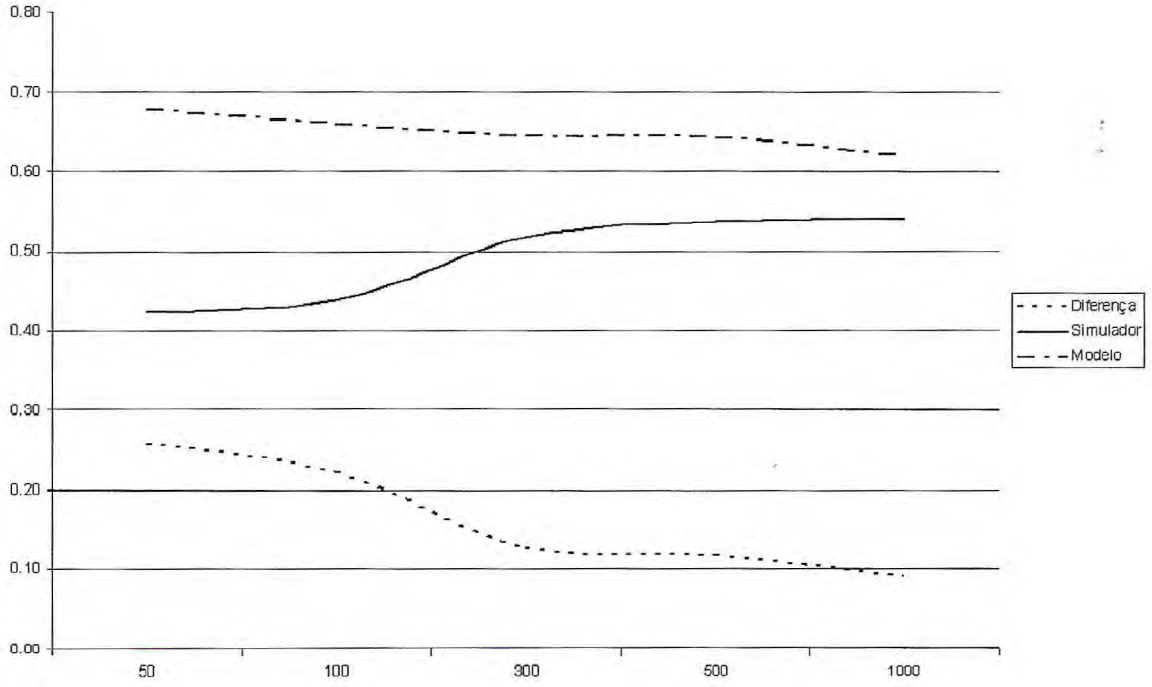

Figura 4.14: Evolução da avaliação do modelo e a avaliação do simulador segundo o número de alvos.

Capítulo 5
Conclusão

O conteúdo deste trabalho mostra diversos exemplos de como a cooperação multidisciplinar pode render resultados na resolução de problemas de otimização. Podemos dividir esta tese em duas partes principais. A primeira consiste de um estudo de dois problemas clássicos da Programação Matemática e a aplicação de técnicas de áreas adjacentes na resolução desses problemas. E a segunda trata da resolução de um problema que possui uma vasta literatura em domínios adjacentes à Programação Inteira e que aqui é tratado por essa disciplina de forma original.

O primeiro problema para o qual expomos os benefícios da cooperatividade é o Problema das Mochilas Múltiplas (MKP), tratado no Capítulo 2. Para este problema, revisitamos o esquema de amálgama entre a Programação Matemática e a Programação por Restrições proposto por [Oliva 2001]. Este esquema consiste na utilização simultânea de técnicas de resolução de ambos os paradigmas, onde a troca de informações entre eles possibilita um ganho na eficiência global do método de resolução.

Estendemos o esquema mútuo através da definição de uma nova restrição global, a qual chamamos *"knapsum"*. Essa restrição é definida por uma desigualdade linear associada à uma restrição de cardinalidade. Mostramos que a filtragem de restrições *knapsum* é rápida (tempo linear) e que é possível gerar planos de corte para o MKP a partir dessa filtragem.

Além disso, propomos uma atualização do esquema de enumeração implícita proposto por [Vimont 2008] que permite uma melhor exploração da árvore de busca. O ganho em performance associado às inovações aqui apresentadas permitiu-nos obter os melhores resultados em termos de tempo computacional para o problema.

Em seguida mudamos para uma disciplina levemente diferente ao abodarmos o Problema da Mochila Quadrática (QKP): trata-se da Programação Quadrática. Propomos um método de resolução que se baseia em uma linearização original aplicada ao problema apresentado por [Gallo 1980].

Contrariamente às linearizações clássicas, esta linearização adiciona somento uma nova variável, que substitui todo o termo quadrático da função objetivo. Isso é possível graças a uma caracterização do envelope convexo de uma restrição quadrática com variáveis binárias. Essa caracterização acarreta, no entanto, um número fatorial de restrições no modelo. Embora isto pareça desencorajante, podemos, através de uma técnica de geração de restrições, considerar este modelo para a resolução do QKP. Esta idéia deu a luz o cálculo de um limite superior original para o QKP. Esse resultado foi comparado ao melhor limite superior conhecido para o QKP [Billionnet 1999] e, embora nosso limite

superior não ultrapasse a qualidade daquele de [Billionnet 1999], a diferença reside em média à menos de 1% para as instâncias consideradas.

Mostramos ainda que as restrições advindas da caracterização proposta podem ser reforçadas no caso do QKP, considerando-se no momento de seus cálculos a existência de uma restrição de capacidade. Uma vez linearizado, mostramos que esse problema, que é a base da Programação Quadrática, também poderia se beneficiar das técnicas desenvoldidas para o MKP.

O resultado final mostra que a eficiência do método depende de uma característica fundamental: a densidade da matriz de coeficientes quadráticos. Com efeito, vemos que quanto mais esparsa é essa matriz, melhor é a performance de nosso método quando comparado ao melhor método encontrado na literatura, atribuído a [Pisinger 2005]. Além disso, para instâncias de tamanho reduzido, nosso método também é capaz de produzir os melhores resultados, segundo nossos experimentos.

Por fim, tratamos um problema prático: a busca de um alvo inteligente. Este trabalho foi desenvolvido conjuntamente com o Ministério da Defesa francês. O problema consiste na elaboração de um plano de pesquisa para encontrar um certo alvo cujo objetivo é de sempre se manter escondido. A natureza desse problema nos remete imediatamente à idéia de um jogo, um duelo matemático. De fato, esta disciplina, a Teoria da Busca, é comumente associada à Teoria dos Jogos [Brown 1980, Thomas 1991], embora tenha sido também em diversos outros ramos [Kan 1977, Garnaev 2000, Soares 2010].

Propomos, a nosso conhecimento, o primeiro modelo de Programação Inteira para o problema capaz de fornecer uma resposta precisa para o problema e que ao mesmo tempo é capaz de cobrir diversos casos presentes na literatura. Uma recapitulação bibliográfica do problema foi realizada afim apresentarmos as variantes possíveis para o problema e retirarmos da literatura a inspiração para a conceitualização do modelo.

De forma resumida, o modelo baseia-se na discretização espaço-temporal e na simulação do comportamento do alvo procurado, através de um número limitado de alvos determinísticos, porém reativos. Vimos que este modelo é poderoso no sentido descritivo, ou seja, ele é capaz de sintetizar diversos casos da literatura e tem um grande potencial de extensibilidade (restando apenas a confecção de novas restrições como expressões lineares). Porém, o modelo possui um número pseudo-polinomial de variáveis e restrições, sendo o tamanho do espaço amostral um fator da complexidade.

Fazemos a estimativa teórica do número de alvos necessários para atingir um nível de confiança e margem de erro razoáveis e constatamos que o número de variáveis e restrições do modelo é intratável por uma abordagem de resolução direta. Logo, propomos um método de decomposição para o modelo com o respaldo da literatura ([Brown 1980, Washburn 1983]): o método de Janelas Deslizantes.

Além do modelo, também conceitualizamos um simulador, capaz de avaliar um plano de pesquisa através da injeção de alvos determinísticos contra este plano. Esse simulador é usado para avaliar e validar as soluções fornecidas pelo modelo. Mesmo usando um número bastante reduzido de alvos, em relação ao limite teórico estabelecido, experimentos computacionais mostram que o modelo é bastante difícil de ser resolvido. No entanto,

podemos afirmar, através das experiências numéricas, que a solução calculada possui boas características e o valor da probabilidade é próximo da solução calculada com um número mais elevado de alvos.

Tivemos a oportunidade de testar casos reais providos pelo Ministério da Defesa francês. O modelo foi capaz de obter uma solução para cada uma das instâncias fornecidas, melhor que aquela adotada atualmente pelo Ministério. Mostramos que, segundo as características da missão e àquelas atribuídas ao pesquisador, a performance e, portanto, a parametrização do modelo é diference. Apesar de não sermos capazes de chegar ao limite teórico estabelecido, mostramos como evolui a diferença entre o valor obtido pelo modelo e aquele calculado pelo simulador concebido e concluímos que o modelo é capaz de fornecer uma solução de qualidade para os problemas propostos.

Em resumo, os esquemas cooperativos multidisciplinares aqui propostos foram capazes de obter resultados significativos quando comparados à literatura e satisfatórios quando aplicados à problemas práticos.

Bibliografia

[Baston 1989] V.J. Baston and F.A. Bostock. *A One-Dimensional Helicopter-Submarine Game.* Naval Research Logistics, vol. 36, 1989. 66

[Beasley 1990] J.E. Beasley. *OR-Library: Distributing test problems by electronic mail.* Journal of the Operational Research Society, vol. 41, no. 11, pages 1069–1072, 1990. 1, 25

[Bienstock 1991] D. Bienstock. *Graph searching, path-width, tree-width and related problems (a survey).* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1991. 66, 67

[Billionnet 1996] A. Billionnet and F. Calmels. *Linear programming for the 0-1 quadratic knapsack problem.* European Journal of Operational Research, vol. 92, no. 2, pages 310–325, 1996. 40, 48, 49, 52

[Billionnet 1999] A. Billionnet, A. Faye and E. Soutif. *A new upper bound for the 0-1 quadratic knapsack problem.* European journal of operational research, vol. 112, no. 3, pages 664–672, 1999. 3, 38, 40, 53, 54, 59, 60, 105, 106

[Billionnet 2004] A. Billionnet and É. Soutif. *An exact method based on Lagrangian decomposition for the 0-1 quadratic knapsack problem.* European Journal of Operational Research, vol. 157, no. 3, pages 565–575, 2004. 39, 55

[Boussier 2008] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi and P. Michelon. *Solving the 0-1 Multidimensional Knapsack Problem with Resolution Search.* VI ALIO/EURO Worshop on Applied Combinatorial Optimization, 2008. 8, 26

[Boussier 2009] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi and P. Michelon. *A multi-level search strategy for the 0-1 Multidimensional Knapsack Problem.* Discrete Applied Mathematics, 2009. 2, 8, 10, 26, 28

[Brown 1980] S.S. Brown. *Optimal Search for a Moving Target in Discrete Time and Space.* Operations Research, vol. 28(6), pages 1275–1289, 1980. 4, 66, 68, 69, 86, 94, 106

[Caprara 1998] A. Caprara, D. Pisinger and P. Toth. *Exact solution of the quadratic knapsack problem.* 1998. 52, 55

[Chaovalitwongse 2004] W. Chaovalitwongse, P.M. Pardalos and O.A. Prokopyev. *A new linearization technique for multi-quadratic 0-1 programming problems.* Operations Research Letters, vol. 32, no. 6, pages 517–522, 2004. 2, 41

[Danskin 1968] J.M. Danskin. *A Helicopter versus Submarine Search Game.* Operations Research, vol. 16, pages 509–517, 1968. 68

[Dixon 1957] W.J. Dixon and F.J. Massey. *Introduction to statistical analysis.* 1957. 84

[Eagle 1984] J.N. Eagle. *The Optimal Search for a Moving Target When the Search Path Is Constrained.* Operations Research, vol. 32, 1984. 69

[Fomin 1998] F.V. Fomin. *Helicopter search problems, bandwidth and pathwidth.* Discrete Applied Mathematics, 1998. 66

[Gal 1979] S. Gal. *Search Games with Mobile and Immobile Hider.* SIAM Journal on Control and Optimization, vol. 17, no. 1, pages 99–122, 1979. 77

[Gallo 1980] G. Gallo, PL Hammer and B. Simeone. *Quadratic knapsack problems.* Combinatorial Optimization, pages 132–149, 1980. 2, 39, 105

[Garfinkel 1972] R. Garfinkel and G.L. Nemhauser. Integer programming. John Wiley & Sons, 1972. 22

[Garnaev 1993] A.Y. Garnaev. *A Remark on a Helicopter-Submarine Game.* Naval Research Logistics, vol. 40, pages 745–753, 1993. 66

[Garnaev 2000] A.Y. Garnaev. Search games and other applications of game theory, 2000. 68, 71, 106

[Glover 1975] F. Glover. *Improved linear integer programming formulations of nonlinear integer problems.* Management Science, pages 455–460, 1975. 41

[Glover 2008] F. Glover and H.D. Sherali. *Second-order cover inequalities.* Mathematical Programming, vol. 114, no. 2, pages 207–234, 2008. 10, 20

[Gueye 2005] S. Gueye and P. Michelon. *Miniaturized Linearizations for Quadratic 0/1 Problems.* Annals of Operations Research, vol. 140, no. 1, pages 235–261, 2005. 41

[Gueye 2009] S. Gueye and P. Michelon. *A linearization framework for unconstrained quadratic (0-1) problems.* Discrete Applied Mathematics, vol. 157, no. 6, pages 1255–1266, 2009. 2, 41

[Hohzaki 1999] R. Hohzaki and K. Iida. *A Solution for a Two-Person Zero-Sum Game with a Concave Payoff Function.* Nonlinear Analysis and Convex Analysis, World Science Publishing Co., London, pages 157–166, 1999. 4, 72, 76

[Hohzaki 2000] R. Hohzaki and K. Iida. *A Search Game When a Search Path Is Given.* European Journal of Operational Research, vol. 124, pages 114–124, 2000. 4, 76

[Hohzaki 2006] R. Hohzaki. *Search Allocation Game.* European Journal of Operational Research, vol. 172, pages 101–119, 2006. 4, 71, 72

[Hooker 1999a] J.N. Hooker and M.A. Osorio. *Mixed logical-linear programming.* Discrete Applied Mathematics, vol. 96, pages 395–442, 1999. 10

[Hooker 1999b] J.N. Hooker, G. Ottosson, E.S. Thorsteinsson and H.J. Kim. *On integrating constraint propagation and linear programming for combinatorial optimization.* In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pages 136–141, 1999. 1

[Hooker 2002] J.N. Hooker. *Logic, optimization, and constraint programming.* INFORMS Journal on Computing, vol. 14, no. 4, pages 295–321, 2002. 10

[Kan 1977] Y.C. Kan. *Optimal Search of a Moving Target.* Operations Research, vol. 25(5), 1977. 4, 69, 106

[Koopman 1956a] B.O. Koopman. *The theory of Search: part I, Kinematic Bases.* Operations Research, vol. 4, pages 324–346, 1956. 4

[Koopman 1956b] B.O. Koopman. *The theory of Search: part II, Target Detection.* Operations Research, vol. 4, pages 503–531, 1956. 4, 65

[Koopman 1957] B.O. Koopman. *The theory of Search: part III, the optimum distribution of search effort.* Operations Research, vol. 5, pages 613–626, 1957. 4, 65

[Koopman 1980] B.O. Koopman. *Search and Screening.* Pergamon, pages 221–227, 1980. 65

[Krumbein 1965] W.C. Krumbein and F.A. Graybill. An introduction to statistical models in geology. McGraw-Hill, 1965. 84

[L. Blin 2008] N. Nisse L. Blin P. Fraigniaud and S. Vial. *Distributed Chasing of Network Intruders.* Theoretical Computer Science, vol. 399, pages 12–37, 2008. 66

[Létocart 2009] L. Létocart, A. Nagih and G. Plateau. *Reoptimization in lagrangian methods for the quadratic knapsack problem.* Rapport technnique LIPN, 2009. 39

[Lueker 1975] G.S. Lueker. *Two NP-complete problems in nonnegative integer programming.* Comput. Sci. Lab., Univ. Princeton, Rep, vol. 178, 1975. 39

[Martello 1990] S. Martello and P. Toth. Knapsack problems: algorithms and computer implementations. 1990. 52

[Meinardi 1964] J.J. Meinardi. *A Sequentially Compounded Search Game.* Theory of Games: Techniquea and Applications, The English Universities Press, London, pages 285–299, 1964. 68

[Michelon 1996] P. Michelon and L. Veilleux. *Lagrangean methods for the 0-1 quadratic knapsack problem.* European Journal of Operational Research, vol. 92, no. 2, pages 326–341, 1996. 39, 55

[Milano 2004] M. Milano and M. Trick. *Constraint and integer programming.* Constraint and Integer Programming: Toward a Unified Methodology, page 1, 2004. 1

[Nisan 2007] N. Nisan. Algorithmic game theory. Cambridge Univ Pr, 2007. 1

[Oliva 2001] C. Oliva, P. Michelon and C. Artigues. *Constraint and linear programming: Using reduced costs for solving the zero/one multiple knapsack problem.* In Proc. of the Workshop on Cooperative Solvers in Constraint Programming (CoSolv 01), Paphos, Cyprus, pages 87–98, 2001. 1, 10, 12, 26, 105

[Osorio 2001] M.A. Osorio and F. Glover. *Logic cuts using surrogate constraint analysis in the multidimensional knapsack problem.* In Third International Workshop on Integration of AI and OR Techniques (CPAIOR01), http://www-icparc. doc. ic. ac. uk/cpAIOR01, 2001. 10

[Osorio 2002] M.A. Osorio, F. Glover and P. Hammer. *Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions.* Annals of Operations Research, vol. 117, no. 1, pages 71–93, 2002. 10, 12

[Pisinger 2005] D. Pisinger, A.B. Rasmussen and R. Sandvik. *Solution of large-sized quadratic knapsack problems through aggressive reduction.* INFORMS Journal on Computing, 2005. 3, 39, 40, 53, 54, 55, 57, 59, 60, 106

[Quadri 2009] D. Quadri, E. Soutif and P. Tolla. *Exact solution method to solve large scale integer quadratic multidimensional knapsack problems.* Journal of combinatorial optimization, vol. 17, no. 2, pages 157–167, 2009. 39

[R. Hohzaki 2006] T. Komiya R. Hohzaki N. Ohsiro and E. Fukuda. *A Search Game With Durable Searching Resources.* Scientiae Mathematicae Japonicae, vol. 19, pages 1125–1141, 2006. 68

[Sherali 1999] H.D. Sherali and W.P. Adams. A reformulation-linearization technique for solving discrete and continuous nonconvex problems. Kluwer Academic Publishers, 1999. 2

[Soares 2010] R. P. Soares. *Procura em Grafos.* Dissertação de Mestrado, 2010. 68, 106

[Thomas 1991] L. C. Thomas and A. R. Washburn. *Dynamic Search Games.* Operations Research, vol. 39(3), pages 415–422, 1991. 4, 66, 69, 70, 106

[Vasquez 2001] M. Vasquez and J.K. Hao. *A hybrid approach for the 0-1 multidimensional knapsack problem.* In International Joint Conference on Artificial Intelligence, volume 17, pages 328–333. Citeseer, 2001. 10

[Vimont 2008] Y. Vimont, S. Boussier and M. Vasquez. *Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem.*

Journal of combinatorial optimization, vol. 15, no. 2, pages 165–178, 2008. 10, 17, 18, 27, 105

[Washburn 1980] A.R. Washburn. *Search-Evasion Game in a Fixed Region.* Operations Research, vol. 28, pages 1290–1298, 1980. 68

[Washburn 1983] A.R. Washburn. *Search for a Moving Target: The FAB Algorithm.* Operations Research, vol. 31(4), pages 739–751, 1983. 66, 69, 94, 106

[Washburn 2001] A.R. Washburn and R. Hohzaki. *The Diesel Submarine Flaming Datum Problem.* Military Operations Research, vol. 4, pages 19–30, 2001. 65

[Wolsey 1998] L.A. Wolsey. Integer programming, 1998. 19