Branch-and-Bound para problemas de Otimização Combinatória

Rafael Antônio Marques Gomes, Haroldo Gambini Santos
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: rafael.amgomes@gmail.com, haroldo.santos@gmail.com

***Resumo*—Este trabalho apresenta detalhes do estudo do método *branch-and-bound*, mais especificamente o método e a sua importância para o desenvolvimento de uma formulação estendida de Programação Inteira de uma notável variante do clássico Problema de Escalonamento de Enfermeiras, o qual leva em consideração a distribuição das enfermeiras em turnos em um determinado período. Como os serviços de enfermagem possuem um dos maiores componentes de custo nos orçamentos de hospitais, é essencial que se desenvolva uma boa política de planejamento. Em particular, é muito importante utilizar o tempo e esforço eficientemente, tanto para distribuir o mais uniformemente possível a carga horária entre as enfermeiras quanto atender preferências pessoais.**

***Keywords*-Branch-and-Bound, Otimização, Programação Inteira, Escalonamento de Enfermeiras.**

I. Introdução

A busca por um algoritmo que resolva genericamente o problema de alocação de enfermeiras é muito difícil principalmente pelas suas características idiossincráticas. A simples consideração de disponibilidade limitada de equipe para alocação de enfermeiras muitas vezes já torna o problema tão difícil quanto problemas NP-difíceis como o problema de coloração de vértices em grafos [1]. A utilização do método de *Branch-and-Bound* na busca de soluções satisfatórias para esta abordagem, têm sido usada com sucesso para resolver problemas similares de maneira exata.

O objetivo desta pesquisa é analisar os princípios do funcionameno do método de *Branch-and-Bound* para Programação Inteira, a maneira como este método separa o problema original em novos subproblemas (processo de ramificação) porém mais restritos que facilitam a solução final através da geração de limites.

O artigo está organizado da seguinte forma. A Seção III descreve a metodologia empregada no trabalho, em seguida a Seção IV apresenta uma análise de complexidade do método apresentado, enquanto que a Seção V mostra os experimentos realizados e por fim a Seção VI discute os resultados obtidos.

II. Fundamentação Teórica

A. Trabalhos relacionados

Uma dificuldade é a comparação entre as abordagens envolvidas visto que muitas vezes são consideradas instâncias diferentes para o mesmo problema. De fato, o critério que define a qualidade (e mesmo a viabilidade) de um quadro de horários é bastante dependente do sistema hospitalar específico a ser trabalhado. Entretanto, algumas considerações são comuns na maioria dos trabalhos encontrados na literatura, como as preferências dos personagens envolvidos: as enfermeiras, em geral, preferem trabalhar uma certa sequência de dias no mesmo horário/turno na semana.

Visando oferecer um estímulo na área e uma análise precisa dos métodos de solução, foi proposta em 2010 a Primeira Competição Internacional de Escalonamento de Enfermeiras, a *International Nurse Rostering Competition (INRC)* [2]. Esta competição procurava incentivar a pesquisa no campo de programação de horários e estabelecer um problema *benchmark* que fosse amplamente aceito pela comunidade de programação de horários além de permitir a pesquisadores contrastarem seus resultados de forma precisa e justa através de um conjunto de instâncias de entrada disponibilizadas pela INRC.

Nesta competição, Burke e Curtois [3] apresentaram uma abordagem do modelo e os resultados de dois algoritmos aplicados para as instâncias da INRC. O primeiro algoritmo foi aplicado a um determinado conjunto de instâncias onde se fundamentou na técnica de Ejeção de Cadeias. O segundo algoritmo, o método de *branch and price*, foi aplicado para outro conjunto de instâncias que permitia um tempo maior de execuçao. O método de *branch and price* é um método onde cada nó da árvore de *branch and bound* é uma relaxação na programação linear que é resolvida utilizando geração de colunas. A implementação clássica de geração de colunas utiliza um problema mestre e subproblemas geradores de colunas. O problema problema mestre restrito (PMR), através de suas variáveis duais, direciona os subproblemas na busca por novas colunas que trazem informações novas para o PMR. A técnica de geração de colunas também foi utilizada na abordagem proposta por Bulog [4] para a competição de escalonamento de enfermeiras.

Lü e Hao [5] utilizaram um algoritmo que pode ser considerado uma busca-local multi-início. Ele muda alternadamente entre um procedimento de busca local e um mecanismo de reinício. A partir de uma solução inicial factível gerada aleatoriamente, o procedimento de busca local é

utilizado para melhorar a solução atual, minimizando o custo das violações das restrições fracas. Quando o procedimento de busca local não pode melhorar a qualidade da solução dentro de um determinado número de iterações, mais uma rodada do procedimento de busca local é utilizada para otimizar ainda mais a qualidade da solução.

Nonobe [6] utilizou um algoritmo baseado em metaheurística para o problema de otimização por restrições (COP), neste caso a busca tabu. Dado um número de restrições e seus pesos, COP procura encontrar uma atribuição de valores a variáveis tais que a soma das penalidades de restrições violadas sejam minimizadas.

Anteriormente a INRC, Dias *et al.* [7] desenvolveram uma busca tabu e um algoritmo genético para resolver problemas de escalonamento de enfermeiras em hospitais brasileiros. Os testes mostraram que o algoritmo genético ligeiramente superou a busca tabu, mas na prática, ambas as abordagens foram bem aceitas pelos usuários onde estes se mostraram significativamente superiores às montagens manuais de escalas. A disponibilização de uma interface gráfica para ajustes manuais de alterações no cronograma também permitiram agilidade e melhor aceitação por parte dos usuários.

B. Programação Inteira

Problemas de Programação Linear (PL), no âmbito matemático, são problemas de otimização nos quais a função objetivo e as restrições são todas do tipo linear.

Programação Linear é uma importante área da otimização por várias razões, como por exemplo, muitos problemas práticos em pesquisa operacional podem ser expressos como problemas de programação linear. Estes problemas incluem: Planejamento de Produção [8], Predição de estrutura de proteínas e Projeto de Medicamentos [9], Programação de horários [10] entre outros.

Geometricamente, as restrições lineares definem um poliedro convexo conforme Figura 1, que é chamado de conjunto dos pontos viáveis. Uma vez que a função objetivo é também linear, todo ótimo local é automaticamente um ótimo global. A função objetivo ser linear também implica que uma solução ótima pode apenas ocorrer em um ponto da fronteira do conjunto de pontos viáveis.

Se todas as variáveis do problema pertencerem ao conjunto dos números inteiros, temos uma variação da Programação Linear chamada Programação Inteira (PI) ou Programação Linear Inteira (PLI). Estas variáveis, quando sendo especificamente do tipo binárias, são frequentemente utilizadas na otimização, pois traduzem problemas de decisão, conhecidas também como variáveis de decisão sim-ou-não restritas a dois valores, 0 e 1. Portanto, a j-ésima decisão sim-ou-não seria representada por:

$$x_j = \begin{cases} 1, \textit{ se a decisão for sim;} \\ 0, \textit{ caso contrário;} \end{cases}$$

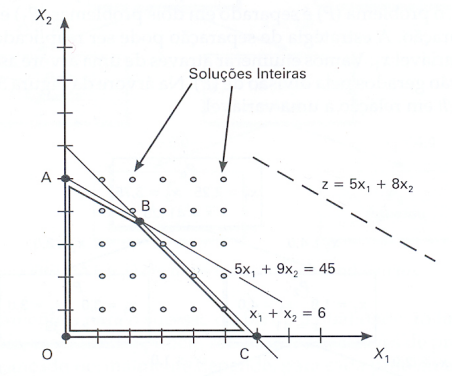

Figura 1. Poliedro (bidimensional) resultante das condições de um PPL.

Quando somente algumas das variáveis são inteiras e outras contínuas, temos a "Programação Inteira Mista" (PIM).

Ao contrário da PL que pode-se encontrar a solução ótima em um tempo razoável, muitos problemas de Programação Inteira são considerados NP-difíceis. Se as variáveis forem binárias, ou seja, assumirem somente os valores zero ou um, temos um caso especial da PI, que também pode ser classificado como NP-dificil.

A análise de pior caso, no entanto, não impediu que nas últimas décadas ocorressem progressos no desenvolvimento de algoritmos computacionais capazes de resolver consistentemente diversas instâncias relevantes de vários problemas em otimização combinatória. Esses avanços ocorrem em diferentes frentes. Avanços em Programação Linear [11] e no estudo da teoria poliédrica [12] permitiram que PI fosse aplicada com sucesso em problemas de grande porte. Problemas com milhões de variáveis binárias foram resolvidos. Esses resultados, entretanto, são relacionados a problemas específicos.

Para um mesmo problema de otimização combinatória existem diversas formulações de Programação Inteira válidas. Para o Problema do Caixeiro Viajante, por exemplo, vários artigos como [13] e [14] discutem o desempenho computacional bem como as qualidades teóricas das diferentes formulações. Idealmente, as restrições de uma formulação devem definir com exatidão a envoltória convexa dos pontos inteiros factíveis como ilustrado na Figura 2. Nesse caso o problema seria resolvido como um programa linear. Essa abordagem, entretanto, pode requerer um número muito grande de restrições, não sendo prática na maioria dos casos. Desse modo, buscam-se formulações que sejam boas aproximações da envoltória convexa. Uma formulação F1 é dita mais forte do que uma F2 caso F1 esteja estritamente contida em F2. Formulações mais fortes apresentam melhores limites duais, o que restringe o crescimento da árvore de busca com uma poda mais efetiva.

Um modelo de Programação Linear Inteira, pode ser ilustrado como:

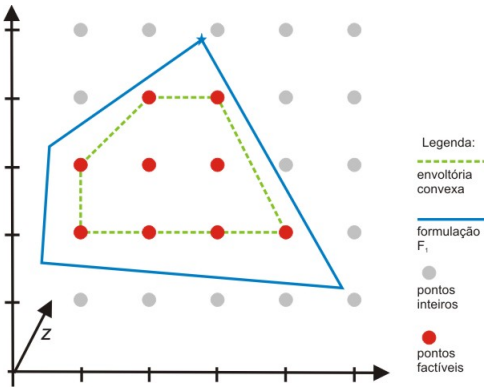

Figura 2. Uma formulação ideal definida pela envoltória convexa e uma formulação fraca.

maximizar $21x_1 + 11x_2$
sujeito a $7x_1 + 4x_2 <= 13$
$x_1, x_2 >= 0$
x_1, x_2 inteiros

C. O método Branch-and-Bound

Blazewicz *et al.* [15] resumiram o princípio do método *Branch-and-Bound* (BB) como uma enumeração de todas as soluções viáveis de um problema de otimização combinatorial de minimização ou maximização, tal que propriedades ou atributos não compartilhados por qualquer solução ótima são detectados previamente. Um atributo, chamado também de ramo da árvore de enumeração (ilustrdo na Figura 3), define um subconjunto do conjunto de todas as soluções viáveis do problema original onde cada elemento do subconjunto satisfaz este atributo.

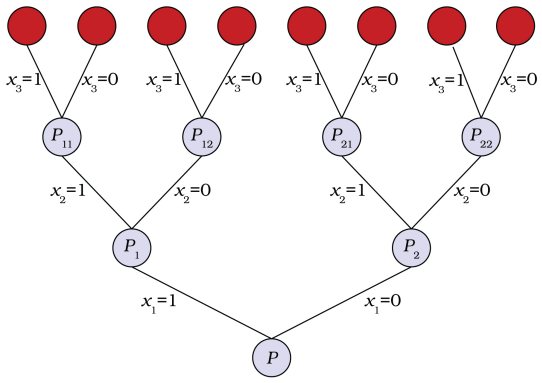

Figura 3. Árvore de enumeração: Problema com 3 variáveis binárias x_1, x_2, x_3

Apenas uma fração das soluções factíveis é realmente examinada, visto que, conforme afirma Render [16], o método BB é um algoritmo que busca por uma solução ótima através do exame de somente uma pequena parte do número total de possíveis soluções, pois ele trabalha quebrando o espaço de soluções viáveis em subproblemas menores até que uma solução ótima seja alcançada.

O termo *branch* refere-se ao fato de que o método efetua partições no espaço das soluções, ou seja, divide um problema em problemas menores. Uma maneira de se criar estes subproblemas usando fixação de variáveis discretas. A fixação recursiva de diferentes variáveis cria uma árvore, onde temos:

- **Nós internos:** esses nós representam todas as soluções que podem ser obtidas respeitando as fixações já feitas;
- **Folhas:** representam soluções completas

Se fosse considerada somente a etapa do *branch*, teríamos um algoritmo exato que em um número finito de passos forneceria uma solução ótima para o problema, mas isto na prática é extremamente ineficiente, visto que para n variáveis binárias temos 2^n nós a serem explorados, o que será detalhando na Seção IV como sendo de alto custo computacional. Desta maneira, surge o termo *bound* que ressalta a poda de algumas sub-árvores através do uso de limites calculados ao longo da enumeração, que resultará na prova da otimalidade da solução ou da sua infactibilidade.

Considerando um problema onde queremos achar o lucro máximo da solução ótima:

$$z = max\ f(x)$$

Mesmo que z seja difícil de calcular, eventulamente podemos ter limites, conforme ilustrados na Figura 4, para z que possa ser calculado com mais facilidade.

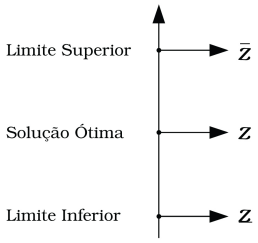

Figura 4. Limites.

A idéia geral da aplicação deste método no problema proposto é no princípio da relaxação do problema de programação inteira. Esta etapa da relaxação é executada em nós internos da árvore, considerando algumas fixações de variáveis. Este procedimento oferece um Limite Inferior (LI) e um limite superior (LS) para esses nós.

Existe ainda o conceito de solução incumbente do problema, que é a melhor solução encontrada durante a busca. Essa solução pode aparecer durante a execução de uma heurística ou durante o percurso na árvore (ao se chegar em nós folha) onde em um problema de maximização, temos um Limite Inferior para o problema. A solução de um problema

relaxado oferece uma solução cujo lucro é melhor ou igual ao da solução ótima do problema original, o que também se tratando de maximização, temos um Limite Superior.

Existem duas razões pelas quais a poda de um nó se faz necessária. A primeira diz respeito ao limite, onde por exemplo a relaxação (Limite Superior) indica que não há possibilidade de se melhorar a solução incumbente e outra razão é a infactibilidade onde as fixações já feitas induzem a uma situação infactível. Em ambos os casos, o nó e todo os seus filhos são podados.

Um exemplo de implementação de um algoritmo de *Brach-and-Bound* pode ser representado conforme o pseudocódigo a seguir:

PSEUDOCÓDIGO *Branch-and-Bound*

```
1  Iniciar com algum problema P0
2  Faça S = P0, o conjunto de subproblemas ativos
3  melhor_agora = oo
4  i = 0
5  Repita enquanto S não está vazio:
6      Escolha um subproblema (solução parcial) P
       pertencente a S e o remova de S
7      Expandí-lo em subproblemas menores P1, P2,..., Pk
8      Para cada Pi:
9      Se Pi é uma solução completa
10         atualizar melhor_agora
11     senão se LimiteInferior(Pi) < melhor_agora
12         adicione Pi para S
13 Retorne melhor_agora
```

III. MÉTODOS

O método a ser seguido neste trabalho pode ser descrito em três fases. Primeiro será avaliado o Problema apresentado na Competição Internacional, em seguida, uma formulação será desenvolvida para o problema e após esta etapa, os resultados obtidos serão validados no *software* de validação disponibilizado para medir a qualidade do modelo proposto.

A. O Problema

O problema de escalonamento de enfermeiras [2] envolve a atribuição de turnos às enfermeiras levando em conta várias restrições. Em geral, devemos considerar dois tipos básicos de restrições [17]:

- **Restrições Rígidas:**
  Do inglês, *Hard Constraints*, são restrições que obrigatoriamente devem ser satisfeitas;
- **Restrições Flexíveis:**
  Também do inglês, *Soft Constraints*, conjunto de restrições que devem ser satisfeitas se possível, mas para as quais sabemos que nem todas poderão ser atendidas;

Diante da dificuldade em se cumprir todas as exigências das restrições fracas, utiliza-se para isso variáveis de folga que medem o número de violações destas restrições. A qualidade final da solução estará diretamente vinculada a estas violações.

B. Instâncias

Visando a possibilidade de comparação de resultados, serão utilizadas as instâncias e restrições disponibilizadas pela 1ª Competição Internacional de Escalonamento de Enfermeiras, pois esta competição tem como objetivo justamente incentivar a pesquisa nesta área além de estabelecer um problema *benchmark*.

Todas estas instâncias estão disponibilizadas no site da competição, onde cada uma possui os seguintes atributos:

- ***SchedulingPeriod:*** determinas os dias abrangidos pelo planejamento de horários;
- ***StartDate:*** data inicial do período do planejamento;
- ***EndDate:*** data final do período do planejamento;
- ***Skills:*** as habilidades/capacitações que as enfermeiras podem ter;
- ***ShiftTypes:*** para cada tipo de turno, existe um Id, hora inicial, hora final, descrição e habilidades necessárias;
- ***Patterns:*** os vários tipos de padrões de sequência de turnos existentes na instância;
- ***Contracts:*** representa as regulamentações trabalhistas dos empregados. Nesta seção estão localizadas a maioria das restrições descritas anteriormente;
- ***Employees:*** uma lista dos enfermeiras que estão disponíveis. Cada enfermeira possui um único Id e trabalha de acordo com o Contrato;
- ***CoverRequirements:*** O número de enfermeiras necessários para "cobrir" determinado turno de trabalho;
- ***DayOffRequests:*** para cada requisição de folga em um dia, é dado o Id do empregado e a data;
- ***DayOnRequest:*** assim como *DayOffRequests*, para cada requisição para trabalhar, é dado o Id do empregado e a data;
- ***ShiftOffRequests:*** para cada requisição para não trabalhar determinado turno, é necessário o Id do empregado, a data e o turno não desejado;
- ***ShiftOnRequests:*** ao pedir para trabalhar determinado turno, é dado o Id do empregado, a data e o turno desejado.

C. Formulação utilizando Programação Inteira

1) Variáveis de decisão: Considerando os dados de entrada, o problema de se encontrar uma solução factível é formulado em termos das seguintes variáveis de decisão:

$$x_{n,s,d} = \begin{cases} 1, \text{ se a enfermeira } n \text{ foi alocada no turno } s \\ \quad do\ dia\ d; \\ 0, \text{ caso contrário;} \end{cases}$$

$$y_{n,y} = \begin{cases} 1, \text{ se a enfermeira } n \text{ trabalhou em algum dia} \\ \quad do\ final\ de\ semana\ i; \\ 0, \text{ caso contrário;} \end{cases}$$

2) Função objetivo: A função objetivo desta formulação é minimizar a ocorrência das violações das variáveis de folga de acordo com as restrições as quais estão sujeitas, como por exemplo, evitar que a enfermeira exceda o número de turnos trabalhados. Neste caso, a função objetivo é modelada como o exemplo a seguir:

Minimize:

$$\sum_{n \in N} = \overline{vna}(n)$$

onde, $\overline{vna}$ é a variável de folga para turnos excedidos pela enfermeira *n*

3) Restrições fortes: Conforme mencionado na Seção III-A, as restrições fortes [17] são aquelas que precisam ser satisfeitas a qualquer custo para serem aplicadas na prática. Para este problema, existem duas restrições fortes:

- Todos os diferentes tipos de turnos devem ser atribuídos às enfermeiras;
- Uma enfermeira pode apenas trabalhar em um turno por dia, isto é, dois turnos diferentes não podem ser atribuídos a uma enfermeira em um mesmo dia;

4) Restrições fracas: Estas restrições não são obrigatoriamente satisfeitas, porém deseja-se que seja o máximo possível. No "mundo real", uma escala que satisfaz todas as restrições rígidas e flexíveis geralmente não existe. As violações das restrições flexíveis [17] são geralmente usadas para avaliar a qualidade de uma solução. As restrições flexíveis disponíveis na competição são:

- **Máximo número de atribuições:**
  Representa o maior número de turnos que podem ser alocados para uma enfermeira. Unidades de tempo consecutivo possuem números consecutivos atribuídos.
- **Mínimo número de atribuições:**
  O menor número de turnos que podem ser alocados para uma enfermeira. Uma numeração semelhante ao máximo número de atribuições que pode ser utilizada.
- **Máximo número de dias consecutivos trabalhados:**
  É a maior sequência de dias consecutivos os quais uma enfermeira é escalada para trabalhar. Cada campo do mesmo dia possui o mesmo valor. O primeiro dia recebe o valor zero e os dias consecutivos recebem números consecutivos.
- **Mínimo número de dias consecutivos trabalhados:**
  Semelhantemente ao item anterior, é a menor sequência de dias consecutivos os quais uma enfermeira é escalada para trabalhar. Os campos também possuem o mesmo número, o primeiro dia também recebe zero e os dias seguintes recebem números consecutivos.
- **Máximo número de dias consecutivos não trabalhados:**
  É a maior sequência de dias consecutivos os quais uma enfermeira não é escalada para não trabalhar.
- **Mínimo número de dias consecutivos não trabalhados:**
  Ao contrário do anterior, esta é a menor sequência de dias consecutivos em que uma enfermeira não é escalada para trabalhar.
- **Máximo número de finais de semana consecutivos trabalhados:**
  É o número máximo de fins de semana consecutivos atribuídos para cada enfermeira. Se a enfermeira trabalhar apenas em um dos dois dias do fim de semana é considerado fim de semana completo trabalhado. Todos os campos nos dias do mesmo final de semana recebem o mesmo valor. Finais de semana consecutivos recebem números consecutivos.
- **Finais de semana completos:**
  Tem como valor verdadeiro se uma enfermeira tem que trabalhar todos os dias nos fins de semana trabalhados. Os campos no mesmo dia recebem o mesmo valor, os dias consecutivos no mesmo final de semana possuem valores consecutivos. Deve haver uma diferença de no mínimo dois entre o valor de um final de semana e o ultimo dia do final de semana anterior. Uma penalidade alta e atribuída se os dias trabalhados no final de semana não são consecutivos.
- **Finais de semana trabalhados idênticos:**
  Verdadeiro se uma enfermeira trabalha em turnos idênticos em todos os fins de semana trabalhados. Os campos para o mesmo turno em todos os dias do final de semana são atribuídos com o mesmo valor. Turnos diferentes são atribuídos com diferentes números.
- **Uma simples atribuição por dia:**
  Uma enfermeira somente pode ser escalada para trabalhar apenas um turno por dia. Os campos de tempo recebem os mesmos valores. Os campos dos dias diferentes recebem valores diferentes.
- **Dois dias livres após um turno noturno:**
  O campo representante do primeiro turno noturno recebe um valor. Já todos os outros campos, exceto este do turno noturno, recebem valores sequenciais. Os campos restantes são atribuídos com valores que representam indefinição.
- **Requerimento de dia trabalhado/não trabalhado:**
  Todos os campos que representam solicitações de dias de folga recebem o mesmo valor. Os outros dias recebem valores diferentes e os campos restantes são atribuídos com valores que representam indefinição. A numeração referente a solicitação de dias de folga é semelhante a solicitação de dias para serem trabalhados.
- **Requerimento de turno trabalhado/não trabalhado:**
  O campo que representa a solicitação de um turno não trabalhado recebe um valor. Campos de diferentes requisições de turnos não trabalhados recebem valores diferentes. Todos os outros campos restantes são atribuídos com valores que representam indefinição. A numeração é completamente semelhante a requisição de turnos para serem trabalhados.

- **Habilidades alternativas:**
  Para exercer certo cargo ou até mesmo requisitos, alguns horários precisam de enfermeiras que possuam habilidades especificas. Os campos pertencentes as enfermeiras que não possuem as habilidades necessárias recebem um valor. Diferentes campos recebem valores diferentes e todos os campos restantes são declarados indeterminados.
- **Sequência de turnos indesejáveis:**
  Turnos indesejáveis são uma sequencia de atribuições que uma enfermeira não deseja trabalhar. Existem diferenças entre sequências que são indesejadas em dias específicos (como por exemplo: uma enfermeira geralmente não gosta de trabalhar a noite precedendo um feriado ou final de semana e geralmente gostam de trabalhar nas sextas-feiras antecedentes à finais de semana trabalhados) e sequências que são indesejadas durante o período de planejamento em geral (exemplo: uma enfermeira não gosta de trabalhar durante a noite antes de um turno pela manhã).

Um modelo de implementação de restrição fraca, pode ser escrito da seguinte maneira:

- Número máximo de alocação da enfermeira *n* na semana *j*:

$$\sum_{s \in NS(n), d \in W_j} x_{n,s,d} \leq \overline{na}(C(n)) + \overline{vna}(n,j)$$

$$\forall n \in N, j \in \{1..nW\}$$

IV. Análise de Complexidade

O método *Branch-and-Bound* contém em sua estrutura de dados uma árvore binária. Árvore binária é um conjunto finito de elementos que está vazio ou é particionado em três subconjuntos disjuntos. O primeiro subconjunto contém um único elemento chamado de pai (problema inicial) e os outros dois subconjuntos chamados de filhos.

Esta árvore vai se ramificando até chegar ao número de variáveis existentes do problema. Para um problema com n variáveis, há 2^n soluções a serem consideradas. Portanto cada vez que n for incrementado em 1 o número de soluções dobra, ou seja, se ela contiver n nós no nível k, ela conterá $2n$ nós no nível $k+1$. O total de nós da árvore completa será $\sum_{j=0}^{d} 2^n$. Esta complexidade é formalizada como crescimento exponencial de ordem O(2^k).

Apesar da complexidade deste método ser exponencial e teoricamente difícil de ser resolvido para instâncias grandes, percebe-se que na prática os resultados são satisfatórios visto que a poda da árvore restringe o universo de ramos a serem percorridos.

V. Experimentos

Para avaliação dos métodos propostos, experimentos foram realizados. Inicialmente, as instâncias foram interpretadas e modeladas no formato *mathprog* que é uma linguagem de modelagem livre e destinada a descrever modelagens de programação linear, reconhecida por vários resolvedores como por exemplo o GLPK, que é distribuído também sob licença livre.

Tabela I
Características das Instâncias

Instâncias	$\|Contr.\|$	$\|Enf.\|$	$\|Turnos\|$	$\|Espec.\|$	$\|Dias\|$
sprint_late01	3	49	5	2	35
sprint_late02	3	49	5	2	35
sprint_late06	3	50	5	2	35
sprint_late10	3	50	5	2	35

Uma vez no formato GLPK, este *software* permite a exportação para diversos formatos, incluindo o formato LP, aceito pela maior parte dos resolvedores, incluindo o CPLEX e o COIN-CBC.

Utilizou-se o resolvedor linear GLPK, versão 4.44, o qual foi compilado com o compilador GCC, versão 4.4.1, com a opção -O3 para gerar o arquivo de entrada para o CPLEX, visto que este resolvedor não interpreta diretamente a linguagem mathprog. A escolha por este resolvedor se justifica pela robustez e melhores resultados alcançados por este *software*.

Os testes foram executados em um computador com um processador Intel Core I5 3.2GHz, 16Gb de memória RAM, rodando o sistema operacional Linux Ubuntu Server 10.10.

A. Instâncias utilizadas

Foram consideradas quatro instâncias da competição para validação do modelo apresentado. Estas instâncias foram escolhidas de acordo com uma das etapas da competição (*sprint_late*), pois nesta etapa, os dados são fornecidos com maior antecedência para preparação dos modelos. Algumas das características das instâncias escolhidas estão apresentadas na Tabela I, onde:

- *Contr.*: número de tipos de contratos;
- *Enf.*: quantidade de enfermeiras;
- *Turnos*: quantidade de tipos de turnos;
- *Espec.*: número de especialidades de profissionais de enfermagem;
- *Dias*: número de dias do horizonte de planejmaneto.

B. Execução dos experimentos

Os experimentos foram executados em um primeiro momento sem limite de tempo, visto que o limitador para a execução tem sido o grande volume de memória utilizado. Mesmo ultrapassando os limites da memória do computador, já foi possível avaliar alguns resultados. Através de um *software* disponibilizado pela competição, é possível converter a solução em uma formato interpretado por este validador onde é retornado o valor da penalidade da solução gerada, ou seja, o quanto as restrições fracas estão sendo violadas pelo modelo proposto. Estes resultados estão apresentados

Tabela II
RESULTADOS

Instâncias	Penalidade	Competição
sprint_late01	228	37
sprint_late02	139	42
sprint_late06	348	42
sprint_late10	383	43

na Tabela II bem como os melhores resultados alcançados na competição até o momento.

VI. CONCLUSÃO

Neste trabalho foi apresentado um algoritmo que tem como propósito resolver genericamente o problema de alocação de enfermeiras utilizando a técnica de *Branch-and-Bound* e Programação Inteira. Para isto, foi utilizado como referência a 1ª Competição de Escalonamento de Enfermeiras, onde as instâncias foram disponibilizadas.

Os resultados obtidos neste trabalho demonstram que a utilização de métodos exatos para a solução deste tipo de problema são promissores e podem resultar em soluções cada vez melhores na prática.

Algumas propostas de trabalhos futuros podem ser apresentadas, como por exemplo a utilização do método de geração de colunas, utilização de busca local ou mesmo heurística para geração de soluções iniciais para os métodos exatos.

REFERÊNCIAS

[1] T. Jensen and B. Toft, *Graph Coloring Problems*. Wiley, 1995.

[2] S. Haspeslagh, P. D. Causmaecker, M. Stolevik, and A. Schaerf, *First International Nurse Rostering Competition 2010*, CODeS, Department of Computer Science, KULeuven Campus Kortrijk, Belgium; SINTEF ICT, Department of Applied Mathematics, Norway; DIEGM, University of Udine, Italy, May 5 2010.

[3] E. K. Burke and T. Curtois, "An ejection chain method and a branch and price algorithm applied to the instances of the first international nurse rostering competition, 2010," in *The First International Nurse Rostering Competition (INRC 2010)*, 2010.

[4] E. Bulog, "A generic nurse rostering algorithm for the inrc2010 instances," in *The 45th Annual Conference of the ORSNZ*, 2010.

[5] Z. Lü and J. Hao, "Adaptive local search for the rst international nurse rostering competition," in *The First International Nurse Rostering Competition (INRC 2010)*, 2010.

[6] K. Nonobe, "Inrc2010: An approach using a general constraint optimization solver," in *The First International Nurse Rostering Competition (INRC 2010)*, 2010.

[7] T. Dias, D. Ferber, C. d. Souza, and A. Moura, "Constructing nurse schedules at large hospitals," *International Transactions in Operational Research*, vol. 10, pp. 245–265, 2003.

[8] Y. Pochet and L. A. Wolsey, *Production Planning by Mixed Integer Programming*. Springer, 2006.

[9] J. Xu, Y. Xu, D. Kim, and M. Li, "Raptor: Optimal protein threading by linear programming," *Journal of Bioinformatics and Computational Biology*, vol. 1, pp. 95–117, 2003.

[10] H. G. Santos, "Formulações e algoritmos para o problema de programação de horários em escolas," Ph.D. dissertation, Instituto de Computação - Universidade Federal Fluminense, 2007, orientadores: Luiz Satoru Ochi and Eduardo Uchoa.

[11] E. N. Johnson, G., and W. Savelsbergh, "Progress in linear programming-based algorithms for integer programming: An exposition," *INFORMS Journal on Computing*, vol. 12, pp. 2–23, 2000.

[12] K. Aardal and S. van Hoesel, "Polyhedral techniques in combinatorial optimization I: Theory," *Statistica Neerlandica*, vol. 50, pp. 3–26, March 1996.

[13] A. Orman and H. Williams, *Optimisation, Econometric and Financial Analysis*. Springer Berlin Heidelberg, 2007, ch. A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem, pp. 91–104.

[14] G. Pataki, "Teaching integer programming formulations using the traveling salesman problem," *SIAM Review*, vol. 45, pp. 116–123, 2003.

[15] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling Computer and Manufacturing Process*, 2, Ed. Springer-Verlag, 1996.

[16] B. Render and R. M. Stair, *Quantitative Analysis for Management*, 6, Ed. New Jersey: Prentice-Hall Inc, 1997.

[17] D. Causmaecker and G. Vanden Berghe, "Relaxation of coverage constraints in hospital personnel rostering," in *Practice and Theory of Automated Timetabling (PATAT), Fourth International Conference*, L. N. in Computer Science, Ed., vol. 2740, 2003, pp. 129–147.