UNIVERSIDADE FEDERAL DE OURO PRETO

Técnicas de Programação Inteira para o Problema de Escalonamento de Enfermeiras

Rafael Antônio Marques Gomes
Universidade Federal de Ouro Preto

Orientador: Haroldo Gambini Santos

Co-orientador: Túlio Ângelo Machado Toffolo

Dissertação submetida ao Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto para obtenção do título de Mestre em Ciência da Computação

Ouro Preto, Dezembro de 2012

Técnicas de Programação Inteira para o Problema de Escalonamento de Enfermeiras

Rafael Antônio Marques Gomes
Universidade Federal de Ouro Preto

Orientador: Haroldo Gambini Santos

Co-orientador: Túlio Ângelo Machado Toffolo

Dedico este trabalho a minha filha Lígia*, minha esposa* Raíssa*, aos meus pais* Toninho *e* Terezinha*, meus irmãos* Ederson *e* Marcos*. Todas pessoas de suma importância na minha vida.*

Técnicas de Programação Inteira para o Problema de Escalonamento de Enfermeiras

Resumo

Esta dissertação apresenta técnicas de Programação Inteira (PI) para o problema da Competição Internacional de Escalonamento de Enfermeiras (INRC). A partir de uma formulação compacta e monolítica onde a atual geração dos resolvedores executam de maneira não satisfatória, melhores estratégias de geração de cortes e heurísticas primais são propostas e avaliadas. Um grande número de experimentos computacionais com estas técnicas produziram os seguintes resultados: a otimalidade da grande maioria das instâncias foi provada, as melhores soluções conhecidas foram melhoradas em até 15% e fortes limitantes duais foram obtidos. No espírito da reprodução científica, todo o código foi implementado utilizando a Infra-Estrutura Computacional para Pesquisa Operacional (COIN-OR).

Integer Programming Techniques for the Nurse Rostering Problem

Abstract

This dissertation presents Integer Programming (PI) techniques to tackle the problem of the International Nurse Rostering Competition (INRC). Starting from a compact and monolithic formulation on which the current generation of solvers performs poorly, improved cut generation strategies and primal heuristics are proposed and evaluated. A large number of computational experiments with these techniques produced the following results: the optimality of the vast majority of instances was proved, the best known solutions were improved up to 15% and strong dual bounds were obtained. In the spirit of reproducible science, all code was implemented using the COmputational INfrastructure for Operations Research (COIN-OR).

Declaração

Esta dissertação é resultado de meu próprio trabalho, exceto onde referência explícita é feita ao trabalho de outros, e não foi submetida para outra qualificação nesta nem em outra universidade.

Rafael Antônio Marques Gomes

Agradecimentos

Agradeço a todos que me ajudam direta ou indiretamente neste trabalho.

Agradeço a *minha esposa e minha filha*, pelo paciência e carinho.

Agradeço a *meus pais e irmãos*, pelo incentivo.

Agradeço aos *meus familiares*, pelo torcida.

Agradeço ao *Haroldo*, meu orientador.

Agradeço ao *Túlio Toffolo*, meu co-orientador.

Agradeço aos *professores*, pela preocupação.

Agradeço aos *amigos*, pelos conselhos.

Agradeço aos *esquecidos* não mencionados explicitamente aqui.

Agradeço ao *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq).

Muito Obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

"Computer science is no more about computers than astronomy is about telescopes."
— Edsger Dijkstra

Nomenclatura

COIN-OR	*COmputational INfrastructure for Operations Research*
COP	*Constraints Optimization Problem* (Problema de Otimização por Restrições)
GUB	(*Generalized Upper Bound*)
HPM	Heurística de Programação Matemática
INRC	Competição Internacional de Escalonamento de Enfermeiras
LI	Limite Inferior
LS	Limite Superior
LSA	Limite Superior Anterior
MIP	*Mixed Integer Program* (Problema de Programação Inteira Mista)
NRP	*Nurse Rostering Problem* (Problema de Escalonamento de Enfermeiras)
PI	Programação Inteira
SPP	*Set Packing Polytope* (Empacotamento de Nós)
VNS	*Variable Neighborhood Search* (Busca em Vizinhança Variável)

Capítulo 1

Introdução

Existem diversas pesquisas voltadas para soluções computacionais para o problema de Escalonamento de Enfermeiras (Burke, De Causmaecker, Berghe and Landeghem 2004). Muitos trabalhos anteriores se concentram em casos de estudo específicos, sendo estes muitas vezes particularidades de determinadas instituições. Como incentivo a pesquisas nesta área e também diante da dificuldade em se comparar diferentes estratégias, recentemente foi criada uma competição Internacional de Escalonamento de Enfermeiras: a *International Nurse Rostering Competition (INRC)* (Haspeslagh, De Causmaecker, Stolevik and A. 2010). Para este problema um significante número de diferentes algoritmos foram apresentados para um conjunto de diversas instâncias disponibilizadas por esta competição. É importante ressaltar que os resultados destas instâncias são constantemente atualizados desde então com a melhor solução conhecida.

Neste trabalho são apresentadas formulações e técnicas de Programação Inteira para o Problema de Escalonamento das Enfermeiras (NRP - *Nurse Rostering Problem*) tratado na INRC.

As técnicas propostas podem ser divididas em dois grupos: o primeiro grupo é voltado à melhoria de limites duais. Neste caso não se interessa somente na rápida geração de soluções de alta qualidade, mas também há o interesse em uma estimativa precisa de um limite inferior de custo igual ou próximo à solução ótima. Isto obviamente resulta em um tempo de processamento computacional elevado, mas é um passo crítico para métodos que buscam provar a otimalidade. No segundo grupo são apresentadas técnicas para acelerar a produção de soluções viáveis próximas da otimalidade. Estas técnicas por último mencionadas podem ser utilizadas individualmente por aqueles interessados

no uso desta formulação em aplicaçõees reais.

Os experimentos computacionais mostraram que as técnicas propostas apresentam-se como métodos rápidos na produção de bons limites primais e duais. Os algoritmos propostos, além de serem heurísticas competitivas, permitiram que se provasse a otimalidade para a grande maioria das instâncias da INRC. Para a maioria das instâncias em que otimalidade não foi provada, o algoritmo obteve soluções até 15% melhores do que a melhor conhecida até então.

No espírito de desenvolvimento aberto, de modo a facilitar a reprodutibilidade dos resultados, a implementação dos procedimentos de geração de cortes foi feito utilizando o *software* de *branch-and-cut* (Forrest and Lougee-Heimer 2005) de código fonte aberto da COIN-OR Foundation, CBC (Lougee-Heimer 2003). Como adição ao código do CBC, foram propostas e implementadas rotinas alternativas de separação de cortes de Clique e Ciclo Ímpar. O impacto da inclusão dessas desigualdades foi avaliado experimentalmente e verificou-se que elas superam significativamente as rotinas originais considerando o tempo necessário para produzir melhores limites duais para as instâncias da INRC. Estas rotinas também estão sendo disponibilizadas como um projeto de código fonte aberto.

1.1 Revisão Bibliográfica

Neste trabalho, a abordagem proposta baseou-se no problema definido pela *International Nurse Rostering Competition* (INRC), patrocinada pela Conferência Internacional de Práticas e Teorias em Automatização de Quadro de Horários, PATAT. Competidores estavam habilitados a submeter técnicas específicas para cada tipo de instância. Dentre as abordagens apresentadas na competição, discutiremos as propostas finalistas. Mais detalhes sobre as abordagens podem ser consultadas no site da INRC[1].

Valouxis et al. (Valouxis, Gogos, Goulas, Alefragis and Housos 2012), vencedores do desafio, desenvolveram uma estratégia de duas fases para produzir soluções factíveis e de alta qualidade. Na primeira fase, o problema de alocações de dias é resolvido, através da definição de dias de folga e trabalho para todas as enfermeiras dentro do horizonte de planejamento. Após esta etapa, cada enfermeira tem uma escala de alocações única que deve ser seguida, sem a definição de turnos. Em seguida, um problema de Programação

[1] https://www.kuleuven-kulak.be/nrpcompetition/competitor-ranking

Inteira é definido e resolvido para a primeira semana (sete dias consecutivos) do horizonte de planejamento. A formulação de Programação inteira garante a não violação das restrições fortes e por esta razão o número total de enfermeiras deve ser igual à demanda de trabalho exigida. Nesta fase, as únicas restrições fracas consideradas são aquelas que não dependem de alocações para tipos de turnos específicos, visto que nesta fase os turnos não são considerados. Estas restrições são utilizadas no custo da função. Todas combinações possíveis de trabalho-folga para o período de uma semana são consideradas para todas as enfermeiras, resultando em 128 permutações dentro da semana. Cada padrão de trabalho-folga é representado por uma sequência binária de 7 bits, o que permitiu uma maneira eficiente de armazenamento e processamento.

Neste sentido, uma solução factível para todas as enfermeiras é produzida e esta cobre a demanda diária, além de satisfazer diversas restrições fracas. Problemas similares são resolvidos para todas as semanas consecutivas, até que todo o horizonte de planejamento seja coberto sem possibilidade de novas melhorias. A escolha de 7 dias como o número de dias na estrutura proposta foi baseada em diversos experimentos, onde constatou-se que, para um número menor de dias, o problema era resolvido na otimalidade de maneira extremamente rápida. No entanto, o espaço de busca se tornava pequeno a ponto de não contribuir com nenhuma melhoria no problema original. Por outro lado, expandir o período para mais do que 7 dias gerava problemas de Programação Inteira impossíveis de serem resolvidos no espaço de tempo disponibilizado.

Adicionalmente, três heurísticas de busca local foram utilizadas para melhorar a solução corrente através de movimentos que exploram várias regiões distintas do espaço de busca do problema. A idéia por trás dos processos pode ser percebida como movimentos complexos no espaço de soluções, considerando vários movimentos simples de uma enfermeira individualmente. A idéia básica é recombinar agendamentos parciais para criar novas escalas, mantendo as exigências de demandas diárias. Para cada movimento, a recombinação ótima é encontrada através da utilização de uma formulação de PI (Programação Inteira). Movimentos de melhoria são sempre aceitos, de modo que o processo pode ser visto como uma heurística de subida randômica. Os três processos, que mantêm sempre a viabilidade da solução, são: "corte de um dia e troca", "corte de dois dias e troca" e "2-*opt*".

Na segunda fase, utiliza-se a solução da fase anterior, na qual já se definiu quais enfermeiras devem trabalhar em cada dia. Um problema de Programação Inteira é então formulado para alocar um tipo específico de turno, de acordo com a demanda do dia. As restrições do modelo PI asseguram que todas as enfermeiras disponíveis sejam

alocadas aos tipos de turnos adequados. O modelo PI desta fase considera apenas as restrições fracas que envolvem alocações de turnos e estas restrições, conforme já dito anteriormente, são utilizadas no custo da função. Problemas similares são resolvidos para todos os dias, até que todo o horizonte de planejamento seja coberto por completo. Ressalta-se que nesta fase o número de restrições é significativamente menor do que na fase anterior.

A sequência das duas fases é executada diversas vezes, iniciando cada vez de uma escala de agendamento diferente, gerada de maneira aleatória até que o tempo de execução permitido seja alcançado. Esta sequência se mostrou eficaz ao longo de uma ampla variedade de instâncias do problema. Apesar de não terem provado a otimalidade das instâncias, Valouxis et al. (Valouxis, Gogos, Goulas, Alefragis and Housos 2012) concluiram que provavelmente poderiam ter obtido a solução ótima para várias instâncias, visto que outras abordagens na competição chegaram aos mesmos melhores custos apresentados.

Burke e Curtois (Burke and Curtois 2010) aplicaram um método de dois diferentes algoritmos às instâncias da competição. O primeiro algoritmo, busca em profundidade variável, é baseado em cadeias de ejeção (Glover 1996) e foi aplicado nas instâncias da categoria *sprint*, que tinham o tempo de execução máximo limitado a 10 segundos, de acordo com as regras da competição. O segundo método, um algoritmo *branch-and-price* (Barnhart, Johnson, Nemhauser, Savelsbergh and Vance 1998), foi aplicado às instâncias das categorias *medium* e *long*, que tinham tempo de execução máximo limitado a 10 minutos e 10 horas, respectivamente. As instâncias do problema foram convertidas para o modelo de alocação do quadro de funcionários proposto e documentado pela mesma equipe. Depois, o *software Roster Booster*, incluído na abordagem (Burke and Curtois 2010), foi utilizado. O método de *branch-and-price* proposto foi capaz de resolver diversas instâncias na otimalidade dentro do tempo permitido. No entanto, instâncias ocultas foram liberadas somente durante a competição e a abordagem proposta Burke and Curtois (Burke and Curtois 2010) se deparou com um imprevisto: o dia inicial do horizonte de planejamento. Até então, todas as instâncias que já haviam sido liberadas, inclusive aquelas categorizadas como *hint* das possíveis instâncias ocultas, tinham como primeiro dia a data de 01 de Janeiro de 2010, sexta-feira, o que sempre resultava em 4 finais de semana, fixos. Para 17 das 20 instâncias ocultas, a data base do horizonte de planejamento passou a ser o dia 01 de Junho de 2010, terça-feira, o que dependendo do número de dias do planejamento, alterava o número de finais de semana. O resultado desta mudança foi a incompatibilidade das instâncias do tipo *hidden* com a abordagem

proposta. Burke and Curtois (Burke and Curtois 2010) creditaram o resultado final adverso na competição a esta mudança de data base do horizonte de planejamento.

Outro trabalho recente desenvolvido por Burke et al. (Burke, Li and Qu 2010) é um modelo multi-objetivo hírido aplicado à instâncias do mundo real de um hospital. Este método combina Programação Inteira Mista (MIP) com a heurística *Variable Neighborhood Search* (VNS - Busca em Vizinhança Variável) (Mladenovic and Hansen 1997) para tratar o NRP. Uma formulação MIP é usada primeiro para resolver o subproblema que abrange todo o conjunto de restrições fortes e um subconjunto de restrições fracas. Logo após, um VNS básico continua com o papel de procedimento de pós-processamento para melhorar as soluções resultantes do MIP. O maior foco do VNS é satisfazer as restrições que não estavam incluídas no procedimento antecessor, ou seja, no MIP. Esta abordagem buscou apresentar a importância da integração das duas metodologias utilizadas: algoritmos exatos e metaheurísticas.

Bilgin et al. (Bilgin, Demeester, Mısır, Vancroonenburg, Berghe and Wauters 2010) usaram uma hiper-heurística combinada com uma heurística gulosa aleatória. Esta hiper-heurística consiste do método heurístico de seleção que assume aleatoriamente uma heurística de nível mais baixo a partir de uma lista e também um simples critério de aceitação de movimento, aplicando um *Simulated Annealing*. Esta hiper-heurística aplicada a heurística gulosa aleatória, tenta melhorar a solução explorando de maneira gulosa as trocas de partes das escalas das enfermeiras. Enquanto o algoritmo original considera estritamente as melhores trocas, a abordagem apresentada pertuba a solução depois de um certo número de iterações sem melhora. A pertubação envolve uma troca da pior escala de uma determinada enfermeira com uma escala de uma outra enfermeira selecionada aleatoriamente. Para as instância *sprint*, *medium hint* e *medium late*, a abordagem híbrida supera claramente o modelo de Programação Linear Inteira (MIP) no tempo dado pela competição. Para a maioria dos instâncias da categoria *long*, por outro lado, a implementação de MIP encontra soluções melhores do que a abordagem híbrida.

Nonobe (Nonobe 2010) usou metaheurísticas baseada em um algoritmo para o problema de otimização por restrições (COP), onde o mesmo resolvedor é utilizado para as três etapas da competição. Segundo o autor, considerando que muitos problemas de otimização são formulados como COPs, um bom algoritmo de COP pode ser considerado como uma proposta de uso geral para estes problemas, ou seja, dada uma série de restrições e seus pesos, COP procura encontrar alocações de valores para as variáveis do problema tais que a soma ponderada das penalidades das restrições violadas seja

minimizada. O resolvedor utilizado na competição adota um algoritmo de busca tabu que modifica a solução por uma operação de troca de valores das variáveis por outros. Embora a operação de movimento seja muito simples, torna possível um cálculo rápido de cada iteração. Para melhorar o desempenho, o resolvedor emprega mecanismos para controlar dinamicamente a validade das proibições e os pesos das restrições usadas para avaliar as soluções durante a busca na vizinhança. A principal idéia se baseia em gastar menos tempo no desenvolvimento de algoritmos, já que a reformulação do problema como um problema de satisfação de restrições exige que apenas as restrições definidas pelos usuários venham a ser implementadas.

1.2 O Problema de escalonamento de enfermeiras

O Problema de escalonamento de enfermeiras pode ser descrito por uma visão enfermeira-dia, enfermeira-tarefa ou até mesmo uma visão do padrão enfermeira-turno (Cheang, Li, Lim and Rodrigues 2003). Na visão enfermeira-dia, as alocações são indexadas para cada enfermeira e para cada dia. Desta maneira, a solução pode ser diretamente representada por uma matriz onde cada célula $m_{i,j}$ contém um conjunto de turnos para serem executados pela enfermeira i no dia j. De forma abrangente, pode-se dizer que este conjunto pode ter qualquer número de turnos, mas no caso da INRC, assim como na maioria dos casos práticos, uma enfermeira só pode trabalhar um turno por dia. Um dia é geralmente dividido em quatro turnos: Diurno (D), Vespertino (E), Noturno (N) e Madrugada (M), além, claro, da possibilidade de Folga (-). A Tabela 1.1 seguinte apresenta parte de uma escala trabalhista a qual indica os turnos atribuídos às enfermeiras, em uma visão enfermeira-dia.

Os dados de entrada disponibilizados pela competição fornecem informações suficientes para representar um problema real. Os diversos problemas possuem um determinado número de enfermeiras, horizonte de planejamento normalmente referente a um mês de trabalho e contratos. Normalmente existem em torno de 3 a 4 contratos por problema. Cada enfermeira está vinculada a um único contrato específico e este contrato é que determina as regras de alocações a serem seguidas, portanto, as variáveis de restrições estarão sempre fazendo referência ao contrato da enfermeira.

Tabela 1.1: Exemplo de solução em uma visão enfermeira-dia

Enfermeira	Seg	Ter	Qua	Qui	Sex	Sáb	Dom
N1	D	D	D	-	-	E	E
N2	E	N	E	E	D	-	-
N3	-	E	M	-	D	-	N

1.2.1 Restrições

O problema de alocação das enfermeiras envolve a atribuição de turnos às enfermeiras levando em consideração diversas restrições. Em geral, devemos considerar dois tipos básicos de restrições:

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

Neste trabalho, considera-se as seguintes restrições fortes, previamente definidas pela INRC:

- uma enfermeira não pode trabalhar mais de um turno por dia;
- a demanda de enfermeiras para cada turno deve ser satisfeita durante todo o planejamento.

Existem também as seguintes restrições fracas:

- mínimo/máximo número de turnos atribuídos as enfermeiras;
- mínimo/máximo número de dias de folga consecutivos;
- máximo número de dias de trabalho consecutivos;
- máximo número de fins-de-semana trabalhados consecutivos;
- número de dias de folga após uma série de turnos noturnos;

- máximo número de fins-de-semana trabalhados em um período de quatro semanas;
- fim de semana completo: se uma enfermeira trabalhar somente em um dia do final de semana, a penalidade é registrada;
- turnos idênticos no fim-de-semana: caso os turnos trabalhados no fim-de-semana sejam diferentes, é caracterizada uma penalidade;
- trabalha dia sim/não: pedido das enfermeiras para trabalhar ou não em determinado dia, se a restrição não for respeitada, compromete a qualidade da solução;
- trabalha turno sim/não: semelhante à anterior, mas se trata de turnos e não de dias;
- sequência indesejada: é uma sequência de atribuições de dias ou turnos que vai contra as preferências de uma enfermeira baseado em seu contrato;
- habilidades alternativas: uma enfermeira não pode ser atribuída à um turno que demanda mais qualificação do que a mesma possui.

Dentre as restrições fracas, é importante ressaltar as definições de sequências indesejadas. Estas sequências ou padrões indesejados é uma seqüência de alocações de turnos que está vinculada a um contrato de uma determinada enfermeira. Pode-se distinguir entre os padrões indesejados como sendo do tipo:

- de turno específico;
- de dia;
- de folga em um dia (nenhum tipo de alocação de turno).

A definição de um padrão indesejado pode ocorrer em qualquer dia do horizonte de planejamento ou em um dia específico. Alguns exemplos são:

- nenhum turno ou um turno específico poderá ser trabalhado antes de uma seqüência de dias livres. Por exemplo, o turno da noite não pode ser trabalhado antes de um fim de semana livre;
- nenhum dia livre pode ocorrer antes de qualquer seqüência de dias posteriores trabalhados. Por exemplo, se uma enfermeira trabalha em um turno do final de semana, ela também deverá trabalhar na sexta-feira;

- padrões de turnos consecutivos indesejados. Por exemplo, L-E-L. São seqüência de turnos que devem ser evitadas em um determinado contrato.

1.3 Organização do Texto

O texto desta dissertação encontra-se organizado da seguinte forma:

No Capítulo 2 é apresentada uma formulação de Programação Inteira (PI) para o problema de Escalonamento de Enfermeiras.

No Capítulo 3 são apresentadas técnicas de melhoria dos limites duais.

No Capítulo 4 são discutidas as Heurísticas de Programação Matemática (HPM) implementadas, bem como suas diferentes estruturas de vizinhanças.

No Capítulo 5 estão os experimentos e os resultados obtidos com os métodos propostos neste trabalho.

No Capítulo 6, conclusões sobre o assunto são discutidas e alguns possíveis trabalhos futuros são definidos.

Capítulo 2

Uma formulação de Programação Inteira para o problema da INRC

Nesta Seção é apresentada uma formulação de Programação Inteira (PI) para o Problema de Escalonamento de Enfermeiras (NRP). Esta modelagem auxilia na descrição do problema, uma vez que propõe um modelo que contempla todas as restrições contidas nas instâncias da competição Internacional de Escalonamento de Enfermeiras (INRC).

2.1 Dados de Entrada

$\boldsymbol{N}$ conjunto de enfermeiras

$\boldsymbol{C}$ conjunto de contratos

$\boldsymbol{\tilde{c}_n}$ contrato da enfermeira n

$\boldsymbol{S}$ conjunto de turnos

$\boldsymbol{\tilde{S}}$ conjunto de turnos noturnos

$\boldsymbol{D}$ conjunto de dias com elementos sequencialmente enumerados a partir de 1

$\boldsymbol{\Pi}$ conjunto de todos os pares ordenados$(d_1, d_2) \in D \times D : d_1 \leq d_2$ representando janelas no período de planejamento

$\tilde{\boldsymbol{W}}_{\boldsymbol{c}}$ conjunto dos finais-de-semana no horizonte de planejamento de acordo com a definição de finais de semana no contrato c, com elementos enumerados de 1 até $\tilde{w}_c$

$\tilde{\boldsymbol{D}}_{\boldsymbol{ic}}$ conjunto de dias no i-ésimo fim-de-semana do contrato c

$\tilde{\boldsymbol{r}}_{\boldsymbol{ds}}$ número de enfermeiras necessárias no dia d e turno s

$\acute{\boldsymbol{P}}_{\boldsymbol{c}}$ conjunto de padrões de turnos de trabalho indesejáveis para o contrato c

$\hat{\boldsymbol{P}}_{\boldsymbol{c}}$ conjunto de padrões de dias de trabalho indesejáveis para o contrato c

A configuração das restrições fracas dependem de cada contrato, isto é, cada contrato tem um peso associado (que também pode ser nulo) na penalização da violação de cada restrição fraca. Limites informando quão apertada é uma restrição fraca dada também estão relacionadas ao contrato.

As restrições fracas foram divididas em dois grupos. No primeiro grupo, denotado aqui como Restrições Fracas de Faixa, foram incluídas as restrições que estabelecem uma faixa de valores inteiros válidos para uma variável no formato $\underline{v} \leq v \leq \overline{v}$. Valores fora desta faixa são penalizados nas variáveis de folga de acordo com a sua distância para o valor válido mais próximo.

No segundo grupo, das *Restrições Binárias Fracas*, são incluídas aquelas que possuem caráter binário: ou são satisfeitas ou não são.

Na Tabela 2.1 cada restrição fraca é associada a um índice. Estes índices serão utilizados para expressar o estado da constante que informa o limite mínimo e máximo para uma dada restrição fraca de faixa i e um contrato c, que serão denotados aqui por $\underline{\gamma}_c^i$ e $\overline{\gamma}_c^i$, respectivamente. O peso para violar o i-ésimo limite mínimo e máximo destas restrições é denotado aqui por $\underline{\omega}_c^i$ e $\overline{\omega}_c^i$, respectivamente. Para as restriçõs fracas binárias, o peso da violação é definido por ω_c^i. Finalmente, denota-se por $\underline{\alpha}_n^i$,$\overline{\alpha}_n^i$,α_n^i as variáveis de folga associadas com a violação do limite inferior/superior das restrições fracas de faixa e binárias i para uma determinada enfermeira n, respectivamente. Índices adicionais nas variáveis α_n^i podem ser utilizados, por exemplo, quando uma violação deve ser computada para uma localização específica, de modo que α_{nk}^7 é a variável de folga relacionada para a violação da restrição referente ao final de semana completo para a enfermeira n no k-ésimo final de semana.

Tabela 2.1: Índices para restrições de faixa e lógicas

	Restrições Fracas de Faixa
1	número total de alocações
2	dias trabalhando contíguos
3	fias folgando contíguos
4	número total de finais de semana trabalhando em quatro semanas
5	finais de semanas consecutivos trabalhando
6	número de dias de folga depois de um turno noturno
	Restrições fracas binárias
7	finais de semanas completos
8	não desejável turno noturno antes de um final de semana livre
9	mesmo turno nos dias do final de semana
10	habilidades alternativas
11	padrão de turnos trabalhados indesejados
12	padrão de dias indesejados trabalhados
13	turnos e dias indesejados

Algumas sequências específicas de turnos trabalhando (restrição fraca 11) podem ser indesejadas, exemplo: *Late*, *Evening*, *Late*. O conjunto destes padrões para o contrato c é especificado em $\acute{P}_c$ e cada padrão $\acute{p} \in \acute{P}_c$ tem um tamanho $\tilde{s}(\acute{p})$ e contém $\acute{p}[1], \ldots, \acute{p}[\tilde{s}(\acute{p})] \in S$. Padrões relacionados a dias também são considerados na restrição fraca 12: sequências de dias trabalhando/folgando devem ser evitados. Por exemplo: não trabalhar na sexta-feira e trabalhar no final de semana seguinte. O conjunto destes padrões é definido por $\hat{P}_c$ com elementos $\hat{p} \in \hat{P}_c$ de tamanho $\tilde{s}(\hat{p})$. Para especificar a partir do padrão quais dias representam a opção "não trabalhando", foi definido um conjunto virtual de dias $\dot{D}$ com numeração negativa representando esta opção, de modo que os elementos do padrão $\hat{p}[1], \ldots, \hat{p}[\tilde{s}(\hat{p})]$ são restritos para estar em $D \cup \dot{D}$.

2.2 Variáveis de decisão

As principais variáveis de decisão são indexadas por três índices binários x_{nsd}:

$$x_{nsd} = \begin{cases} 1 & \text{se a enfermeira } n \text{ tem uma alocação no turno } s \text{ e dia } d \\ 0 & \text{caso contrário} \end{cases}$$

Adicionalmente, foram utilizadas quatro variáveis auxiliares:

$$y_{ni} = \begin{cases} 1 & \text{se a enfermeira } n \text{ trabalha no final de semana } i \\ 0 & \text{caso contrário} \end{cases}$$

$$w_{nd_1d_2} = \begin{cases} 1 & \text{se a enfermeira } n \text{ trabalha do dia } d_1 \text{ até o dia } d_2 \\ 0 & \text{caso contrário} \end{cases}$$

$$r_{nd_1d_2} = \begin{cases} 1 & \text{se a enfermeira } n \text{ tem folga do dia } d_1 \text{ até o dia } d_2 \\ 0 & \text{caso contrário} \end{cases}$$

$$z_{ni_1i_2} = \begin{cases} 1 & \text{se a enfermeira } n \text{ trabalha do final de semana } i_1 \text{ até o final de semana } i_2 \\ 0 & \text{caso contrário} \end{cases}$$

Para simplificar a declaração das restrições, considerou-se a variável adicional y_{n0}, que é sempre fixada em zero.

2.3 Função objetivo

Antes de apresentar a função objetivo, observou-se que algumas variáveis de folga (e suas respectivas restrições) não necessitam serem inclusas explicitamente. Este é o caso das restrições referentes a seleção de uma janela de trabalho/folga de um conjunto Π através da ativação das variáveis $w_{nd_1d_2}$ and $r_{nd_1d_2}$, respectivamente. Obviamente este é

o caso das restrições fracas 2 e 3, (Tabela 2.1) e também o caso para a restrição fraca 7, uma vez que cada ativação de $w_{nd_1d_2}$ finalizando/iniciando no meio de um final de semana deve ser penalizado. Denota-se por $\sigma_{cd_1d_2}$ e τ_{cd_2} o peso de todas as violações ocorridas a partir de um período trabalhado (ou folga) em um bloco contínuo iniciando no dia d_1 e finalizando no dia d_2 para enfermeiras do contrato c, respectivamente. O cálculo deste peso é obtido a partir de um parâmetro de entrada para violação de uma determinada restrição multiplicado pelo número de violações. As restrições fracas 10 e 13 são também diretamente penalizadas na variável x_{nsd} com coeficientes ν_{nsd}. Da mesma forma, restrição fraca 5 é penalizada na variáveis $z_{ni_1i_2}$ com coeficientes ψni_1i_2.

Minimize:

$$\sum_{n\in N}\left[\begin{array}{l}\displaystyle\sum_{(d_1d_2)\in\Pi}(\sigma_{\tilde{c}_n d_1 d_2} w_{nd_1d_2} \;+\; \tau_{\tilde{c}_n d_1 d_2} r_{nd_1d_2}) + \\ \\ \displaystyle\sum_{s\in S}\sum_{d\in D}\nu_{nsd}x_{nsd} \;+\; \overline{\omega}^1_{\tilde{c}_n}\,\overline{\alpha}^1_n \;+\; \underline{\omega}^1_{\tilde{c}_n}\,\underline{\alpha}^1_n + \\ \\ \displaystyle\sum_{i\in\{1\ldots\tilde{w}_c\}}(\omega^4_{\tilde{c}_n}\alpha^4_{ni} \;+\; \underline{\omega}^8_{\tilde{c}_n}\underline{\alpha}^8_{ni} \;+\; \underline{\omega}^9_{\tilde{c}_n}\underline{\alpha}^9_{ni}) + \\ \\ \displaystyle\sum_{i_1,i_2\in\tilde{W}_{\tilde{c}_n}:i_2\geq i_1}\psi_{ni_1i_2}z_{ni_1i_2} + \\ \\ \displaystyle\sum_{d\in D}\underline{\omega}^6_{\tilde{c}_n}\underline{\alpha}^6_n \;+\; \sum_{\acute{p}\in\acute{P}_{\tilde{c}_n}}\alpha^{11}_{n\acute{p}} \;+\; \textstyle\sum_{\hat{p}\in\hat{P}_{\tilde{c}_n}}\alpha^{12}_{n\hat{p}} \\ \\ \end{array}\right]$$

2.4 Restrições

A seguir, as restrições serão apresentadas. Restrições 2.1 e 2.2 modelam as duas restrições fortes do problema INRC: para disponibilizar cobertura suficiente de enfermeiras em todos os dias e limitar os turnos de trabalho para as enfermeiras para no máximo um por dia. Restrições 2.3 e 2.4 vinculam a ativação das variáveis x com a ativação das

variáveis y que indicam finais de semana trabalhados. Restrições 2.5 até 2.9 asseguram que cada ativação de janela de trabalho (variáveis w) é imediatamente seguido pela ativação de uma variável r com a respectiva janela de folga e vice versa. Isto implica na seleção de períodos de trabalho e de descanso contínuos de tamanhos diferentes para todo o horizonte de planejamento.

As próximas restrições são todas restrições fracas, que significa que elas podem ser violadas desde que inclua uma variável de folga (variáveis α) que serão penalizadas na função objetivo quando ativadas. Restrições 2.10 modelam o mínimo e o máximo dias trabalhados no horizonte de planejamento. Restrições 2.11 limitam o número máximo de finais de semana trabalhados em quatro semanas. Restrições 2.12 estabelecem um número mínimo de dias de folga depois de uma sequência de turnos noturnos. Restrições 2.13 garantem que uma enfermeira não está alocada para um período noturno em um dia que precede a um final de semana de folga. Para um final de semana, a alocação dos turnos deve ser igual para todos os dias trabalhados, conforme estabelecido nas restrições 2.14 e 2.15. Padrões indesejados para dias e turnos são modelados nas restrições 2.16 e 2.17.

$$\sum_{n \in N} x_{nsd} \;=\; \tilde{r}_{ds} \;\forall d \in D, s \in S \tag{2.1}$$

$$\sum_{s \in S} x_{nsd} \;\leq\; 1 \;\forall n \in N, d \in D \tag{2.2}$$

$$y_{ni} \;\geq\; \sum_{s \in S} x_{nsd} \;\forall n \in N, i \in \tilde{W}_{\tilde{c}_n}, d \in \tilde{D}_{i\tilde{c}_n} \tag{2.3}$$

$$y_{ni} \;\leq\; \sum_{s \in S, d \in \tilde{D}_{i\tilde{c}_n}} x_{nsd} \;\forall n \in N, i \in \tilde{W}_{\tilde{c}_n} \tag{2.4}$$

$$\sum_{s \in S} x_{nsd} \;=\; \sum_{(d_1,d_2) \in \Pi \,:\, d \in \{d_1,\ldots,d_2\}} w_{nd_1d_2} \;\forall n \in N, d \in D \tag{2.5}$$

$$\sum_{s \in S} x_{nsd} \;=\; 1 - \Big(\sum_{(d_1,d_2) \in \Pi \,:\, d \in \{d_1,\ldots,d_2\}} r_{nd_1d_2} \Big) \;\forall n \in N, d \in D \tag{2.6}$$

$$\sum_{(d_1,d_2) \in \Pi \,:\, d \in \{d_1 \ldots d_2\}} (w_{nd_1d_2} + r_{nd_1d_2}) \;=\; 1 \;\forall n \in N, d \in D \tag{2.7}$$

$$\sum_{d' \in \{1,\ldots,d\}} w_{nd'd} + \sum_{d'' \in D : d'' \,\geq\, d+1} w_{n,d+1,d''} \;\leq\; 1 \;\forall n \in N, d \in D \tag{2.8}$$

$$\sum_{d' \in \{1,\ldots,d\}} r_{nd'd} + \sum_{d'' \in D : d'' \geq d+1} r_{n,d+1,d''} \;\leq\; 1 \;\forall n \in N, d \in D \tag{2.9}$$

$$\underline{\gamma}^1_{\tilde{c}_n} - \underline{\alpha}^1_n \;\leq\; \sum_{s \in S, d \in D} x_{nsd} \;\leq\; \overline{\gamma}^1_{\tilde{c}_n} + \overline{\alpha}^1_n \;\forall n \in N \tag{2.10}$$

$$\sum_{i' \in \{i,\ldots,i+3\}} y_{ni'} \leq \overline{\gamma}^4_{\tilde{c}_n} + \overline{\alpha}^4_{ni} \; \forall n \in N, i \in \{1,\ldots,\tilde{w}_{\tilde{c}_n} - 3\} \tag{2.11}$$

$$\sum_{s' \in \tilde{S}} \underline{\gamma}^6_{\tilde{c}_n} x_{ns'd} + \sum_{s \in S \setminus \tilde{S}, d' \in \{d+1,\ldots,d+\underline{\gamma}^6_{\tilde{c}(n)}\}} x_{nsd'} \leq \underline{\gamma}^6_{\tilde{c}_n} + \underline{\alpha}^6_{nd} \; \forall n \in N, d \in D : d \leq |D| - \underline{\gamma}^6_{\tilde{c}(n)} \tag{2.12}$$

$$\sum_{s \in \tilde{S}} \sum_{d \in \tilde{W}_{i\tilde{c}_n} : d \geq 2 \wedge d \leq d' \forall d' \in \tilde{W}_{i\tilde{c}_n}} x_{n,s,d-1} + y_{ni} \leq 1 + \alpha^8_{ni} \; \forall n \in N, i \in \tilde{W}_{i\tilde{c}_n} \tag{2.13}$$

$$\alpha^9_n \geq x_{nsd_1} - x_{nsd_2} \; \forall n \in N, s \in S, i \in \tilde{W}_{\tilde{c}_n}, d_1, d_2 \in \tilde{D}_{i\tilde{c}(n)} : d_1 < d_2 \tag{2.14}$$

$$\alpha^9_n \geq x_{nsd_2} - x_{nsd_1} \; \forall n \in N, s \in S, i \in \tilde{W}_{\tilde{c}_n}, d_1, d_2 \in \tilde{D}_{i\tilde{c}(n)} : d_1 < d_2 \tag{2.15}$$

$$\sum_{j \in \{1,\ldots,\tilde{s}(\acute{p})\}} x_{n,\acute{p}[1],d+j-1} \leq \tilde{s}(\acute{p}) + \alpha^{11}_{n\acute{p}} \; \forall n \in N, \acute{p} \in \acute{P}_{\tilde{c}_n}, d \in \{1,\ldots,|D| - \tilde{s}(\acute{p}) + 1\} \tag{2.16}$$

$$\sum_{s \in S} \sum_{j \in \{1,\ldots,\tilde{s}(\hat{p}) : \hat{p}[j] \geq 1\}} x_{n,s,\hat{p}[j]} + \sum_{j \in \{1,\ldots,\tilde{s}(\hat{p}) : \hat{p}[j] \leq -1\}} (1 - \sum_{s \in S} x_{n,s,-\hat{p}[j]}) \leq \tilde{s}(\hat{p}) + \alpha^{12}_n \; \forall n \in N, \hat{p} \in \hat{P}_{\tilde{c}_n} \tag{2.17}$$

Capítulo 3

Melhoria do Limite Dual

O problema considerado contém muitas variáveis binárias vinculadas por várias restrições GUB (*generalized upper bound*). Restrições deste tipo definem um grafo de conflitos implícito (Atamtürk, Nemhauser and Savelsbergh 2000) indicando o conjunto de pares de variáveis cuja ativação simultânea é proibida. A relaxação da programação linear nestes problemas pode ser significativamente reforçada pela inclusão de desigualdades derivadas de um empacotamento de nós (SPP) (Padberg 1973). As classes mais comuns de cortes para SPP são os cortes de clique e de ciclos ímpares. Uma desigualdade de clique para um conjunto C de variáveis conflitantes tem a forma $\sum_{j \in C} x_j \leq 1$ e uma desigualdade de ciclo-ímpar com variáveis conflitantes C pode ser definida como: $\sum_{j \in C} x_j \leq \lfloor \frac{|C|}{2} \rfloor$. É notável que na prática os cortes de cliques são os mais importantes (Borndorfer 1998). O impacto destes cortes foi explorado em alguns problemas difíceis de quadro de horários (Avella and Vasil'ev 2005, Burke, Mareček, Parkes and Rudová 2011). Considerando as rotinas genéricas de separação de cliques, os mais comuns são os métodos de cliques de estrela e os métodos de cliques de linha (Eso 1999, Hoffman and Padberg 1993, Borndorfer 1998). Estas são rotinas de separação muito rápidas e são utilizadas no gerador de cortes da versão atual da biblioteca COIN-OR. O algoritmo proposto considera uma separação de clique agressiva: em vez de procurar para *a* desigualdade de clique mais violada, procura-se por *todas* as desigualdades de clique violadas. Alguns resultados prévios indicam que esta é a melhor estratégia. Em (Burke, Mareček, Parkes and Rudová 2011), por exemplo, embora autores utilizem um código de *branch-and-bound* para buscar o clique mais violado, resultados computacionais motivaram a inclusão de cortes violados não-ótimos encontrados durante a busca. O resultado é consistente em relatos de aplicações de outros cortes aplicados para diferentes mode-

los, tais como Chvàtal-Gomory cuts (Fischetti and Lodi 2007). A opção de inserir um grande número de desigualdades violadas ao mesmo tempo é também responsável para reforçar a importância dos cortes de Gomory (Cornuéjols 2007). A rotina de separação de clique proposta tem dois principais componentes:

1. um módulo para separar todos cliques violados no sub-grafo de conflito induzido pelas variáveis fracionárias;

2. um módulo de *lifting* que estende os cliques gerados considerando o grafo de conflitos completo;

O módulo de separação de cliques foi implementado usando uma versão melhorada do algoritmo *Bron-Kerbosch* (Bron and Kerbosch 1973). Esta versão implementa uma regra de pivoteamento otimizada (Brito and Santos 2011) para acelerar a descoberta de cliques maximais com altos pesos. Esta regra atribui prioridade maior para visitar primeiramente nós com alto grau modificado (soma do grau do nó e de seus vizinhos) e pesos. Embora este algoritmo tenha um desempenho exponencial em seu pior caso, a regra de pivoteamento heurística torna o algoritmo apropriado não somente para execução no contexto de enumeração mas também para execução com tempos restritos, já que os cliques mais violados tendem a ser descobertos primeiro. Apesar disso, constatou-se através de experimentos que todas as desigualdades de clique violadas nas instâncias deste trabalho podem ser enumeradas em fração de segundos usando esta abordagem. É importante lembrar também que, mesmo se um subconjunto de cliques for inserido, a melhor solução não seria perdida, pois o *branching* iria cuidar do restante. Esta situação não ocorre em geração de colunas: uma interrupção do algoritmo de *pricing* antes da coluna ótima ser descoberta na última iteração, tornaria impossível provar a otimalidade da solução encontrada. Em outras palavras, para algoritmos exatos a separação de cortes pode ser difícil, mas não para geração de colunas, como demonstrado em (Poggi de Aragão and Uchoa 2003). A importância da realização do *lifting* em desigualdade de clique pode ser explicada com o grafo de conflito 3.1. Nós dentro da área cinza indicam variáveis com valores não-nulos na solução fracionária. Nesta solução, somente os nós x_2,x_3,x_4 poderiam contribuir para definir a desigualdade de clique máximo violada. Apesar disso, posteriores relaxações de programação linear podem incluir três diferentes cliques violados k_3[1] alternando a variável inativa. Se a desigualdade do clique k_4 for inserida na primeira solução fracionária, re-timizações adicionais da programação linear

[1] um clique com três nós

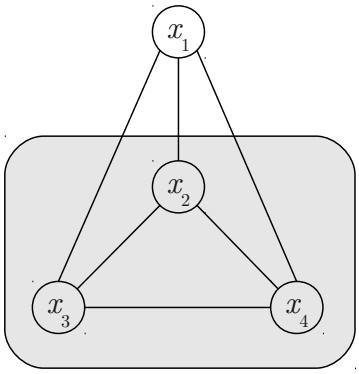

Figura 3.1: Exemplo de um k_3 que pode ser elevado para um k_4

podem ser evitadas. Além disso, uma matriz de restrições menos densa será obtida com a inserção inicial destas restrições dominantes.

É notável que a separação de ciclos-ímpares contribuem somente um pouco para melhorias do limite inferior (Borndorfer 1998, Méndez-Díaz and Zabala 2008). Apesar disso, esta inclusão no procedimento *branch-and-cut* é barato, uma vez que estas desigualdades podem ser separadas em tempo polinomial usando um algoritmo de menor caminho (Grotschel, Lovasz and Schrijver 1993). Desigualdades de ciclos-ímpares podem ser fortalecidos pela inclusão de um centro de uma roda, como a variável x_6 no grafo apresentado na Figura 3.2. Neste caso, para um ciclo ímpar com variáveis C e W sendo o conjunto de candidatos para serem incluídos no centro da roda de C, a desigualdade **??** é válida:

$$\sum_{j \in W} \lfloor \frac{|C|}{2} \rfloor x_j + \sum_{j \in C} x_j \leq \lfloor \frac{|C|}{2} \rfloor \tag{3.1}$$

O grafo de conflitos é construído através da análise da matriz de restrição. Embora a formulação apresentada seja completa para a modelagem do problema da INRC, observa-se que resolvedores podem detectar um grafo de conflito maior se as seguintes desigualdades válidas forem inseridas:

$$\sum_{d' \in \{1..d_1\}} r_{nd'd_1} + \sum_{d'' \in \{d_2..|D|\}} w_{nd_2d''} \leq 1 \ \forall n \in N, (d_1, d_2) \in \Pi \ : \ d_2 - d_1 = 2\} \tag{3.2}$$

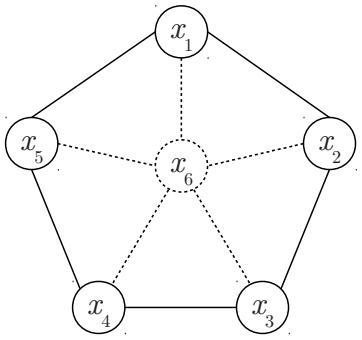

Figura 3.2: Exemplo de um ciclo ímpar e sua extensão possível para uma roda

$$\sum_{d' \in \{1,\ldots,d_1\}} w_{nd'd_1} + \sum_{d'' \in \{d_2,\ldots,|D|\}} r_{nd_2d''} \leq 1 \; \forall n \in N, (d_1, d_2) \in \Pi \; : \; d_2 - d_1 = 2\} \qquad (3.3)$$

Observa-se também que um subconjunto de variáveis está diretamente vinculado à maior parte dos custos na função objetivo: variáveis $w_{nd_1d_2}$ e $r_{nd_1d_2}$. Na solução ótima da relaxação da programação linear, estas variáveis frequentemente aparecem com valores fracionários, prejudicando a qualidade do limite dual. Como o número destas variáveis ativas para cada enfermeira é bastante limitado, optou-se por uma separação específica de corte para estas variáveis. Neste trabalho foi adotada uma rotina que separa o valor fracionário para um grupo restrito dessas variáveis para cada enfermeira através da implementação dos planos de corte de *Fenchel* (Boyd 1992),(Boyd 1994). A rotina cria problemas lineares de separação considerando vetores de incidência factíveis para as variáveis $w_{nd_1d_2}$ e $r_{nd_1d_2}$ que aparecem com valores fracionários na solução da relaxação linear, do mesmo modo como foi implementado em (Santos, Uchoa, Ochi and Maculan 2012). Estes cortes serão chamados a seguir de cortes de janela.

Capítulo 4

Heurísticas de Programação Matemática

A fase de busca local explora o espaço de busca através de diversas vizinhanças, usando um esquema inpirado em Descida em Vizinhança Variável (*Variable Neighborhood Descent - VND*) (Hansen and Mladenović 1997). Considerando os resultados obtidos com recentes utilizações de heurísticas *RINS* (Danna, Rothberg and Le Pape 2003a), foram propostas duas diferentes estruturas de vizinhança que são baseadas na resolução de partes pequenas do problema original para otimalidade. As diferenças entre as vizinhanças estão nas regras consideradas para gerar tais subproblemas.

O uso de Programação Inteira Mista (MIP) na produção de soluções de boa qualidade com alta restrição de tempo vem sendo uma crescente tendência em otimização (Martins, Souza, Souza and Toffolo 2009, Uchoa, Toffolo, de Souza, Martins and Fukasawa 2012). Um termo abrangente nesta área é a *otimização de subproblemas*. Para acelerar a melhora de soluções factíveis, resolvedores trabalham em problemas menores desempenhando a busca local. Os subproblemas podem ser definidos como uma fixação de variáveis fracas, assim como no método *Local Branching* e outros similares (Fischetti and Lodi 2003, Hansen, Mladenović and Urosević 2006), ou com fixações fortes de variáveis como em *Relaxation Induced Neighborhood Search (RINS)*(Danna, Rothberg and Le Pape 2003b). Este último trabalho mencionado apresentou melhores resultados em testes utilizando instâncias MIPLIB (Koch, Achterberg, Andersen, Bastert, Berthold, Bixby, Danna, Gamrath, Gleixner, Heinz, Lodi, Mittelmann, Ralphs, Salvagnin, Steffy and Wolter 2011).

As propostas de heuríticas MIP empregam um esquema de otimização de subproblemas de heurísticas híbridas. Regras de heurística são usadas para criar subproblemas $\mathcal{P}'(\mathcal{H})$, definidos pela fixação forte de um dado conjunto de variáveis $\mathcal{H}$ do problema original $\mathcal{P}$. Os algoritmos consistem basicamente de duas fases subsequentes: a fase de construção e a fase de busca local. A fase de construção elabora uma solução factível inicial usando uma abordagem heurística. A busca MIP é usada em todo o tempo restante para explorar grandes vizinhanças até que um mínimo local seja encontrado. O procedimento retorna a melhor solução encontrada em max_{tempo} segundos.

Observou-se, em experimentos preliminares, que a estratégia de fixação forte obteve melhor desempenho que a da fixação fraca. Tais resultados são compatíveis com os experimentos observados em (Danna, Rothberg and Le Pape 2003b). Dados estes resultados, uma atenção especial é dedicada a fixação forte das vizinhanças. As seções seguintes apresentam ambas as fixações fortes e fracas das vizinhanças.

4.1 Fixação Forte

4.1.1 Geração de Solução Inicial

A solução inicial é um procedimento simples que constrói soluções factíveis que serão utilizadas nos experimentos através de um procedimento guloso. Este método constrói uma matriz de alocação $M_{|N|\times|D|}$, inicializando todas as células m_{ij} da tabela com dias de folga (*days off*). Sequencialmente, para cada dia d e turno s, a demanda $\tilde{r}_{ds}$ é satisfeita pela seleção, uma a uma, de uma enfermeira n para que esta nova alocação desencandeie um pequeno incremento na função objetivo naquele instante considerando a solução parcial aumentada definida em $M_{|N|\times|D|}$. Este processo é repetido até que toda a demanda por enfermeiras esteja alocada. O algoritmo tem complexidade de tempo proporcional a $O(|N|^2 \times |D|)$.

4.1.2 Estruturas de vizinhança

Dada uma solução factível S_0, um vizinho é obtido basicamente da seguinte maneira: (i) define-se um conjunto de alocações de enfermeiras que serão fixadas, de acordo com a solução S_0 e (ii) resolve-se na otimalidade o subproblema obtido com as fixações. Em testes preliminares, duas das vizinhanças avaliadas apresentaram resultados muito

melhores. Os parágrafos seguintes descrevem estas duas vizinhanças.

Fixação de Dias

Na estrutura da vizinhança Fixação de Dias, os subproblemas são gerados pela fixação de todas as alocações de enfermeiras que trabalham em um determinado número de dias $|D| - ndias$ do horizonte de planejamento, onde $ndias$ é um parâmetro da vizinhança.

Na primeira iteração ($iter = 0$), um subproblema é criado a partir da solução inicial S, fixando todas as alocações feitas às enfermeiras, exceto aquelas no intervalo de dias entre 1 e $ndias$. O subproblema é então resolvido e a solução ótima é obtida. Se a solução do subproblema melhorar a solução corrente, então S é atualizada. Na próxima iteração, outro subproblema é gerado a partir da solução corrente, pela fixação de todas as alocações de enfermeiras, exceto aquelas entre os dias dia_A e dia_B (equações 4.1 e 4.2):

$$dia_A = 1 + (iter \times step) \tag{4.1}$$

$$dia_B = ndias + (iter \times step) \tag{4.2}$$

As equações 4.1 e 4.2 calculam, respectivamente, o início e o fim de uma janela na qual os dias pertencentes a ela permanecem não fixados. Nestas equações, $iter$ é o número de iterações correntes e $step$ é um parâmetro que define um número de dias entre dois subproblemas consecutivos. Se o valor de dia_A ou dia_B é maior que o número de dias, $|D|$, considera-se o dia para ser a sobra deste valor dividido por $|D|$. O algoritmo procede até $|D|/step$ iterações consecutivas sem que qualquer melhora seja alcançada, o que indica que um ótimo local para a vizinhança foi encontrado. A figura 4.1 mostra como a busca no espaço de soluções é realizada pela estrutura de vizinhança de fixação dos dias.

As vizinhanças tem dois parâmetros, $step$ e $ndias$. Como dito anteriormemte, a primeira indica o número de dias entre dois subproblemas consecutivos. Quanto menor o valor, maior é o número de diferentes subproblemas diferentes na vizinhança. O outro parâmetro, $ndias$, define o número de dias que não serão fixados nos subproblemas. Trata-se de um parâmetro crítico, portanto. Se $ndias$ assumir valores muito pequenos, os subproblemas gerados não serão capazes de melhorar a solução corrente. Por outro

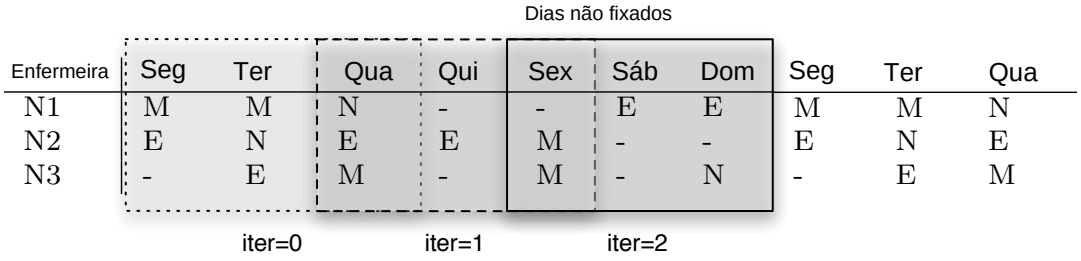

Figura 4.1: *Fixação dos Dias* de uma vizinhança com $ndias = 3$ e $step = 2$.

lado, se *ndias* assumir valores muito grandes, os subproblemas gerados podem ser difíceis demais de serem resolvidos.

Fixação de Turnos

Na estrutura de vizinhança Fixação de Turnos, os subproblemas são criados pela fixação de todas as alocações de $|S| - 1$ turnos. Na primeira iteração, apenas as alocações do primeiro turno permanecem não fixados. Na segunda iteração, somente as alocações do segundo turno não são fixadas, e assim sucessivamente. Como o número de diferentes subproblemas gerados na vizinhança é igual à $|S|$, o algoritmo procede até que $|S|$ iterações consecutivas sem melhora sejam alcançadas. Esta vizinhança não possui nenhum parâmetro.

Dado que as instâncias possuem, em média, de 3 a 5 turnos, pode parecer que os subproblemas desta vizinhança são difíceis de resolver. Na prática, no entanto, estes subproblemas puderam ser resolvidos em pouco tempo, conforme será apresentado na Seção 5.

4.1.3 Heurística MIP de Fixação Forte

Na última Sessão, foram apresentadas as estruturas das vizinhanças usadas pela heurística de programação matemática (HPM). A heurística funciona de uma forma semelhante ao método VND, buscando cada uma das vizinhanças até que seus mínimos locais sejam encontrados. Decididiu-se utilizar apenas as seguintes estruturas no algoritmo proposto:

$\boldsymbol{\mathcal{N}_1^m}$: Estrutura de vizinhança de *Fixação dos Dias* com $ndias = 2m$ e $step = m$;

$\boldsymbol{\mathcal{N}_2}$: Estrutura de vizinhança de *Fixação dos Turnos*;

Inicialmente, o algoritmo executa uma busca completa nas vizinhanças $\mathcal{N}_1^m$ e $\mathcal{N}_2$. Após esta etapa, o algoritmo aumenta o valor de m em uma unidade, procurando pela melhor solução na vizinhança $\mathcal{N}_1^{m+1}$. Se ocorrer qualquer melhora durante a última busca, o algoritmo procura na vizinhança $\mathcal{N}_2$ antes de incrementar o valor de m. Caso contrário, o algoritmo apenas incrementa o valor de m, movendo para a vizinhança $\mathcal{N}_1^{m+1}$. Este processo se repete até que $m > |D|/2$ ou até que o tempo limite seja atingido. É importante destacar que quando $m = |D|/2$, o subproblema gerado pela vizinhança $\mathcal{N}_1^{|D|/2}$ é o problema original. Nesta situação, a solução retornada é uma solução ótima do problema original, e portanto a busca é encerrada.

O algoritmo 4.1 apresenta o pseudocódigo da heurística proposta. Nesta figura, os procedimentos $\mathcal{N}_1^m(S_0)$ e $\mathcal{N}_2(S_0)$ retornam, respectivamente, os melhores vizinhos de S_0 nas vizinhanças $\mathcal{N}_1^m$ e $\mathcal{N}_2$. Se nenhuma solução melhor que S_0 é encontrada, então a própria solução S_0 é retornada.

Algoritmo 4.1: Pseudo-código do Algoritmo HPM

Entrada: S_0, m
1: $S^* \leftarrow \mathcal{N}_1^m(S_0)$
2: $S^* \leftarrow \mathcal{N}_2(S^*)$
3: $m = m + 1$
4: **enquanto** $m \leq |D|/2$ e o tempo limite não for alcançado **faça**
5: $S \leftarrow \mathcal{N}_1^m(S^*)$;
6: **se** S for uma solução melhor que S^* **então**
7: $S^* \leftarrow \mathcal{N}_2(S)$;
8: **fim se**;
9: $m = m + 1$;
10: **fim enquanto**;
11: **retorne** S^*;

4.2 Fixação Fraca

4.2.1 Local Branching

O primeiro algoritmo de fixação fraca, doravante denominado HPM-$\mathcal{LB}$ (Heurística de Programação Matemática com *Local Branching*), utiliza uma implementação da

heurística *Local Branching* (Fischetti and Lodi 2007) (Hansen, Mladenović and Urosević 2006) como busca local. Para uma dada instância do problema, existe sempre o mesmo número de variáveis x_{nsd} que recebem valores 1 e 0 em soluções factíveis, o que simplifica o cálculo da distância de uma solução de referência. Nesse sentido, apresenta-se o corte *Local Branching* modificado para uma solução de referência com variáveis $\overline{x}_{nsd}$ e tamanho k, denotado aqui por $\mathcal{LB}(\overline{x}, k)$ como:

$$\sum_{n\in N}\sum_{s\in S}\sum_{d\in D\,:\,\overline{x}_{nsd}=1} x_{nsd} \geq \sum_{n\in N}\sum_{s\in S}\sum_{d\in D} \overline{x}_{nsd} - k \tag{4.3}$$

O parâmetro k indica, portanto, o número de variáveis (binárias) que poderão ser alteradas. O pseudocódigo da heurísticas HPM-$\mathcal{LB}$ é apresentado no Algoritmo 4.2. A solução inicial, S_0, é gerada de forma análoga à apresentada na seção 4.1.1. A construção se dá por meio de uma matriz de alocação $M_{|N|\times|D|}$, com todas as células m_{ij} da tabela inicializadas com dias de folga (*days off*). Sequencialmente, para cada dia d e turno s, a demanda $\tilde{r}_{ds}$ é satisfeita pela seleção, uma a uma, de uma enfermeira n para que esta nova alocação desencandeie o menor incremento possível na função objetivo naquele instante, levando em conta a solução parcial definida em $M_{|N|\times|D|}$. Este processo é repetido até que toda a demanda por enfermeiras seja satisfeita.

Algoritmo 4.2: Pseudo-código do Algoritmo HPM-$\mathcal{LB}$

Entrada: S_0, k
1: $S \leftarrow S_0$;
2: $S^* \leftarrow \mathcal{LB}(S, k)$;
3: **enquanto** S^* for uma solução melhor que S **faça**
4: $\quad S \leftarrow S^*$;
5: $\quad S^* \leftarrow \mathcal{LB}(S, k)$;
6: **fim enquanto**;
7: **retorne** S^*;

4.2.2 Elite Set Polishing

A segunda heurística de fixação fraca, denotada aqui por HPM-$\mathcal{ESP}$ (Heurística de Programação Matemática com *Elite Set Polishing*), utiliza como busca local a heurística *Elite Set Polishing* ($\mathcal{EPS}$). Uma busca local do tipo foi apresentada em (Pigatti, Poggi

de Aragão and Uchoa 2005), podendo ser utilizada no contexto de metaheurísticas populacionais ou em qualquer outro método de pesquisa que mantenha um conjunto de soluções. A heurística está relacionada ao trabalho de (Rothberg 2007), mas em vez de realizar a fixação forte nas variáveis, calcula o peso para as mudanças delas baseadas no consenso das soluções envolvidas. Na sua forma mais simples, o $\mathcal{ESP}$ para duas soluções de referência $\overline{x}^1_{nsd}$ e $\overline{x}^2_{nsd}$ com tamanho k, $\mathcal{ESP}(\overline{x}^1, \overline{x}^2, k)$ é definido como:

$$\sum_{n\in N}\sum_{s\in S}\sum_{d\in D\,:\,\overline{x}^1_{nsd}+\overline{x}^2_{nsd}=2} 2x_{nsd} + \sum_{n\in N}\sum_{s\in S}\sum_{d\in D\,:\,\overline{x}^1_{nsd}+\overline{x}^2_{nsd}=1} x_{nsd} \geq \sum_{n\in N}\sum_{s\in S}\sum_{d\in D}\overline{x}_{nsd} - k \quad (4.4)$$

No caso da heurística de vizinhança *Elite Set Polishing* ($\mathcal{ESP}$), serão necessárias duas soluções iniciais de entrada para o algoritmo, S_0 e S_1. Para gerar duas soluções, é feita uma pequena alteração no algoritmo apresentado na seção 4.1.1. O procedimento é executado $|D|$ vezes, uma vez partindo de cada dia. A título de exemplo, na primeira iteração é executado exatamente o algoritmo apresentado na seção 4.1.1. Na segunda iteração, a construção da solução é iniciada a partir do segundo dia, seguido do terceiro, quarto, quinto e etc. Na terceira iteração, a construção é iniciada a partir do terceiro dia, e assim sucessivamente. As duas soluções eleitas serão aquelas que forem mais diferentes entre si, ou seja, aquelas que tiverem o menor número de variáveis ativas em comum.

O Algoritmo 4.3 apresenta o pseudocódigo da heurística HPM-$\mathcal{EPS}$.

Algoritmo 4.3: Pseudo-código do Algoritmo HPM-$\mathcal{ESP}$

Entrada: S_0, S_1, k

1: $S^* \leftarrow \mathcal{EPS}(S_0, S_1, k)$;
2: **enquanto** S^* for uma solução melhor que S_0 **faça**
3: $\quad S_1 \leftarrow S_0$;
4: $\quad S_0 \leftarrow S^*$;
5: $\quad S^* \leftarrow \mathcal{EPS}(S_0, S_1, k)$;
6: **fim enquanto**;
7: **retorne** S^*;

Capítulo 5

Experimentos Computacionais

A implementação dos algoritmos foi feita em C++ usando as bibliotecas de código aberto do COIN-OR. Esta abordagem pemitiu integrar a fundo as rotinas com os resolvedores MIP COIN-OR (Forrest and Lougee-Heimer 2005, Ralphs and Guzelsoy 2005, Linderoth and Ralphs 2005) e também comunicar com resolvedores comerciais através do *Open Solver Interface (OSI)* (Ralphs, Saltzman and Ladnyi 2004). Resolvedores de código fechado podem também ter códigos adicionais integrados usando *callbacks*, mas esta abordagem é em última análise limitada pelo qual os *callbacks* estão disponíveis e quantas decisões por eles são delegadas. Assim, todas as rotinas de geração de cortes foram implementadas e testadas usando o resolvedor COIN *Branch-and-Cut* (CBC) (Forrest and Lougee-Heimer 2005) que é o resolvedor MIP de código aberto mais rápido disponível (Mittelmann 2012).

O código foi compilado com o GCC/g++ versão 4.6. Todos experimentos foram executados em diversos computadores Core I7 3.4GHz com 16Gb de memória RAM rodando Linux Ubuntu 10.10 64-bits. Foi utilizado o resolvedor CPLEX versão 12.2.0 e COIN-OR CBC 2.7.6.

O conjunto de instâncias utilizadas nos experimentos foram as disponibilizadas durante a INRC, incluindo as instâncias difíceis da categoria *hidden*. As características das instâncias do problema são apresentadas nas Tabelas 5.1 e 5.2, onde a coluna "Con." indica o número de contratos, "Enf" a quantidade de enfermeiras, "Tur" o número de turnos, "Pad" o número de padrões, "Dias" a quantidade de dias do horizonte de planejamento , "T.F." a quantidade de turno requisitados para não trabalhar por contrato, "D.F" o número de dias requisitados para não trabalhar por contrato e por fim, a coluna

"D.T'.' como a quantidade de dias preferenciais para trabalhar. Mais informações sobre estas instâncias podem ser obtidas também em (Haspeslagh, De Causmaecker, Stolevik and A. 2010) ou no website[1] da competição.

Antes de apresentar a avaliação desta proposta, é importante chamar a atenção para os resultados do estado da arte de alguns experimentos com Resolvedores de Programação Inteira. O objetivo é determinar o quão poderosos esses resolvedores estão lidando com as instâncias da INRC com a formulação proposta e a aplicação de apenas algumas configurações adicionais, se existirem.

5.1 Resolvedores com parâmetros inalterados

Foram incluídos experimentos com o resolvedor comercial CPLEX (IBM 2011) com parâmetros inalterados.

Nas Tabelas 5.3 e 5.4, os resultados do CPLEX executando com diferentes ênfases de otimização com tempos de execução restritos a 10 minutos e uma hora são incluídos. Os limites inferior e superior (li/ls) são apresentados através do cálculo do gap $\frac{ls-li}{li} \times 100$.

Os resultados são resumidos por grupos de instâncias incluindo a média e desvio padrão (m/d.p) dos valores do gap. Estes valores levam em consideração somente os gaps válidos, ou seja, contabilizam apenas as instâncias para as quais o Cplex foi capaz de encontrar pelo menos uma solução primal viável.

Uma vez que soluções factíveis para as instâncias da INRC puderam ser geradas rapidamente usando uma heurística gulosa (veja Capítulo 4), e considerando que o CPLEX possa ler estas soluções iniciais geradas, foram incluídos experimentos onde o CPLEX inicia a sua execução a partir de uma solução inicial factível já produzida (colunas com $s = gr(.)$). Para as instâncias da categoria *sprint*, o CPLEX sempre encontrou a solução ótima em poucos minutos, portanto os resultados para esta categoria não serão reportados.

Os resultados das Tabela 5.3 e 5.4 indicam que embora existam instâncias maiores que são facilmente resolvidas pelo resolvedor, de forma que a otimalidade seja provada em

[1] `http://www.kuleuven-kulak.be/nrpcompetition`

Tabela 5.1: Características das Instâncias *long* e *medium*

Instância			Con.	Enf.	Tur.	Pad.	Dias	T.F.	D.F.	D.T.
long	early	01	3	49	5	2	28	3	245	490
		02	3	49	5	2	28	3	245	490
		03	3	49	5	2	28	3	245	490
		04	3	49	5	2	28	3	245	490
		05	3	49	5	2	28	3	245	490
	hidden	01	3	50	5	2	28	9	0	0
		02	3	50	5	2	28	9	0	0
		03	3	50	5	2	28	9	0	0
		04	3	50	5	2	28	9	0	0
		05	4	50	5	2	28	9	0	0
	late	01	3	50	5	2	28	9	0	0
		02	4	50	5	2	28	9	0	0
		03	3	50	5	2	28	9	0	0
		04	4	50	5	2	28	9	0	0
		05	3	50	5	2	28	9	0	0
medium	early	01	4	31	4	1	28	0	310	93
		02	4	31	4	1	28	0	310	93
		03	4	31	4	1	28	0	310	93
		04	4	31	4	1	28	0	310	93
		05	4	31	4	1	28	0	310	93
	hidden	01	4	30	5	2	28	9	0	0
		02	4	30	5	2	28	9	0	0
		03	4	30	5	2	28	9	0	0
		04	4	30	5	2	28	9	0	0
		05	4	30	5	2	28	9	0	0
	late	01	4	30	4	1	28	7	300	90
		02	3	30	4	1	28	7	300	90
		03	4	30	4	1	28	0	300	90
		04	3	30	4	1	28	7	300	90
		05	4	30	5	2	28	7	300	90

menos de 10 minutos, há vários casos em que o resolvedor (colunas $s = \emptyset$) não encontrou *nenhuma* solução factível em uma hora de processamento, mesmo com a ativação das ênfases heurísticas.

Tabela 5.2: Características das Instâncias *Sprint*

Instância			**Con.**	**Enf.**	**Tur.**	**Pad.**	**Dias**	**T.F.**	**D.F.**	**D.T.**
sprint	early	01	4	10	4	1	28	3	50	100
		02	4	10	4	1	28	3	50	100
		03	4	10	4	1	28	3	50	100
		04	4	10	4	1	28	3	50	100
		05	4	10	4	1	28	3	50	100
		06	4	10	4	1	28	3	50	100
		07	4	10	4	1	28	3	50	100
		08	4	10	4	1	28	3	50	100
		09	4	10	4	1	28	3	50	100
		10	4	10	4	1	28	3	50	100
	hidden	01	3	10	3	1	28	4	50	100
		02	3	10	3	1	28	4	50	100
		03	3	10	4	1	28	8	70	150
		04	3	10	4	1	28	8	50	100
		05	3	10	4	1	28	8	50	100
		06	3	10	3	1	28	4	100	200
		07	3	10	3	1	28	4	100	200
		08	3	10	4	1	28	8	120	250
		09	3	10	4	1	28	8	100	200
		10	3	10	4	1	28	8	100	200
	late	01	3	10	4	1	28	8	50	100
		02	3	10	3	1	28	4	39	100
		03	3	10	4	1	28	8	50	100
		04	3	10	4	1	28	8	50	100
		05	3	10	4	1	28	8	50	100
		06	3	10	4	1	28	0	50	100
		07	3	10	4	1	28	0	50	100
		08	3	10	4	1	28	0	0	0
		09	3	10	4	1	28	0	0	0
		10	3	10	4	1	28	0	50	100

Apesar de informar uma solução inicial como entrada (colunas $s = gr(.)$) resolver o problema da factibilidade, a qualidade da solução final depis de uma hora de processamento ainda continua longe do aceitável, com gaps de aproximandamente 70% em

algumas situações. Estes resultados demonstram que apesar dos avanços dos resolvedores MIP genéricos , em muitos casos de aplicações do mundo real a hibridização destes resolvedores com métodos que consideram informações específicas do problema é muito importante e em muitos casos, absolutamente necessário.

5.2 Planos de cortes

O objetivo do procedimento de separação é aumentar a velocidade das melhorias do limite inferior e consequentemente provar a otimalidade rapidamente. No primeiro experimento foram executadas várias rodadas de separação de cortes usando a proposta do procedimento de separação por cliques, nomeada aqui como `eclq` e as rotinas de separação de cliques inclusas na própria biblioteca de geração de cortes do COIN-OR, denotado como `cgl`, limitados pelo tempo de 100 segundos para instâncias da categoria *sprint* e 600 segundos para instâncias maiores. Para medir as melhorias, foi registrada para cada instância e instante de tempo a distância relativa (gap) para o melhor limite superior: a solução ótima ou a melhor conhecida. Considerando um dado limite inferior lb e um limite superior ub, o gap é $\frac{ub-lb}{ub} \times 100$. As Figuras 5.1, 5.2 e 5.3 apresentam a evolução da média do gap em tempo para os grupos de instâncias *long*, *medium* e *sprint*, respectivamente. Observa-se que a inclusão das desigualdades levantadas permite uma rápida redução do gap. Além disso, os cortes `eclq` ainda progridem enquanto os cortes `cgl` não podem realizar qualquer alteração significativa nos limites duais. A separação via ciclos ímpares não apresentou nenhuma surpresa nesta abordagem: conforme visto em trabalhos anteriores, este método não tem um impacto significante para a melhoria dos limites duais. Uma razão para isto é que muitas violações via ciclos-ímpares encontradas são k_3, de modo que a separação de clique já encontra. Violações de ciclos-ímpares de tamanho 5 ou superiores são escassos na relaxação do nó raiz. No entanto, estes são cortes de segurança (ou seja, não dependem de arredondamento dos números calculados limitados com precisão de ponto flutuante) que pode ser separado instantaneamente, de modo que vale a pena manter os procedimentos de em *branch-and-cut*, mesmo que tenha uma contribuição marginal observada na relaxação do nó raiz.

Após uma série de experimentos com todos os cortes disponíveis na Biblioteca de Geração de Cortes COIN-OR (CGL), constatou-se que os seguintes planos de cortes genéricos foram também úteis na melhoria do limite dual: *Mixed Integer Gomory* (Balas, Ceria, Cornueljols and Natra 1996), *Two-Step Mixed Integer Rounding* (Dash, Goycoolea

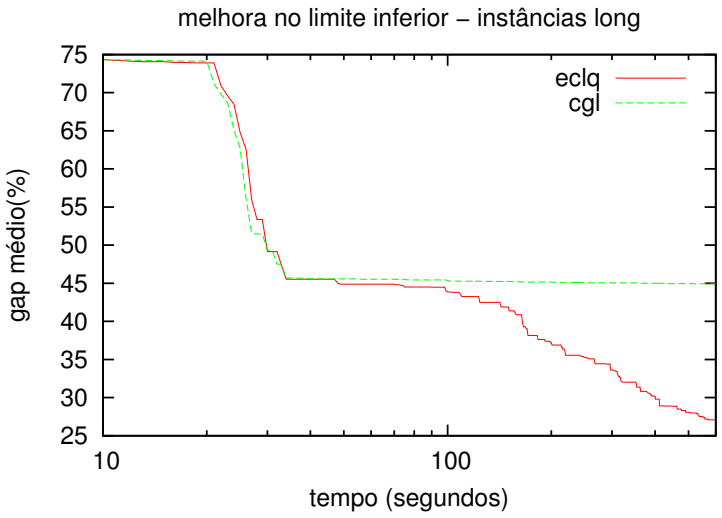

Figura 5.1: Melhoria do limite dual para COIN-OR construído na geração de cortes (`cgl`) e o procedimento proposto de separação de corte (`eclq`)

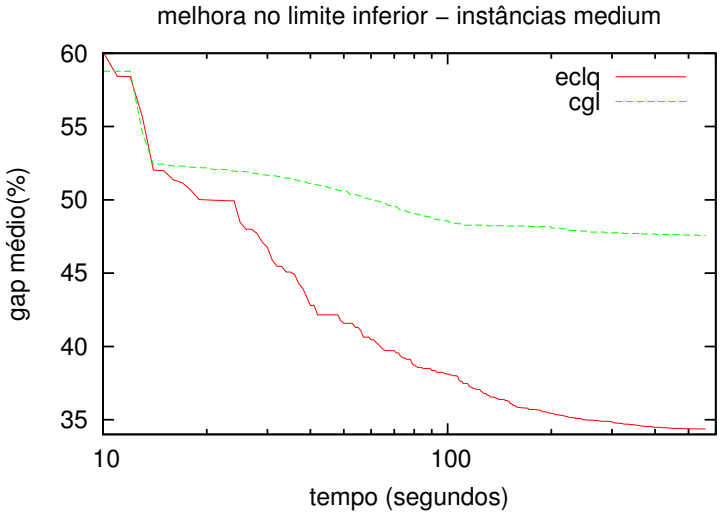

Figura 5.2: Melhoria do limite dual para COIN-OR construído na geração de cortes (`cgl`) e o procedimento proposto de separação de corte (`eclq`)

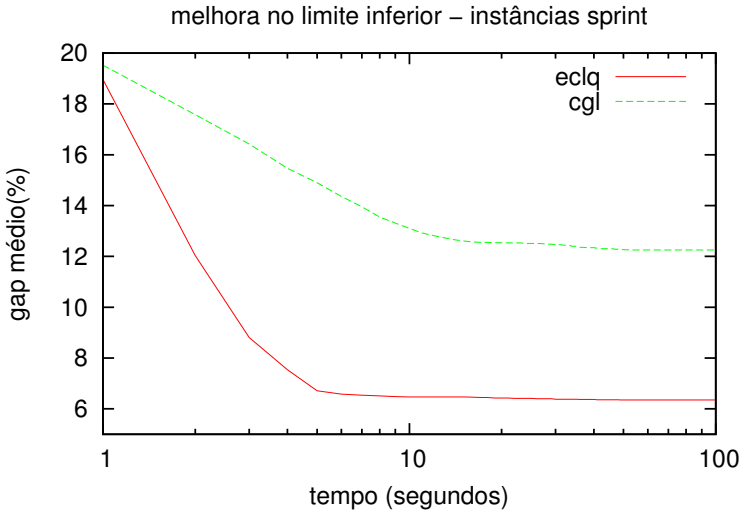

Figura 5.3: Melhoria do limite dual para COIN-OR construído na geração de cortes (`cgl`) e o procedimento proposto de separação de corte (`eclq`)

and Gunluk 2009), *RedSplit* (Andersen, Cornuejols and Y. 2005) e cortes *Zero-Half* $(0/\frac{1}{2})$ (Andreello, Caprara and Fischetti 2007). Cortes *Zero-Half* não estão disponíveis ainda na última versão oficial CGL, mas os autores gentilmente ofereceram os códigos para os experimentos deste trabalho. A contribuição de todos estes cortes adicionais aplicados em conjunto com os cortes desta abordagem para melhorar o limite inferior para cada grupo de instâncias é apresentada na Tabela 5.5. Considerando o limite da programação da relaxação linear (lp) e o limite inferior obtido no final da aplicação do corte no nó raiz no CBC (lb) foram computados para cada instância as melhorias: $min\{\frac{lb-lp}{lp+\epsilon} \times 100, 100\}$, onde ϵ é uma pequena constante para evitar a divisão por zero. Um resumo destes resultados estão apresentados em 5.5. Como pode ser visto, embora os cortes de Gomory tenham uma alta relevância para as instâncias pequenas, para as instâncias maiores esta relevância passa a ser de menor expressão. O motivo é que estes cortes tendem a produzir restrições muito densas para grandes programas lineares e são provavelmente descartadas pelo código *branch and cut* do CBC nas instâncias maiores. Os cortes de Janela propostos, por outro lado, parece ter maior importância nas instâncias maiores.

5.3 Heurísticas de Programação Matemática

As heurísticas implementadas utilizam o CPLEX para resolver os subproblemas. O modo paralelo foi desabilitado, para que tanto as heurísticas quanto o CPLEX fossem executados sequencialmente. Todas as 60 instâncias da INRC (Haspeslagh, De Causmaecker, Stolevik and A. 2010) foram testadas. No caso das heurísticas de fixação fraca, o parâmetro k assumiu os valores 5% e 7%, representando respectivamente estes percentuais do total de variáveis ativas da solução corrente. No caso das heurísticas de fixação forte, o parâmetro m assumiu valores de 1 a 9.

Os resultados obtidos pela heurística HPM são apresentados nas tabelas 5.6 e 5.7. Nesta tabela, a coluna MSC apresenta as melhores soluções conhecidas, incluindo as encontradas neste trabalho (marcadas com um ⊛). As colunas seguintes apresentam os melhores resultados encontrados na literatura (LSA), os melhores resultados do CPLEX, os melhores resultados obtidos pela *HPM* com m na faixa de 1 a 9 e o melhor resultado obtido em cada um destes. Para cada resultado, a tabela reporta o melhor limite superior (LS) obtido e o gap entre limite superior (LS) e a melhor solução conhecida (MSC): $\frac{LS-\text{MSC}}{LS} \times 100$.

Os resultados para a categoria *sprint* não serão apresentados pois, para todas elas, a heurística foi capaz de encontrar a solução ótima dentro de 3 minutos com m iniciando com valores entre o intervalo [1,9]. Os resultados para as demais categorias *long* e *medium* serão apresentados nas Tabelas 5.6 e 5.7, respectivamente.

Alguns comentários sobre os resultados apresentados nas Tabelas 5.6 e 5.7:

- O impacto dos parâmetros m é muito perceptível e o melhor resultado médio foi obtido quando $m = 1$.
- Os limites superiores fornecidos pela *HPM* são certamente muito bons, especialmente quando $m = 1$ na primeira iteração. Considerando todas as execuções (m iniciando de 1 a 9), a heurística foi capaz de superar a melhor solução da literatura para 6 instâncias. Considerando as instâncias da categoria *sprint*, o gap era superior a 0 apenas para 7 das 60 instâncias. A diferença média entre a melhor solução conhecida também foi muita baixa: 0.5% para as instâncias da categoria *long* e 0.6% para a categoria *medium*, com um gap de 5.3% considerando todas as instâncias, que é especificamente relacionado a uma unidade no valor da função

objetivo para a instância *medium_late02*.

- Desde que a *HPM* foi capaz de encontrar boas soluções de forma robusta em menos de 10 minutos de processamento sequencial, decidiu-se não apresentar resultados de longas execuções. Esta decisão se fundamenta no fato de que a execução da heurística por um longo período de tempo, o resultado é a solução do problema original (veja Sessão 4.1.3), o que pode levar muito tempo para algumas instâncias mais pesadas.

- Na comparação com os limites superiores do método e os melhores resultados da literatura, deve-se levar em conta a diferença da ordem de magnitude dos tempos de execução para as instâncias da categoria *long*. Enquanto *HPM* tem tempos de execução limitados a 600 segundos, Valouxis et al. (Valouxis, Gogos, Goulas, Alefragis and Housos 2012) reporta tempos de até 36.000 segundos nestas instâncias. A comparação é mais justa quando se considera as instâncias da categoria *medium*, desde que executando em tempos semelhantes.

As Figuras 5.4, 5.5 e 5.6 mostram a melhoria do tempo no que se refere ao limite superior de *HPM* usando diferentes valores para o m inicial, para as categorias de instâncias *Long*, *Medium* e *Sprint*, respectivamente. A partir dessa figura, conclui-se que o valores inferiores a 7 para o m inicial, a *HPM* é capaz de produzir boas soluções nos estágios iniciais da busca. A figura mostra também que, como o valor inicial do m torna-se maior, a heurística leva mais tempo para gerar soluções melhores. Tempos adicionais neste caso não são vantajosos, uma vez que a solução final é ainda pior do que as produzidas pela *HPM* com menor m na primeira iteração.

A Tabela 5.8 apresenta os resultados obtidos pelas heurísticas HPM-$\mathcal{LB}$ e HPM-$\mathcal{ESP}$, comparando estes com os melhores resultados alcançados pela heurística HPM. Esta tabela apresenta os resultados apenas para as instâncias mais difíceis, das categorias *long_hidden* e *medium_hidden*. Por esta tabela, percebe-se claramente que a abordagem de fixação fraca utilizada não é apropriada para o problema tratado nesta dissertação.

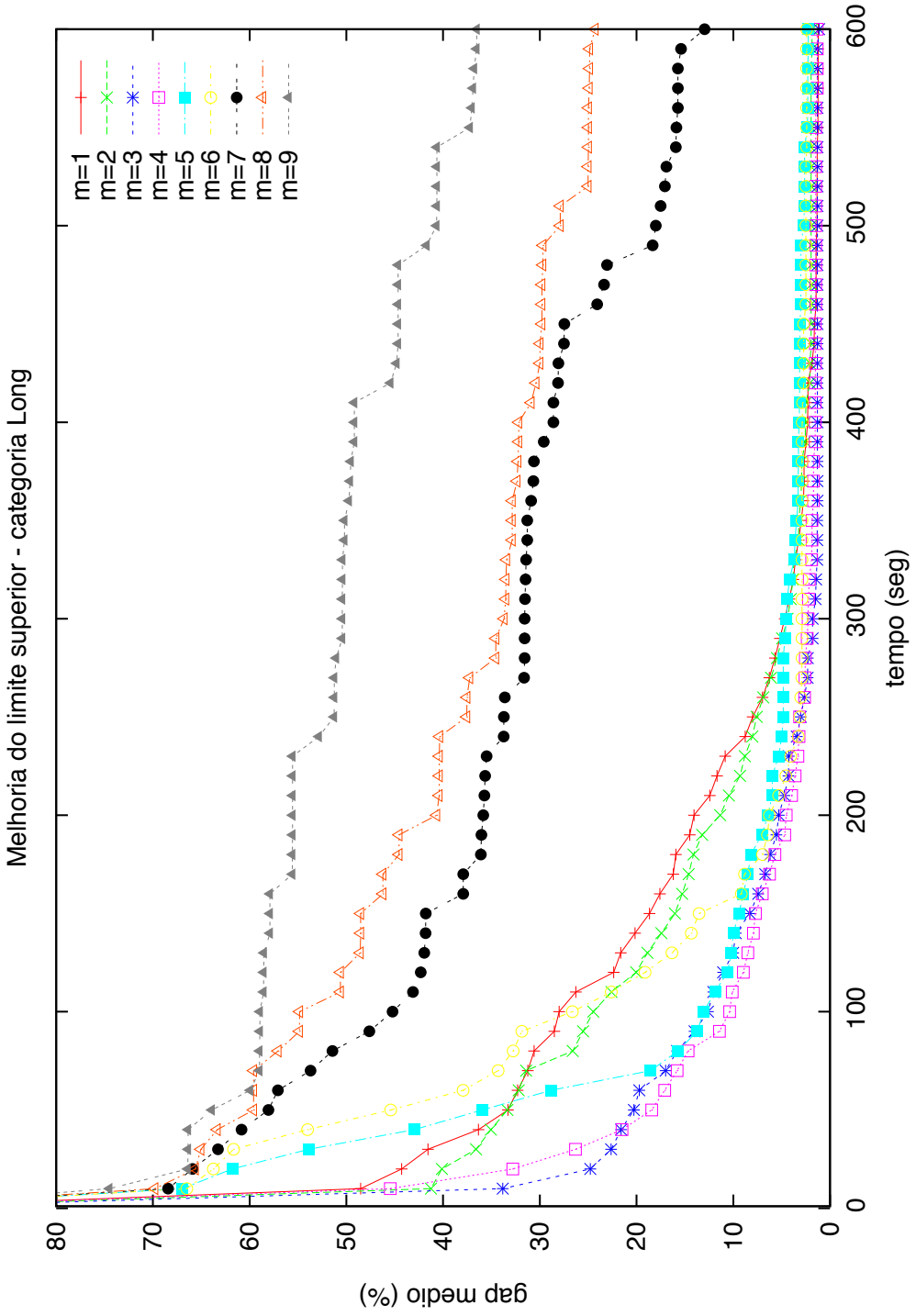

Figura 5.4: Melhorias das *HPM* com diferentes valores para o m inicial nas instâncias Long.

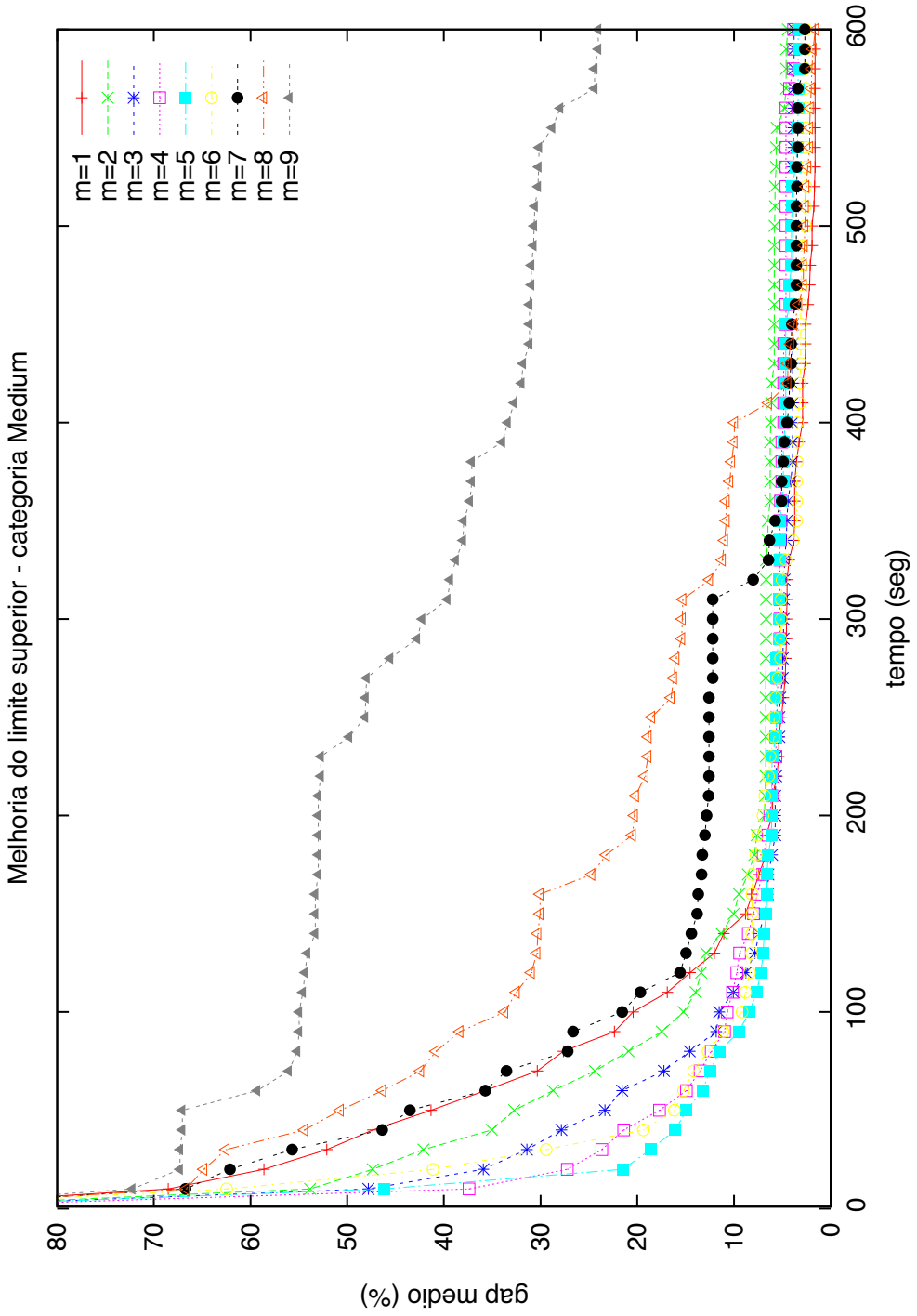

Figura 5.5: Melhorias das *HPM* com diferentes valores para o m inicial nas instâncias Medium.

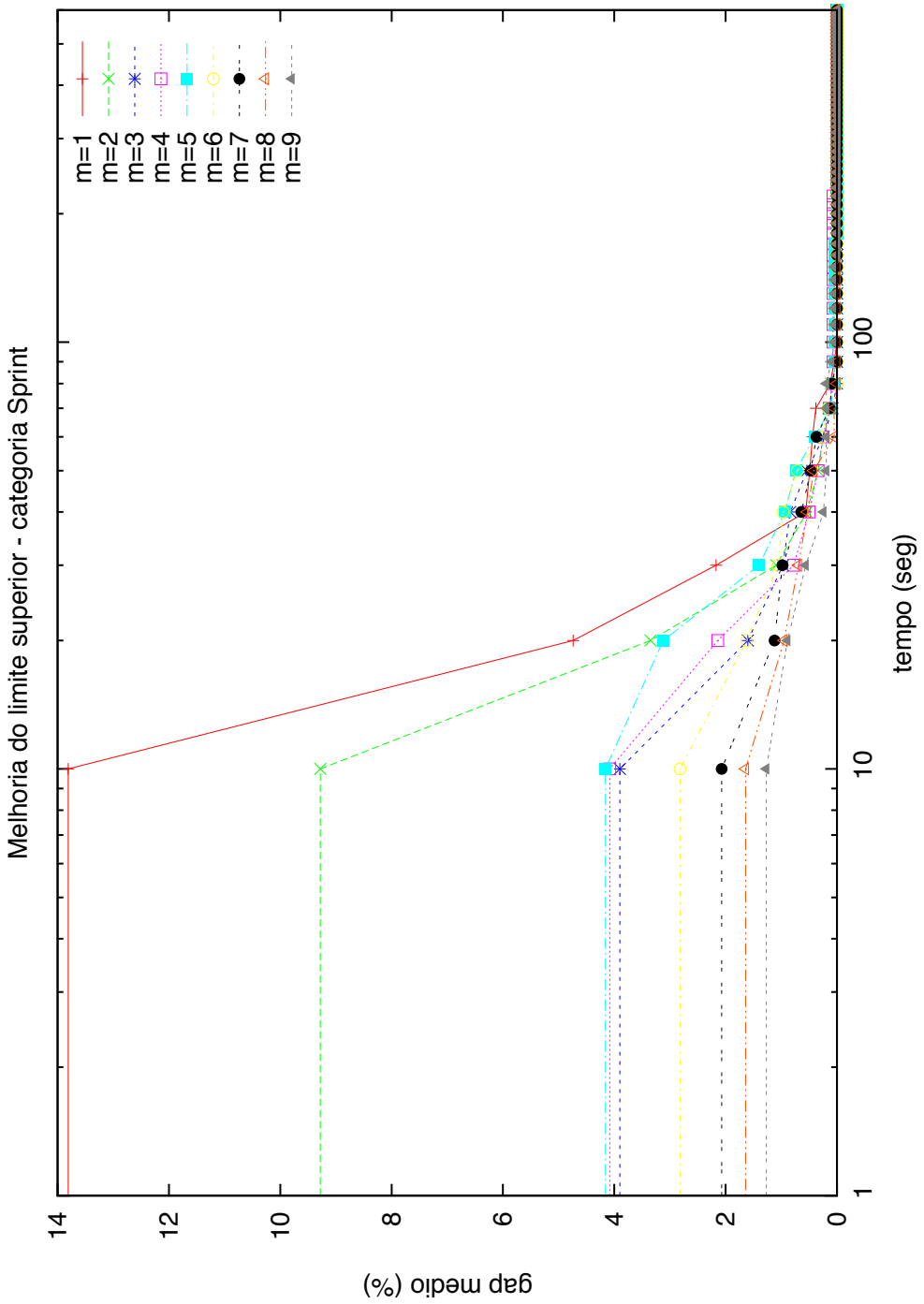

Figura 5.6: Melhorias das *HPM* com diferentes valores para o m inicial nas instâncias Sprint.

5.4 Melhores resultados

Os melhores resultados obtidos de todos os experimentos são apresentados na Tabela 5.9. Células da tabela destacadas com ⊛ indicam alguma melhoria sobre a melhor solução conhecida e reportada ao site da INRC até o momento da escrita deste trabalho. É importante destacar que este site recebeu atualizações nos anos seguintes após a competição, assim as melhores soluções anteriormente conhecidas (coluna LSA) já eram difíceis de serem encontradas. A maioria das instâncias foram resolvidas na otimalidade e as mais difícil entre as instâncias não resolvidas é a *medium_hidden05* onde a distância do limite inferior é atualmente 23.7%.

Tabela 5.3: Resultados do resolvedor comercial CPLEX com parâmetros inalterados para instâncias da categoria *long*

			CPLEX12.1 - Configuração padrão										**CPLEX12.1 - Ênfase heurística**									
			10min / $s = \emptyset$			10min / $s = gr(.)$			1hora / $s = \emptyset$			1hora / $s = gr(.)$		10min / $s = \emptyset$			10min / $s = gr(.)$			1hora / $s = \emptyset$			1hora / $s = gr(.)$		
	Instância		LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap
long	early	01	197	197	0.0	197	197	0.0	197	197	0.0	197	197	0.0	197	197	0.0	197	197	0.0	197	197	0.0	197	197	0.0
		02	219	219	0.0	219	219	0.0	219	219	0.0	219	219	0.0	219	219	0.0	219	219	0.0	219	219	0.0	219	219	0.0
		03	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0
		04	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0
		05	284	284	0.0	284	284	0.0	284	284	0.0	284	284	0.0	284	284	0.0	284	284	0.0	284	284	0.0	284	284	0.0
	hidden	01	319	∞	∞	319	487	34.5	337	615	45.2	334	374	10.7	319	∞	∞	319	487	34.5	337	602	44.0	334	380	12.1
		02	81	∞	∞	81	111	27.0	81	153	47.1	86	92	6.5	81	∞	∞	81	111	27.0	81	143	43.4	86	93	7.5
		03	17	∞	∞	18	61	71.0	17	∞	∞	25	55	55.2	17	∞	∞	18	61	71.0	26	∞	∞	26	51	49.9
		04	14	∞	∞	14	44	68.8	14	∞	∞	19	44	56.1	14	∞	∞	14	44	68.8	19	∞	∞	14	42	67.2
		05	36	∞	∞	36	130	72.7	40	∞	∞	41	41	0.0	36	∞	∞	36	130	72.7	40	∞	∞	40	46	14.1
	late	01	212	∞	∞	204	385	46.9	231	∞	∞	232	237	2.1	212	∞	∞	204	385	46.9	231	∞	∞	233	267	12.7
		02	214	∞	∞	208	409	49.2	229	282	18.8	229	229	0.0	214	∞	∞	208	409	49.2	229	517	55.7	229	229	0.0
		03	213	∞	∞	212	391	45.8	218	250	12.8	218	221	1.4	213	∞	∞	212	391	45.8	216	295	26.8	218	225	3.1
		04	196	∞	∞	197	310	36.5	213	∞	∞	213	230	7.4	196	∞	∞	197	310	36.5	213	∞	∞	212	234	9.3
		05	79	635	87.5	79	270	70.9	79	542	85.4	80	229	65.3	79	635	87.5	79	270	70.9	79	545	85.5	80	269	70.4
	(med. / d.p.)		(14.6 / 35.7)			(34.9/ 29.0)			(20.9 / 29.2)			(13.6 / 23.7)		(14.6 / 35.7)			(34.9 / 29.0)			(25.5 / 30.6)			(16.4 / 24.8)		

Tabela 5.4: Resultados do resolvedor comercial CPLEX com parâmetros inalterados para instâncias da categoria *medium*

			CPLEX12.1 configuração padrão								**CPLEX12.1 ênfase heurística**							
			10min / $s = \emptyset$			10min / $s = gr(.)$		1hour / $s = \emptyset$			10min / $s = \emptyset$			10min / $s = gr(.)$		1hour / $s = \emptyset$			1hour / $s = gr(.)$		
	Instância		LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap	LI	LS	gap
medium	early	01	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0	240	240	0.0	240	240	0.0
		02	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0.0	240	240	0	240	240	0.0	240	240	0.0
		03	236	236	0.0	236	236	0.0	236	236	0.0	236	236	0.0	236	236	0	236	236	0.0	236	**236**	0.0
		04	237	237	0.0	237	237	0.0	237	237	0.0	237	237	0.0	237	237	0	237	237	0.0	237	237	0.0
		05	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0.0	303	303	0	303	303	0.0	303	303	0.0
	hidden	01	60	1093	94.5	59	314	81.3	62	287	78.3	60	1093	94.5	59	314	81.33	66	1060	93.8	64	220	71.1
		02	183	∞	∞	180	352	48.9	188	312	39.6	183	∞	∞	180	352	48.9	189	291	35.2	188	291	35.3
		03	22	449	95.0	22	93	76.0	23	93	74.9	22	449	95.0	22	93	75.99	24	60	60.3	24	49	51.9
		04	58	∞	∞	59	136	57.0	60	97	38.6	58	∞	∞	59	136	56.99	62	96	35.8	62	109	42.8
		05	56	∞	∞	56	233	75.9	84	488	82.9	56	∞	∞	56	233	75.89	84	427	80.4	82	205	60.0
	late	01	148	171	13.5	145	215	32.5	152	157	3.1	149	166	10.5	147	212	30.77	151	158	4.4	151	157	3.5
		02	18	18	0.0	18	18	0.0	18	18	0.0	18	18	0.0	18	18	0	18	18	0.0	18	18	0.0
		03	29	29	0.0	29	29	0.0	29	29	0.0	29	29	0.0	29	29	0	29	29	0.0	29	29	0.0
		04	33	35	4.4	34	37	9.4	34	35	2.9	35	35	0.0	33	37	9.8	35	35	0.0	35	35	0.0
		05	104	114	9.2	103	120	13.9	106	107	1.0	104	123	15.8	103	216	52.55	107	107	0.0	107	107	0.0
	(med. / d.p.)			(18.0 / 36.1)			(26.3 / 32.4)			(21.0 / 33.0)			(18.0 / 36.2)			(28.8/ 32.9)			(20.7 / 32.8)			(17.6 / 26.4)	
										(18.6 / 29.4)													

Tabela 5.5: Contribuição de diferentes cortes para melhorias do limite inferior na relaxação do nó raiz

		Window	Zero-Half	Gomory	RedSplit	TwoMIR
sprint	máx.	50.1	51.5	93.7	52.9	52.8
	min.	1.9	3.7	3.7	3.7	3.7
	med.	19.7	21.6	26.2	22.3	22.6
	desv.padr	15.2	15.3	23.1	15.8	15.9
medium	máx.	100.0	100.0	100.0	100.0	100.0
	min.	0.0	0.0	0.0	0.0	0.0
	med.	50.1	51.2	51.9	50.7	41.1
	desv.padr	46.6	46.3	47.7	46.7	47.9
long	máx.	100.0	100.0	58.9	100.0	36.5
	min.	0.0	0.0	0.0	0.0	0.0
	med.	37.5	38.7	14.5	38.0	5.9
	desv.padr	39.1	39.3	17.8	40.3	9.3

Tabela 5.6: Resultados das HPM para instâncias da categoria *long*

Instância			MSC	LSA		CPLEX ($m = 14$)		melhor HPM ($m \in [1, 9]$)		Resultados de HPM com diferentes valores de m																	
										$m = 1$		$m = 2$		$m = 3$		$m = 4$		$m = 5$		$m = 6$		$m = 7$		$m = 8$		$m = 9$	
				LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap
long	early	01	197	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0	197	0.0
		02	219	219	0.0	219	0.0	219	0.0	219	0.0	220	0.5	220	0.5	220	0.5	220	0.5	219	0.0	219	0.0	219	0.0	219	0.0
		03	240	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0
		04	303	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0
		05	284	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0	284	0.0
	hidden	01	⊛ **346**	363	4.7	487	29.0	346	0.0	347	0.3	348	0.6	346	0.0	346	0.0	359	3.6	346	0.0	355	2.5	534	35.2	506	31.6
		02	⊛ **89**	90	1.1	111	19.8	89	0.0	89	0.0	89	0.0	89	0.0	89	0.0	89	0.0	89	0.0	89	0.0	89	0.0	96	7.3
		03	38	38	0.0	61	37.7	39	2.6	39	2.6	40	5.0	40	5.0	41	7.3	41	7.3	42	9.5	130	70.8	119	68.1	118	67.8
		04	22	22	0.0	44	50.0	22	0.0	22	0.0	22	0.0	22	0.0	22	0.0	22	0.0	22	0.0	37	40.5	41	46.3	112	80.4
		05	41	41	0.0	130	68.5	41	0.0	43	4.7	45	8.9	42	2.4	43	4.7	43	4.7	43	4.7	41	0.0	49	16.3	174	76.4
	late	01	235	235	0.0	385	39.0	239	1.7	239	1.7	241	2.5	239	1.7	241	2.5	242	2.9	244	3.7	244	3.7	244	3.7	257	8.6
		02	229	229	0.0	409	44.0	235	2.6	235	2.6	235	2.6	237	3.4	235	2.6	239	4.2	242	5.4	235	2.6	242	5.4	243	5.8
		03	220	220	0.0	391	43.7	220	0.0	221	0.5	225	2.2	221	0.5	220	0.0	233	5.6	239	7.9	233	5.6	220	0.0	380	42.1
		04	221	221	0.0	310	28.7	222	0.5	225	1.8	231	4.3	228	3.1	224	1.3	229	3.5	227	2.6	222	0.5	228	3.1	233	5.2
		05	83	83	0.0	270	69.3	83	0.0	86	3.5	83	0.0	83	0.0	83	0.0	83	0.0	83	0.0	92	9.8	159	47.8	240	65.4
gap (med./d.p.)				(0.4/1.2)		(28.6/24.7)		(0.5/0.9)		(1.2/1.5)		(1.8/2.6)		(1.1/1.6)		(1.3/2.2)		(2.1/2.5)		(2.3/3.3)		(9.1/20.0)		(15.1/22.7)		(26.0/31.6)	

Tabela 5.7: Resultados das HPM para instâncias da categoria *medium*

Instância				LSA		CPLEX ($m = 14$)		melhor HPM ($m \in [1,9]$)		Resultados de HPM com diferentes valores de m															
										$m=1$		$m=2$		$m=3$		$m=4$		$m=5$		$m=6$		$m=7$		$m=8$		$m=9$	
			MSC	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap	LS	gap
medium	early	01	240	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0
		02	240	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0	240	0.0
		03	236	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0	236	0.0
		04	237	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0	237	0.0
		05	303	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0	303	0.0
	hidden	01	⊛ **111**	130	14.6	314	64.6	111	0.0	116	4.3	111	0.0	118	5.9	118	5.9	116	4.3	117	5.1	118	5.9	120	7.5	307	63.8
		02	221	221	0.0	352	37.2	221	0.0	221	0.0	223	0.9	224	1.3	227	2.6	223	0.9	235	6.0	221	0.0	226	2.2	256	13.7
		03	⊛ **34**	36	5.6	93	63.4	34	0.0	34	0.0	40	15.0	41	17.1	38	10.5	36	5.6	38	10.5	36	5.6	36	5.6	41	17.1
		04	⊛ **78**	80	2.5	136	42.6	80	2.5	80	2.5	80	2.5	84	7.1	81	3.7	82	4.8	81	3.7	81	3.7	80	2.5	147	46.9
		05	⊛ **119**	122	2.5	233	48.9	119	0.0	121	1.7	120	0.8	123	3.3	121	1.7	120	0.8	119	0.0	127	6.3	121	1.7	232	48.7
	late	01	⊛ **157**	158	0.6	215	27.0	157	0.0	161	2.5	160	1.9	160	1.9	161	2.5	161	2.5	157	0.0	159	1.3	161	2.5	164	4.3
		02	18	18	0.0	18	0.0	19	5.3	19	5.3	19	5.3	22	18.2	23	21.7	24	25.0	20	10.0	20	10.0	20	10.0	24	25.0
		03	29	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0
		04	35	35	0.0	37	5.4	35	0.0	35	0.0	36	2.8	35	0.0	36	2.8	36	2.8	37	5.4	37	5.4	35	0.0	36	2.8
		05	107	107	0.0	120	10.8	108	0.9	108	0.9	114	6.1	113	5.3	118	9.3	114	6.1	114	6.1	119	10.1	109	1.8	112	4.5
gap (med./d.p.)				(1.7/3.9)		(20.0/24.9)		(0.6/1.5)		(1.1/1.7)		(2.4/4.0)		(4.0/6.0)		(4.1/6.0)		(3.5/6.4)		(3.1/3.8)		(3.2/3.7)		(2.2/3.1)		(15.1/21.3)	

Tabela 5.8: Resultados obtidos com as heurísticas HPM-$\mathcal{LB}$ e HPM-$\mathcal{ESP}$

Instância	HPM-$\mathcal{LB}$		HPM-$\mathcal{EPS}$		*HPM*
	$k = 5\%$	$k = 8\%$	$k = 5\%$	$k = 8\%$	
long_hidden_01	570	799	1063	818	359
long_hidden_02	101	108	153	131	89
long_hidden_03	148	127	184	132	41
long_hidden_04	114	79	97	95	22
long_hidden_05	161	150	188	209	43
medium_hidden_01	283	330	351	321	116
medium_hidden_02	375	409	344	387	223
medium_hidden_03	84	86	87	82	36
medium_hidden_04	153	152	167	181	82
medium_hidden_05	330	292	348	278	120

Tabela 5.9: limite superior anterior (LSA), limite inferior atualizado (LI) e limite superior atualizado (LS)

Instância			LI	LSA	LS	GAP
long	early	01	197.0	197	197	0.0
		02	219.0	219	219	0.0
		03	240.0	240	240	0.0
		04	303.0	303	303	0.0
		05	284.0	284	284	0.0
	hidden	01	341.0	363	⊛ **346**	1.4
		02	86.0	90	⊛ **89**	3.4
		03	35.3	38	38	7.1
		04	19.0	22	22	13.8
		05	41.0	41	41	0.0
	late	01	232.0	235	235	1.3
		02	229.0	229	229	0.0
		03	219.0	220	220	0.5
		04	214.6	221	222	3.3
		05	83.0	83	83	0.0
medium	early	01	240.0	240	240	0.0
		02	240.0	240	240	0.0
		03	236.0	236	236	0.0
		04	237.0	237	237	0.0
		05	303.0	303	303	0.0
	hidden	01	87.2	130	⊛ **111**	21.5
		02	196.6	221	221	11.1
		03	27.7	36	⊛ **34**	18.5
		04	72.8	80	⊛ **78**	6.7
		05	90.8	122	⊛ **119**	23.7
	late	01	155.7	158	⊛ **157**	0.8
		02	18.0	18	18	0.0
		03	29.0	29	29	0.0
		04	35.0	35	35	0.0
		05	107.0	107	107	0.0

Instância			LI	LSA	LS	GAP
sprint	early	01	56.0	56	56	0.0
		02	58.0	58	58	0.0
		03	51.0	51	51	0.0
		04	59.0	59	59	0.0
		05	58.0	58	58	0.0
		06	54.0	54	54	0.0
		07	56.0	56	56	0.0
		08	56.0	56	56	0.0
		09	55.0	55	55	0.0
		10	52.0	52	52	0.0
	hidden	01	32.0	33	⊛ **32**	0.0
		02	32.0	32	32	0.0
		03	62.0	62	62	0.0
		04	66.0	67	⊛ **66**	0.0
		05	59.0	59	59	0.0
		06	130.0	134	⊛ **130**	0.0
		07	153.0	153	153	0.0
		08	204.0	209	⊛ **204**	0.0
		09	338.0	338	338	0.0
		10	306.0	306	306	0.0
	late	01	37.0	37	37	0.0
		02	42.0	42	42	0.0
		03	48.0	48	48	0.0
		04	73.0	75	⊛ **73**	0.0
		05	44.0	44	44	0.0
		06	42.0	42	42	0.0
		07	42.0	42	42	0.0
		08	17.0	17	17	0.0
		09	17.0	17	17	0.0
		10	43.0	43	43	0.0

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho apresentou Técnicas de Programação Inteira para o Problema de Escalonamento de Enfermeiras. Embora existam vários resultados detalhados publicados na literatura para heurísticas avaliadas usando os conjunto de instâncias da INRC, acredita-se que este é o primeiro trabalho que também dedicou-se à produção computacional de fortes limites duais obtidos a partir da relaxação da Programação Linear. Estes limites permitiram provar a otimalidade para muitas instâncias e sua importância não se restringe a métodos exatos: melhores limites duais permitem uma poda mais eficaz nos nós da árvore de busca, o que é útil para acelerar a busca da heurística MIP em grandes vizinhanças, conforme apresentado neste trabalho. O grande número de experimentos feitos usando o resolvedor de código aberto CBC, permitiu a detecção de erros anteriormente desconhecidos no código do mesmo. Esses erros foram posteriormente corrigidos pelos desenvolvedores. Foi proposto e implementado um gerador de cortes de clique melhor para o CBC, monstrando que ele pode produzir melhores limites duais nos estágios iniciais da busca. Esse código também será liberado como um projeto de código aberto. Este trabalho não foi suficiente para tornar o CBC competitivo nas instâncias INRC quando comparado aos melhores resolvedores comerciais, já que para desenvolver uma heurística MIP competitiva, ainda foi preciso contar com o CPLEX. No entanto, acredita-se que este é um passo significativo para validar e melhorar este importante resolvedor de Programação Linear de código aberto.

A heurística MIP proposta, contruída sobre a formulação apresentada e avaliada com o resolvedor CPLEX, melhorou várias das melhores soluções conhecidas, exigindo menores tempos de execução e ainda sendo competitiva com as melhores heurísticas já propostas para este problema. Acredita-se que os novos limites primais e duais permi-

tirão uma avaliação mais exata da qualidade das heurísticas para este problema.

Como trabalhos futuros, pretende-se utilizar a heurística MIP desenvolvida em outros problemas de *Timetabling* vinculado a um resolvedor genérico. Espera-se bons resultados também com a implementação de um algoritmo de *Branch & Cut & Price* robusto. Envolver geração de colunas e cortes demonstra ser interessante para este problema, visto que resultados promissores já foram encontrados somente com a geração de colunas em outras abordagens.

Apêndice A

Apêndices

A.1 Publicações

Neste apêndice são listados os trabalhos desenvolvidos nesta pesquisa que foram publicados em periódicos ou apresentados em eventos científicos até esta data (19 de dezembro de 2012).

- Santos, H. G., Toffolo, T. A. M., Gomes, R. A. M., Ribas, S. (2012). Integer Programming Techniques for the Nurse Rostering Problem. *International Conference on the Pratice and Theory of Automated Timetabling* (PATAT '12).

- Santos, H. G., Toffolo, T. A. M., Gomes, R. A. M., Vareto, R. H. (2012). Heurísticas e Programação Inteira para o Problema das Enfermeiras. *Congresso Latino-Iberoamericano de Investigatión Operativa* (CLAIO-SBPO '12).

A.2 Site oficial da INRC

A figura A.1 apresenta a página oficial da INRC. Neste *site*, os melhores resultados obtidos são divulgados para que pesquisadores possam contrastar seus resultados. A avaliação é feita através de um *software online*, disponibilizado na própria página da competição.

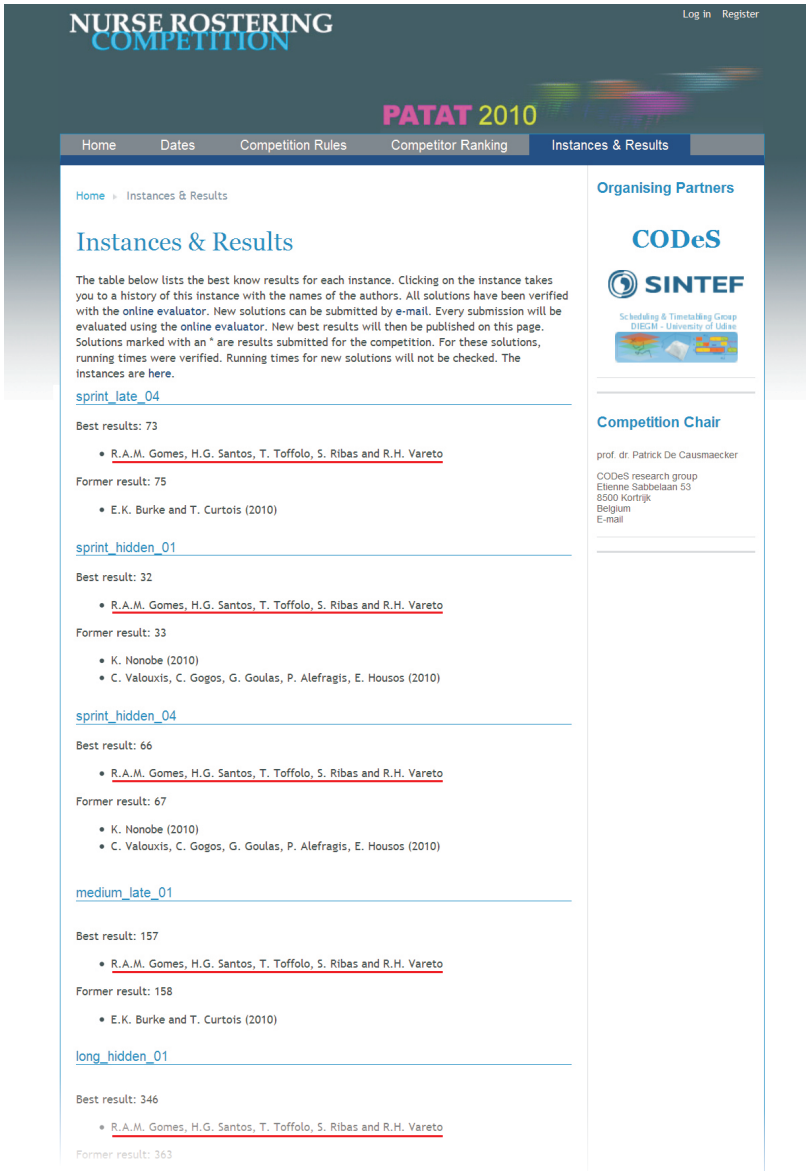

Figura A.1: Publicação dos melhores resultados obtidos no site oficial da competição: http://www.kuleuven-kulak.be/nrpcompetition/instances-results

A.3 Escalas melhoradas de enfermeiras

Neste apêndice são apresentadas algumas das soluções encontradas (escalas) onde houve melhoria da melhor solução conhecida da literatura. As Tabelas A.1, A.2, A.3, A.5 e A.4, além disso, também são apresentadas algumas escalas onde foi obtido o ótimo global da instância, como as escalas apresentadas em A.5 e A.4.

Nestas tabelas, cada linha representa uma enfermeira e cada coluna um dia no ho-

rizonte de planejamento. Em cada célula é indicado o turno que a enfermeira deverá trabalhar. Células vazias indicam um dia de folga da enfermeira.

Tabela A.1: Instância medium_hidden01 - LSA: 122 / LS: 111

Enf.	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
0	E	E	E	L			N	L	N			D	D	E	E	E			E	E	E	D	D			N	N	L
1	E	E	D	D			D	L	N	L	L			E	N	N			E	E	N	L	N			L	L	L
2				L	L	L			L	L	N	L	L			E	E	N	N	N			E	L				
3	L	N	N				DH	DH	DH	DH	DH			DH	DH	DH			DH	DH	DH	DH		DH	DH	DH	DH	DH
4				E	N	N	N			N	N	L	L				E	E			L	N	L	N	L			
5				E	E	E			L	N	N	L	L		E	D	L	L			E	D	E	E	L			
6	L	L	L				E	N	N	L		E	E	L	N			D	D	N	L	L			E	E	E	E
7				D	E	E		E	E	E	E	D	D			E	L	L	N	N			D	D	L			
8	N	N	L				E	E	E	E			N	N	L	N			E	E	N	L	N			E	E	E
9	L	L	N	L			E	D	D	E	E			N	L	N			E	E	D	E	L			N	N	L
10				N	N	N	L				E	N	N	N				D	D	D	D	E	E	L				
11				N	L	L	L	N				N	N	L	L	L	N			N	E	E	E	L				
12				E	D	D	L					L	L	L				L	N	N	N	L	L	N				
13				N	N	N	N				E	E	E	D	E	L	L				L	L	N	L				
14				E	D	D	D	L			E	L	L	L	L	L	N				E	E	DH	N				
15					L	L	L	L						L	D	L					L	N	L	L				
16					E	E	E					E	E	L	N						E	E	E	D				
17	E	D	E					E	E	E	E						N	L	L	L					E	E	E	D
18	D	D	E					E	E	D	D						L	N	L	L					N	L	L	N
19	L	N	N					L	L	L	L						E	E	L	L					E	D	D	D
20							E	D	L	L					E	D	D	E							E	E	E	E
21	N	L	L					E	D	E	L						D	E	E	E					D	D	D	E
22	E	E	E					N	L	N	L						E	E	L	L					L	N	N	N
23					E	E	L					E	E	E	L								L	L				
24	D	E	E					E	E	D	D						E	N	L	L			N	L	N	L	L	N
25												E	E	E	E	L	L	L										
26	N	L	L	L	L	L																	E	D	E	E	E	E
27	L	L	L	L	L	L																	E	N	L	L	L	L
28	E	E	D	E	E	E																	E	E	L	L	L	L
29	DH	DH	DH	DH	DH	DH						DH	DH	D	D	E	DH	DH										

Tabela A.2: Instância medium_hidden03 - LSA: 36 / LS: 34

Enf.	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
0				E	DH	DH	DH	N			L	L	L	N	N			L	L	L	L	N			E	E	E	E
1				L	L	L	N	L			L	L	L	L	L			E	E	E	D	E			N	L	L	N
2	DH	DH	E				E	DH	N			E	E	E	L	N			E	E	DH	DH	L			E	E	DH
3				N	N	N			E	E	D	DH	DH			E	E	D	E	E			DH	D	L	L	L	N
4	E	D	E	L	L	L			E	L	N	N	N			D	DH	E	DH	DH			D	E	E			
5				N	L	L	N	L			E	E	E	E	DH			E	N	N	L	L			D	E	E	D
6			DH	DH	D	D	D			D	DH	E	E	E			L	N	L	L	N			E	E	D	D	E
7				E	D	D		D	DH	E	L	N	N			DH	N	N	L	L			D	DH	DH	DH	DH	E
8	L	L	N				E	D	D	E				L	L	L		DH	D	D	E	L			L	N	N	N
9				L	N	N	N			DH	N	L	L	L	L			L	L	L	L	L			N	L	L	L
10				D	E	E	L				L	L	L	N	L	N	L				E	D	E	E				
11				N	L	L	L				L	N	N	N	N	L	L				N	N	N	L				
12				L	L	L	L				E	D	D	D	E	L	N				E	E	E	D				
13				D	E	E	E				N	L	L	L	L			E	D	D	L	L	L	N				
14				L	N	N	L				E	E	E	L				L	N	N	L	N	L	L	N				
15	E	E	L					E	E	E	D						E	D	E	E					N	N	N	L
16	D	E	E					L	N	L							N	N	L	L					L	L	L	L
17	N	L	L					N	L	N					E	E	D	L							E	E	E	E
18	L	L	L					E	E	E							E	L	N	N					D	D	D	E
19	E	E	D					D	D	D				D	D	E					E	E	E	N				
20	L	L	L					E	N	N					N	N	L					E	E	E				
21	N	N	N					L	L	N					E	E	E				L	N	L	L				
22	L	N					L	N	L	L							E	E	E	E					L	L	L	L
23	N	L	N					L	L	L				E	E	D					D	E	N	L				
24	L	N	L					E	E	L					E	E	D				E	D	L	L				
25	E	E	D	E	E	E	E				E	D	D	E	L	L	L						E	E	E	E	E	D
26																												
27	E	D	E	E	E	E	E	E	L																			
28	D	E	E	E	E	E	D																L	N	L	N	N	L
29											E	E	E	DH	D	L												

Tabela A.3: Instância medium_hidden04 - LSA: 80 / LS: 78

Enf.	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
0	E	E	D	E			E	E	E	D	L			E	D	E	E	L			E	D	E	E	L			E
1	E	D	D	L			E	E	D	D	L			E	E	L	N	L			D	E	E	E	E			E
2	E	D	E	E			D	D	E	E	E			E	N	N	L	L			D	D	E	E	D			N
3				D	E	E	E	D			N	N	N	L	L			E	E	E			D	D	E	E	D	
4	E	D	D	L			D	L	L	N	L			N	L	N	N	L			L	N	L	L	L			L
5	N	N	L	L			D	D	E	E	L			D	E	L	L	L			L	N	L	L	L			N
6	N	N	N	L			E	E	E	E	E			E	D	E	E	E			N	N	L	N	L			D
7	D	E	E	L			N	L	N	N	L			E	D	D	E	E	D	D	E	E	E	E	E			D
8				N	L	L	N	L		L	N	L	L	L	L		E	D	D	D	L			N	N	L	L	L
9				D	D	D	E	E	L		D	E	E	N	L			N	L	L	L				D	D	D	E
10				E	D	D	L				N	L	L	N	L	N	N	L			L	L	L	N	L			
11				N	N	N	N	L	L	L	L			L	N	L	N				E	N	L	L	L			
12				N	N	N	N						D	L	L	N	L				L	L	N	N				
13				E	E	E	L				DH	DH	D	DH	DH	DH		DH	DH	DH	DH	DH	E	DH				
14				E	E	E	L				N	N	N	N	N	L	L				N	N	L					
15	E	E	E	E					D	E	E	D	D					N	L	L	N	L	N	L				E
16	L	L	L					E	D	E	E						E	E	E	E				D	D	D	E
17	L	L	L					E	E	D					E	E	E	D							E	N	N	L
18	L	N	L					L	N	N				L	N	L					E	E	E	E				
19	N	L	N				L	N	N	N				D	E	E					E	E	D	D				
20	L	L	L					L	L	L						D	D	D	D	D					N	L	L	L
21	D	E	E					N	L	L	L						D	E	L	L					E	E	E	E
22	N	L	L				E	N	N	L				D	E	E	DH				D	E	N	L				
23	E	E	E					N	L	L						D	L	N	N	N					E	E	L	
24	L	L	N	L					E	E	D	E	E					N	N	N	N	L	N					N
25											E	E	E	E	E	E	D	E	E	E								
26	L	N	N	N	L	L	L																L	L	N	N	N	N
27	D	E	E	D	L	L	L															D	D	E	N	L	L	
28											E	L	L	L	L	L	N	L										
29	DH	DH	DH	DH	DH	DH	DH	DH	DH	DH												E	DH	D	DH	DH	DH	DH

Tabela A.4: Instância sprint_late04 - LSA: 75 / LS: 73

Enf.	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
0	N	N	N	N		E	N	N	N	N			E	E	E			E	E	E	L				D	E	E	E
1	E	E	E	L	L			E	D	D			E	N	N	N	N			E	D	L			N	N	N	N
2				D	E	E	E	L			E	L	L	L	L			D	D	E	N				E	E	E	D
3				E	E	E	L	L			N	N	N		E	E	E	L				E	E	N	N		D	E	E
4	D	D	D	L	L			E	E	E	L			E	L	L	L			L	L	L			L	L	L	L
5	E	L	L			N	L				E	E	D					E	E		D	D	D	D	E			
6	L			E	N	L					D	D	L	L				L	L		N	L	L	L	L			
7	L			E	D	L					L	L			D	D	D			D	E	E	E	E				
8						D	D				E	E						E	L	L							D	L
9							E	D	L	L			E	D				N	N	N					E	L	L	

Tabela A.5: Instância sprint_hidden08 - LSA: 209 / LS: 204

Enf.	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
0	E	E			E	E	E	E	D			D	D	L				E	D	D	E	E			E	N	N	N
1	N	N	N					E	N	N	N			D	D	D	E	E			D	L	L			D	D	E
2	L	L	L	L			L	L	L	L	L			N	L	L	L	L			E	E	D	D	E			
3			D	E	N	N	N			D	E	N	N	L			L	L	L	L	L			L	L	L	L	L
4	L	L	L	L			E	D	E	E	L			E	L	L			N	N	L			E	D	E	E	L
5	E	E	E					L				E	E	E					E	E				E	N			
6				D	L	L				E	E				E	E	D	D				D	E	E				
7	D	D	E	N					E	L					E	E	N					L	L				E	
8				E	D	D	D				D	L	L				E	N					E	L	L			
9								N	L						N	N					N	N	N					D

Referências Bibliográficas

Andersen, K., Cornuejols, G. and Y., L.: 2005, Reduce-and-split cuts: Improving the performance of mixed integer gomory cuts, *Management Science* **51**, 1720–1732.

Andreello, G., Caprara, A. and Fischetti, M.: 2007, Embedding cuts in a branch and cut framework: a computational study with 0,1/2-cuts, *INFORMS Journal on Computing 1* **19**(2), 229–238.

Atamtürk, A., Nemhauser, G. L. and Savelsbergh, M. W. P.: 2000, Conflict graphs in solving integer programming problems, *European Journal of Operational Research* **121**, 40–55.

Avella, P. and Vasil'ev, I.: 2005, A computational study of a cutting plane algorithm for university course timetabling, *Journal of Scheduling* **8**, 497–514.

Balas, E., Ceria, S., Cornueljols, G. and Natra, N.: 1996, Gomory cuts revisited, *Operations Research Letters* **19**, 1–10.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H.: 1998, Branch-and-price: column generation for solving huge integer programs, *Operations Research* **46**, 316–329.

Bilgin, B., Demeester, P., Mısır, M., Vancroonenburg, W., Berghe, G. and Wauters, T.: 2010, A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition, *the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010)-the Nurse Rostering Competition.*

Borndorfer, R.: 1998, *Aspects of Set Packing, Partitioning, and Covering*, PhD thesis, Faculty of Mathematics at Technical University of Berlin.

Boyd, E.: 1992, Fenchel cutting planes for integer programming, *Operations Research* **42**, 53–64.

Boyd, E.: 1994, Solving 0/1 integer programs with enumeration cutting planes, *Annals of Operations Research* **50**, 61–72.

Brito, S. and Santos, H. G.: 2011, Pivoting in the bron-kerbosch algorithm for maximum-weight clique detection (in portuguese), *Anais do XLIII Simposio Brasileiro de Pesquisa Operacional*.

Bron, C. and Kerbosch, J.: 1973, Algorithm 457: finding all cliques of an undirected graph, *Commun. ACM* **16**, 575–577.

Burke, E. and Curtois, T.: 2010, An ejection chain method and a branch and price algorithm applied to the instances of the first international nurse rostering competition, 2010, *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling PATAT 2010*.

Burke, E. K., De Causmaecker, P., Berghe, G. V. and Landeghem, H. V.: 2004, The state of the art of nurse rostering, *Journal of Scheduling* **7**, 441–499.

Burke, E., Li, J. and Qu, R.: 2010, A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems, *European Journal of Operational Research* **203**(2), 484–493.

Burke, E., Mareček, K., Parkes, A. J. and Rudová, H.: 2011, A branch-and-cut procedure for the udine course timetabling problem, *Ann. Oper. Res.* pp. 1–17.

Cheang, B., Li, H., Lim, A. and Rodrigues, B.: 2003, Nurse rostering problems–a bibliographic survey, *European Journal of Operational Research* **151**(3), 447–460.

Cornuéjols, G.: 2007, Revival of the gomory cuts in the 1990´s, *Annals of Operations Research* **149**(1), 63–66.

Danna, E., Rothberg, E. and Le Pape, C.: 2003a, Exploring relaxation induced neighborhoods to improve mip solutions, *Technical report*, ILOG.

Danna, E., Rothberg, E. and Le Pape, C.: 2003b, Integrating mixed integer programming and local search: A case study on job-shop scheduling problems, *Proceedings CPAIOR'03*.

Dash, S., Goycoolea, M. and Gunluk, O.: 2009, Two step MIR inequalities for mixed-integer programs, *INFORMS Journal on Computing* .

Eso, M.: 1999, *Parallel branch-and-cut for set partitioning*, PhD thesis, Cornell University Ithaca, NY, USA.

Fischetti, M. and Lodi, A.: 2003, Local branching, *Mathematical Programming* **98**, 23–47.

Fischetti, M. and Lodi, A.: 2007, Optimizing over the first Chvàtal closure, *Mathematical Programming B* **110**(1), 3–20.

Forrest, J. and Lougee-Heimer, R.: 2005, CBC user guide, *INFORMS Tutorials in Operations Research.* pp. 257–277.

Glover, F.: 1996, Ejection chains, reference structures and alternating path methods for traveling salesman problems, *Discrete Applied Mathematics* **65**(1-3), 223–253.

Grotschel, M., Lovasz, L. and Schrijver, A.: 1993, *Geometric Algorithms and Combinatorial Optimization*, Springer.

Hansen, P. and Mladenović, N.: 1997, Variable neighborhood search, *Computers and Operations Research* **24**(11), 1097–1100.

Hansen, P., Mladenović, N. and Urosević, D.: 2006, Variable neighborhood search and local branching, *Comput. Oper. Res.* **33**(10), 3034–3045.

Haspeslagh, S., De Causmaecker, P., Stolevik, M. and A., S.: 2010, First international nurse rostering competition 2010, *Technical report*, CODeS, Department of Computer Science, KULeuven Campus Kortrijk. Belgium.

Hoffman, K. and Padberg, M.: 1993, Solving airline crew scheduling problems by branch-and-cut, *Management Science* **39**(6), 657–682.

IBM: 2011, *CPLEX 12.2 User's Manual*.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R., Danna, E., Gamrath, G., Gleixner, A., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. and Wolter, K.: 2011, Miplib 2010, *Mathematical Programming Computation* **3**, 103–163. 10.1007/s12532-011-0025-9.

Linderoth, J. T. and Ralphs, T. K.: 2005, Noncommercial software for mixed-integer linear programming, *in* J. Karlof (ed.), *Integer Programming: Theory and Practice*, Vol. 3 of *Operations Research Series*.

Lougee-Heimer, R.: 2003, The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community, *IBM Journal of Research and Development* **47**(1), 57–66.

Martins, A. X., Souza, M. C., Souza, M. J. and Toffolo, T. A. M.: 2009, GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem, *Journal of Heuristics* **15**, 133–151.

Méndez-Díaz, I. and Zabala, P.: 2008, A cutting plane algorithm for graph coloring, *Discrete Applied Mathematics* **156**, 159–179.

Mittelmann, H.: 2012, Benchmarks for optimization software.

Mladenovic, N. and Hansen, P.: 1997, Variable neighborhood search, *Computers & Operations Research* **24**(11), 1097–1100.

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

Padberg, M.: 1973, On the facial structure of set packing polyhedra, *Mathematical Programming* **5**(1), 199–215.

Pigatti, A., Poggi de Aragão, M. and Uchoa, E.: 2005, Stabilized branch-and-cut-and-price for the generalized assigment problem, *2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics, 2005, Angra dos Reis*, Vol. 19, Elsevier Science, Amsterdam, Holanda, pp. 385–395.

Poggi de Aragão, M. and Uchoa, E.: 2003, Integer program reformulation for robust branch-and-cut-and-price, *Annals of Mathematical Programming in Rio*, Buzios, Brazil, pp. 56–61.

Ralphs, T. K. and Guzelsoy, M.: 2005, The symphony callable library for mixed integer programming, *in* B. Golden, S. Raghavan, E. Wasil, R. Sharda and S. Voa (eds), *The Next Wave in Computing, Optimization, and Decision Technologies*, Vol. 29 of *Operations Research/Computer Science Interfaces Series*, Springer US, pp. 61–76.

Ralphs, T., Saltzman, M. and Ladnyi, L.: 2004, The COIN-OR Open Solver Interface: Technology Overview.

Rothberg, E.: 2007, An evolutionary algorithm for polishing mixed integer programming solutions, *INFORMS JOURNAL ON COMPUTING* **19**, 534–541.

Santos, H. G., Uchoa, E., Ochi, L. and Maculan, N.: 2012, Strong bounds with cut and column generation forÂ class-teacher timetabling, *Annals of Operations Research* **194**, 399–412.

Uchoa, E., Toffolo, T. A. M., de Souza, M. C., Martins, A. X. and Fukasawa, R.: 2012, Branch-and-cut and hybrid local search for the multi-level capacitated minimum spanning tree problem, *Networks* **59**(1), 148–160.

Valouxis, C., Gogos, C., Goulas, G., Alefragis, P. and Housos, E.: 2012, A systematic two phase approach for the nurse rostering problem, *European Journal of Operational Research* .