Heurísticas e Programação Inteira para o Problema das Enfermeiras

Haroldo Gambini Santos[1], Túlio A. M. Toffolo[2], Rafael A. M. Gomes[3], Rafael Henrique Vareto[4]

[1] Prof. Adjunto - Departamento de Computação
Universidade Federal de Ouro Preto(UFOP)
Ouro Preto, MG

[2]Prof. Assistente - Departamento de Computação
Universidade Federal de Ouro Preto(UFOP)
Ouro Preto, MG

[3]Mestrando em Ciência da Computação
Universidade Federal de Ouro Preto(UFOP)
Ouro Preto, MG

[4]Graduando em Ciência da Computação
Universidade Federal de Ouro Preto(UFOP)
Ouro Preto, MG

haroldo.santos@gmail.com, tulio@toffolo.com.br,

rafael.amgomes@gmail.com, rafaelvareto@gmail.com

RESUMO

Este trabalho apresenta detalhes do estudo de heurísticas e Programação Inteira para o clássico Problema de Escalonamento de enfermeiras, o qual leva em consideração a distribuição das Enfermeiras em turnos em um determinado período. Como os serviços de enfermagem possuem um dos maiores componentes de custo nos orçamentos de hospitais, é essencial que se desenvolva uma boa política de planejamento priorizando a distribuição da carga horária da equipe de enfermagem o mais uniformemente possível antendendo ao máximo as preferências particulares. Experimentos computacionais com as técnicas apresentadas nesta abordagem produziram resultados interessantes como a melhoria de diversas soluções conhecidas para as instâncias apresentadas.

KEYWORDS Escalonamento de enfermeiras, Programação Inteira, Heurísticas

ABSTRACT

This work presents the study of mathematical programming heuristics for a variant of the classical Nurse Rostering Problem, which considers the work distribution in shifts in a given time period. Since nursing services cost represents are very significant in a hospital budget it is essential to have a solid planning strategy. Specifically, it is important to provide an even distribution of the workload and to satisfy, whenever it's possible, personal preferences.

KEYWORDS Nurse Rostering, Integer Programming, Heuristics.

1 Introdução

Existe diversas pesquisas voltadas para soluções computacionais do problema de Escalonamento de Enfermeiras [2]. 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)* [9]. 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 serão apresentadas heurísticas de Programação Inteira para o clássico Problema de Escalonamento das Enfermeiras (NRP - Nurse Rostering Problem).

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 próxima 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 serão apresentadas técnicas para acelerar a produção de soluções viáveis próximas da otimalidade. Estas técnicas por último mencionadas, além de serem o foco deste trabalho, podem ser utilizadas individualmente por aqueles interessados no uso desta formulação em aplicaçõees reais.

Os experimentos computacionais mostraram que métodos rápidos na produção de boas soluções primais e duais podem ser obtidas usando as técnicas propostas: os algoritmos propostos não são somente heurísticas bem competitivas mas também provou-se a otimalidade para a grande maioria das instâncias da INRC e melhoramos o melhor resultado conhecido de outras, em torno de 15% delas.

2 O Problema do 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[3]. 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. Abrangentemente falando, este conjunto pode ter qualquer número de turnos, mas no caso da INRC e na maioria dos casos práticos uma enfermeira só pode trabalhar um turno por dia o que geralmente inclui o turno Diurno (D, morning), turno Vespertino (E, evening), turno Noturno (N, night), Folga (-, day-off) dentre outros. A Tabela 1 seguinte apresenta parte de uma escala trabalhista a qual indica os turnos atribuídos às enfermeiras, em uma visão enfermeira-dia.

Tabela 1. Exemplo de solução em uma visão enfermeira-dia

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

Neste trabalho, a abordagem proposta baseou-se no problema definido pela International Nurse Rostering Competition, 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 destacam-se as propostas finalistas. Mais detalhes sobre as abordagens podem ser consultadas no site da Competição Internacional[1].

Valouxis et al.[14], vencedores do desafio, desenvolveram uma estratégia de duas fases onde na primeira fase a carga de trabalho para cada enfermeira e para cada dia da semana era decidida enquanto na segunda fase os turnos diários específicos eram atribuídos. Como a competição exigiu requisitos de qualidade e restrições de tempo, classificaram as instâncias do problema em subproblemas de tamanho computacionalmente gerenciáveis e foram depois resolvidos sequencialmente utilizando Programação Matemática Inteira. Aplicaram técnicas de otimização local para a busca entre combinações parciais de escala das Enfermeiras. Esta sequência é repetida diversas vezes dependendo do tempo computacional disponível.

Burke and Curtois [1] aplicaram um método baseado na cadeia de interferência/ejeção para as instâncias da categoria $sprint$ e o algoritmo $branch - and - price$ para as instâncias das categorias *medium* e *long*. 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 [1], foi utilizado.

2.1 Restrições

O problema de alocação das enfermeiras envolve a atribuição de turnos às enfermeiras levando em conta várias 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 forem possível, mas para as quais sabem-se que nem todas poderão ser atendidas;

Neste trabalho, nós consideramos 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;

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

- 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 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 qualidades do que a mesma possui.

3 Melhoramento Primal: Heurísticas MIP

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 [11, 13]. Um termo difusivo 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 fraca de variáveis, assim como no método *Local Branching* e outros similares[6, 8], ou com fixações fortes de variáveis presentes no *Relaxation Induced Neighborhood Search (RINS)*[5]. Este último mencionado apresentou melhores resultados em testes utilizando instâncias MIPLIB[10].

Antes de apresentar a heurística MIP desenvolvida, será apresentado um procedimento simples que constrói soluções factíveis que serão utilizadas como formação dos experimentos.

3.1 Um Algoritmo Construtivo 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 (day) d e enfermeira (nurse) n, a demanda $\tilde{r}_{ds}$ é satisfeita pela seleção, uma a uma, de um turno (shift) n sendo que esta alocação desencadeia o menor aumento na solução 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 seja alocada. O algoritmo tem complexidade de $O(|N|^2 \times |D|)$.

3.2 Estrutura das Vizinhanças

A fase de busca local explora o espaço de busca através de diversas vizinhanças, utilizando um esquema inpirado em Descida em Vizinhança Variável (*Variable Neighborhood Descent - VND*) [7]. Considerando os resultados obtidos com os usos recentes de heurísticas *RINS* [4], foram propostas duas diferentes estruturas de vizinhança que são baseadas na resolução de partições pequenas do problema original para otimalidade. As diferenças entre as vizinhanças estão nas regras consideradas que geram tais subproblemas.

3.2.1 Estrutura da Vizinhança de Fixação de Dias

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

Antes de se iniciar as iterações de fixação das enfermeiras que trabalham em determinado dia, é realizada a fixação dos $|D| - ndays$ dias com menores custos, ou seja, os $ndays$ dias com maiores custos na função objetivo são deixados livres (os dias restantes são fixados) e resolvidos pela otimalidade. A todo o momento que a solução é melhorada, S é atualizada, quando não

houver mais melhorias na solução S, inicia-se a primeira iteração e assim por diante. Este processo se repete cada vez que $ndays$ é incrementada, respeitando sempre o número total $|D|$ de dias.

Na primeira iteração ($iter = 0$), um subproblema é criado a partir da solução S, fixando toda alocação de enfermeiras, exceto aquelas no intervalo de dias entre 1 e $ndays$. O subproblema é então resolvido para otimalidade. Se a solução é aprimorada, S é atualizada. Na próxima iteração, outro subproblema é gerado pela fixação de todas as alocações de enfermeiras, exceto aquelas entre day_A e day_B (equações 1 e 2):

$$day_A = 1 + (iter \times step) \quad (1)$$

$$day_B = ndays + (iter \times step) \quad (2)$$

As equações 1 e 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 day_A ou day_B é maior que o número de dias, $|D|$, foi considerado 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.

Existem três diferentes maneiras de selecionar uma janela. Primeiramente existe a janela contínua, exatamente aquela descrita anteriormente e que é possível ser visualizada na figura 1. Nesta, a janela não possui lacunas, ou seja, não existe nenhum dia entre day_A e day_B que não pertença à janela.

Outra janela seria aquela que contempla somente números pares, ou seja, existem lacunas na janela (as lacunas são os dias ímpares). A estrutura de deslocamento se mantém a mesma, ou seja, uma janela com $ndays = 5$ começando no dia 1 do mês teria os seguintes dias: $2, 4, 6, 8, 10$; caso o parâmetro $step$ seja 3, a próxima janela será: $8, 10, 12, 14, 16$. O terceiro e último tipo de janela existente neste método é muito semelhante à janela citada anteriormente, mas ao contrário de serem escolhidos os dias pares, são selecionados somente os dias ímpares.

A figura 1 mostra como a busca no espaço de soluções é realizada pela estrutura de vizinhança de fixação dos dias.

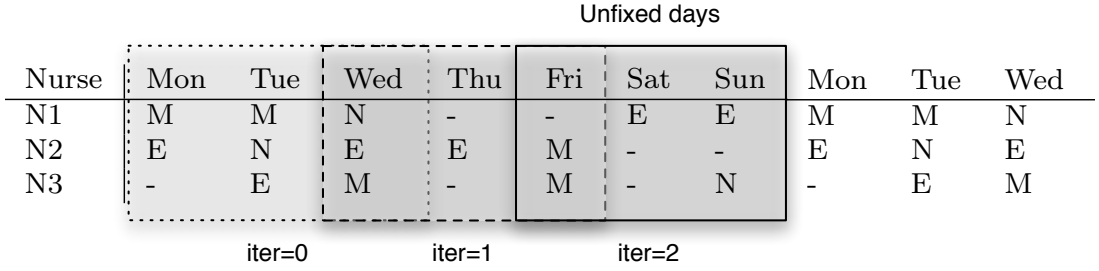

Figura 1. *Fixação dos Dias* de uma vizinhança com $ndays = 3$ e $step = 2$.

As vizinhanças tem dois parâmetros, $step$ e $ndays$. 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. Já o outro parâmetro, $ndays$,

define o tamanho de cada subproblema e define se o mesmo também é crítico. Valores muito pequenos podem criar subproblemas que não possuem nenhuma solução melhor e valores muito grandes podem criar subproblemas intratáveis.

3.2.2 Estrutura de Vizinhança de Fixação de Enfermeiras

Nesta estrutura de vizinhança, que *Fixa as Enfermeiras*, os subproblemas são criados pela fixação das alocações de $|N| - nnurses$, onde $nnurses$ também é um parâmetro da vizinhança, assim como $ndays$. $|N|$ representa o conjunto de todos as enfermeiras e $nnurses$ engloba somente um subconjunto variável de $|N|$.

Na primeira iteração ($iter = 0$), o subproblema também é inicialmente criado a partir da solução S fixando a alocação das enfermeiras, exceto daquelas no intervalo de enfermeiras entre 1 e $nnurses$. Este subproblema é resolvido para otimalidade, e se a solução é melhorada, S é atualizada. Na iteração seguinte, o próximo subproblema é gerado pela fixação das enfermeiras, exceto aquelas entre $nurse_A$ e $nurse_B$ (equações 3 e 4):

$$nurse_A = 1 + (iter \times step) \tag{3}$$

$$nurse_B = nnurses + (iter \times step) \tag{4}$$

As equações 3 e 4 calculam, respectivamente, o início e o fim de uma janela na qual as enfermeiras pertencentes a ela permanecem não fixadas. 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 $nurse_A$ ou $nurse_B$ é maior que o número de enfermeiras, $|N|$, foi considerada a enfermeira para ser a sobra deste valor dividido por $|N|$. O algoritmo procede até $|N|/step$ iterações consecutivas sem que qualquer melhora seja alcançada, o que indica que um ótimo local para a vizinhança foi encontrado.

Assim como na fixação dos dias, antes de começar as iterações, é realizada a fixação das $|N| - nnurses$ enfermeiras com menores custos, ou seja, as $nnurses$ enfermeiras com maiores custos na função objetivo são deixadas livres (as outras enfermeiras são fixadas) e resolvidas pela otimalidade. Consequentemente, a todo o momento que a solução é melhorada, S é atualizada. Quando não houver mais aprimoramentos na solução S, inicia-se a primeira iteração e assim por diante. Este processo se repete cada vez que $nnurses$ é incrementada, respeitando sempre o número total $|N|$ de enfermeiras.

Existem três diferentes maneiras de selecionar uma janela. Primeiramente existe a janela contínua, exatamente aquela descrita anteriormente na estrutura de fixação de dias e que é possível ser visualizada na figura 1. Nesta, a janela não possui lacunas, ou seja, não existe nenhuma enfermeira que esteja entre $nurse_A$ e $nurse_B$ que não pertença à janela. Também há a seleção das enfermeiras pares ou das enfermeiras ímpares.

Semelhante à fixação de dias, a vizinhança das enfermeiras tem dois parâmetros, $step$ e $nnurses$. 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. Já o outro parâmetro, $nnurses$, define o tamanho de cada subproblema e define se o mesmo também é crítico. Valores muito pequenos podem criar subproblemas que não possuem nenhuma solução melhor e valores muito grandes podem criar subproblemas intratáveis.

3.2.3 Estrutura de Vizinhança de Fixação Mista

Esta heurísitca talvez seja a mais complexa desta abordagem, pois nela ocorre a fixação simultânea de um subconjunto de dias ($ndays$) e enfermeiras ($nnurses$). Em outras palavras, os subproblemas são gerados pela fixação de todas as alocações de enfermeiras que trabalham em determinados dias do mês e/ou enfermeiras específicas de um grupo (janela).

Semelhante a maioria dos métodos descritos nesta seção, antes que se iniciem as iterações de fixação simultânea e variável dos dias e enfermeiras, é feita a fixação dos $|D| - ndays$ dias com menores custos e também a fixação das $|N| - nnurses$ enfermeiras com custo mais baixo. Isso significa que os $ndays$ dias e as $nnurses$ enfermeiras com piores custos na função objetivo são deixadas livres (e todo o restante é fixado) e resolvidos pela otimalidade. A todo o momento que a solução é melhorada, S é atualizada. Quando não houver mais aprimoramentos na solução S, inicia-se a primeira iteração e assim por diante. Este processo se repete cada vez que $ndays$ e $nnurses$ são incrementados.

Na primeira iteração ($iter = 0$), um subproblema é gerado a partir da solução S fixando todas as alocações exceto aquelas que se encontram no intervalo entre 1 e $ndays$ e entre 1 e $nnurses$. Deixando estas alocações livres, o subproblema é resolvido pela otimalidade. Assim como nos métodos anteriores, se a solução é melhorada, S consequentemente é atualizada. Nas iterações seguintes, outros subproblemas são gerados pelas fixações das alocações, exceto aquelas entre day_A, day_B, $nurse_A$ e $nurse_B$ (equações 1 , 2, 3 e 4).

As equações 1, 2, 3 e 4 calculam, respectivamente, o início e o fim de uma janela na qual as alocações pertencentes a ela permanecem não fixadas. Nestas equações, $iter$ é o número de iterações correntes e $stepX$ é um parâmetro que define um número de alocações entre dois subproblemas consecutivos.

Ao contrário das anteriores, esta vizinhança possui quatro parâmetros: $stepD$, $stepN$, $ndays$ e $nnurses$. $stepD$ representa o número de dias e $stepN$ o número de enfermeiras entre dois subproblemas consecutivos. Quanto menor o valor, maior é o número de diferentes subproblemas na vizinhança. Já os outros parâmetros, $ndays$ e $nnurses$ definem o tamanho de cada subproblema e determina se o mesmo também é crítico. Valores muito pequenos podem criar subproblemas que não possuem nenhuma solução melhor e valores muito grandes podem criar subproblemas intratáveis.

3.3 Heurística de Programação Matemática

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. Foi decidido utilizar apenas as seguintes estruturas no algoritmo-exemplo:

$\mathcal{N}_1^m$: Estrutura de vizinhança de *Fixação dos Dias* com $ndays = 2m$ e $step = m$;
$\mathcal{N}_2$: Estrutura de vizinhança de *Fixação das Enfermeiras*;

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 1 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 por 1, 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.

A Figura 2 mostra o pseudo-código da heurística proposta. Nesta figura, os processos $\mathcal{N}_1^m(S_0)$ 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 solução S_0 é retornada.

Require: S_0, m
1: $S^* \leftarrow \mathcal{N}_1^m(S_0)$
2: $S^* \leftarrow \mathcal{N}_2(S^*)$
3: $m = m + 1$
4: **while** $m \leq |D|/2$ e tempo limite não é alcançado **do**
5: $S \leftarrow \mathcal{N}_1^m(S^*)$
6: **if** S é uma solução melhor que S^* **then**
7: $S^* \leftarrow \mathcal{N}_2(S)$
8: **end if**
9: $m = m + 1$
10: **end while**
11: **return** S^*;

Figura 2. Pseudo-código do Algoritmo HPM

Se uma vizinhança é capaz de melhorar a solução, VND volta para a primeira vizinhança, reiniciando a busca. O algoritmo proposto sempre incrementa o valor de m. Este método foi adotado após constatar que executar a busca na vizinhança $\mathcal{N}_1^m$, a solução dificilmente é melhorada, porém possui um alto custo computacional.

4 Experimentos Computacionais

O código apresentado nesta abordagem foi escrito em C++ usando as bibliotecas do CPLEX. Esta abordagem permitiu integrar as rotinas com os resolvedores através da Open Solver Interface (OSI) [12]. O código foi compilado no GCC/g++ versão 4.6. Foram executados todos os experimentos em computadores Intel Core i7 3.4GHz com 16Gb de memória RAM e com Sistema Operacional Ubuntu 10.10 64-bits. Foi utilizada o resolvedor CPLEX versão 12.2.0.

4.1 Melhores resultados

Os melhores resultados alcançados em todos os experimentos estão apresentados na Tabela seguinte. As células da tabela marcadas com * indicam alguma melhora sobre a melhor solução conhecida do problema, disponibilizada na página da INRC durante o período da compilação deste trabalho. É importante ressaltar que este website vem recebendo atualizações constantes nos anos posteriores à competição, ou seja, as melhores soluções anteriores também foram difíceis de serem encontradas. Os valores em destaque (negrito), informam que a otimalidade foi provada pela formulação de programação inteira utilizada.

Instância			MSC	Solução Inicial - Cojunto G			Solução Inicial - Conjunto N			Optimal?
				Inic. Obj.	Melhor Obj.	Time	Inic. Obj.	Melhor Obj.	Tempo	
long	early	01	197	354	**197**	106	1532	**197**	13	yes
		02	219	384	**219**	128	1661	**219**	220	yes
		03	240	364	**240**	22	1579	**240**	10	yes
		04	303	475	**303**	12	1582	**303**	13	yes
		05	284	442	**284**	100	1627	**284**	13	yes
	hidden	01	363	926	347	310	6505	***346**	339	no
		02	90	155	***89**	157	1361	***89**	397	no
		03	38	228	42	84	1456	45	226	no
		04	22	188	**22**	166	1309	**22**	156	no
		05	41	361	***41**	308	1362	44	241	yes
	late	01	235	557	239	595	6260	248	264	no
		02	229	586	238	349	6204	236	489	no
		03	220	685	**220**	187	5423	**220**	442	no
		04	222	751	224	188	5367	233	353	no
		05	83	568	**83**	215	6534	86	394	yes
medium	early	01	240	372	**240**	36	884	**240**	18	yes
		02	240	378	**240**	24	900	**240**	25	yes
		03	236	368	**236**	33	906	**236**	53	yes
		04	237	389	**237**	36	894	**237**	69	yes
		05	303	397	**303**	82	938	**303**	43	yes
	hidden	01	130	728	121	132	4139	***111**	281	no
		02	221	802	223	442	3289	223	560	no
		03	34	182	36	419	838	38	491	no
		04	81	310	82	226	741	***81**	232	no
		05	122	638	***119**	89	3336	120	358	no
	late	01	158	856	162	406	3524	162	528	no
		02	18	132	20	456	945	19	329	no
		03	29	194	**29**	182	771	**29**	359	yes
		04	35	143	36	401	780	**35**	382	yes
		05	107	693	112	438	2979	108	448	no
sprint	early	01	56	92	**56**	1	289	**56**	17	yes
		02	58	89	**58**	7	293	**58**	25	yes
		03	51	81	**51**	1	295	**51**	10	yes
		04	59	99	**59**	11	303	**59**	10	yes
		05	58	92	**58**	1	289	**58**	6	yes
		06	54	95	**54**	2	293	**54**	9	yes
		07	56	95	**56**	8	295	**56**	28	yes
		08	56	88	**56**	1	307	**56**	2	yes
		09	55	85	**55**	12	295	**55**	8	yes
		10	52	89	**52**	1	303	**52**	23	yes
	hidden	01	33	83	***32**	23	355	***32**	56	yes
		02	32	72	**32**	12	269	**32**	16	yes
		03	62	116	**62**	32	351	**62**	41	yes
		04	67	108	***66**	14	380	***66**	37	yes
		05	59	102	**59**	49	300	**59**	10	yes
		06	134	330	***130**	19	2176	***130**	43	yes
		07	153	424	**153**	59	1627	**153**	20	yes
		08	209	442	***204**	24	1956	***204**	39	yes
		09	338	513	**338**	14	2368	**338**	40	yes
		10	306	503	**306**	28	1850	**306**	10	yes
	late	01	37	83	**37**	49	304	**37**	63	yes
		02	42	77	**42**	55	299	**42**	25	yes
		03	48	86	**48**	56	300	**48**	51	yes
		04	75	209	***73**	135	1187	***73**	52	yes
		05	44	75	**44**	54	323	**44**	23	yes
		06	42	75	**42**	1	319	**42**	12	yes
		07	42	129	**42**	21	1320	**42**	41	yes
		08	17	105	**17**	5	1291	**17**	14	yes
		09	17	83	**17**	7	1291	**17**	18	yes
		10	43	172	**43**	26	1261	**43**	22	yes

* MSC Melhorado

5 Conclusões e Trabalhos Futuros

O grande objetivo desta abordagem é analisar os mais diferentes métodos relacionados à Programação de Horários existentes na literatura. Existe um grande problema na área da saúde quando a questão é o Escalonamento do quadro funcional de enfermagem. Como ponto inicial do projeto, foi incorporado um algoritmo construtivo para a elaboração inicial das escalas. Ao construir as escalas, o algoritmo é executado enquanto houver enfermeiras disponíveis para serem alocadas em diferentes turnos. Inicialmente, o mais importante é que todas as restrições rígidas sejam atendidas, o que já fornece uma solução factível, porém provavelmente bem distante de uma solução ótima, pois a mesma é avaliada de acordo com o número de restrições flexíveis satisfeitas. Após a solução inicial, inicia-se o refinamento através das heurísticas detalhadas nesta abordagem, onde as vizinhanças são visitadas em busca de ótimos locais. A heurística proposta foi construída sobre uma formulação de Programação Inteira e avaliada com o solver CPLEX. Houve melhorias em várias soluções até então conhecidas, exigindo tempos de computação muito curtos e ainda serem competitivos com as melhores heurísticas para este problema. Acredita-se que os novos e melhorados limites primais e duais permitirá uma avaliação mais precisa da qualidade das heurísticas disponíveis para este problema.

Referências

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

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

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

[4] Danna, E., Rothberg, E., Le Pape, C.: Exploring relaxation induced neighborhoods to improve mip solutions. Tech. rep., ILOG (2003)

[5] Danna, E., Rothberg, E., Le Pape, C.: Integrating mixed integer programming and local search: A case study on job-shop scheduling problems. In: Proceedings CPAIOR'03 (2003)

[6] Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98**, 23–47 (2003)

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

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

[9] Haspeslagh, S., De Causmaecker, P., Stolevik, M., A., S.: First international nurse rostering competition 2010. Tech. rep., CODeS, Department of Computer Science, KULeuven Campus Kortrijk. Belgium (2010)

[10] 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., Wolter, K.: MIPLIB 2010. Mathematical Programming Computation **3**, 103–163 (2011). URL http://dx.doi.org/10.1007/s12532-011-0025-9. 10.1007/s12532-011-0025-9

[11] Martins, A.X., Souza, M.C., Souza, M.J., Toffolo, T.A.M.: GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem. Journal of Heuristics **15**, 133–151 (2009). DOI 10.1007/s10732-008-9079-x. URL http://dl.acm.org/citation.cfm?id=1527562.1527566

[12] Ralphs, T., Saltzman, M., Ladnyi, L.: The COIN-OR Open Solver Interface: Technology Overview (2004). URL http://www.coin-or.org/Presentations/CORS2004-OSI.pdf

[13] Uchoa, E., Toffolo, T.A.M., de Souza, M.C., Martins, A.X., Fukasawa, R.: Branch-and-cut and hybrid local search for the multi-level capacitated minimum spanning tree problem. Networks **59**(1), 148–160 (2012). DOI 10.1002/net.20485. URL http://dx.doi.org/10.1002/net.20485

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