Vizinhança de Grande Porte para o Problema de Agendamento de Cirurgias Eletivas em Hospitais da Rede Pública e Privada

Giselle Paranhos de Andrade[1], **Sérgio Ricardo de Souza**[1], **Moacir Felizardo de França Filho**[1]

[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675 - Nova Gameleira- CEP 30510-000 -
Belo Horizonte, MG, Brasil

{giselle, sergio, franca}@dppg.cefetmg.br

Resumo. *Neste artigo é tratado o Problema de Agendamento de Cirurgias Eletivas (PACE). O presente trabalho pretende constituir uma contribuição da pesquisa científica, na área da Pesquisa Operacional, para o aumento da eficiência da capacidade instalada da oferta de cirurgia de hospitais de atendimento geral, contribuindo assim para equilibrar as listas de espera para cirurgia, e tornar o bloco cirúrgico um serviço mais eficiente e rentável. O PACE será tratado como um Problema de Programação em Máquinas Paralelas Idênticas. Através da análise de vários experimentos, para instâncias com até 836 tarefas, a metaheuristica Iterated Greedy (IG) apresentou um melhor desempenho quando comparado com outras três metaheurísticas.*

1. Introdução

O problema das listas de espera para cirurgia nos hospitais constitui uma das questões mais importantes e que precisa, necessariamente, de uma solução urgente.

O aumento das listas de espera para cirurgia está relacionado com importantes avanços tecnológicos cirúrgicos e anestésicos, verificados ao longo das últimas três décadas. Tais avanços revelaram-se fundamentais no aumento do alcance, da segurança e da eficiência dos procedimentos cirúrgicos oferecidos pelos sistemas de saúde. Por outro lado, o aumento sucessivo do número de pessoas, em lista de espera, para cirurgia resulta, também, de uma incapacidade de resposta do sistema de saúde à procura atual de cirurgias.

Uma unidade de particular interesse é o Centro Cirúrgico (CC), ou melhor, a utilização eficiente das salas cirúrgicas, uma vez que o CC é a unidade responsável pelo maior custo do hospital e, segundo [Macario et al. 1995], é o principal centro de receita da instituição.

Os fatores como lista de espera, agendamento de cirurgias, alocação de salas cirúrgicas e rentabilidade no CC são pesquisados e tratados no problema de Agendamento de Casos Cirúrgicos (Surgical Case Scheduling - SCS), problema que consiste em alocar recursos hospitalares e cirúrgicos para a realização de cirurgias, [Carter and Tovey 1992]. O problema de SCS é considerado, na literatura, um problema clássico de otimização

combinatória, pertencente à classe NP-Difícil, segundo [Carter and Tovey 1992]. Logo, as técnicas heurísticas e as metaheurísticas, de maneira geral, tem sido largamente utilizadas na resolução de problemas desta natureza.

Doravante, denotar-se-á o SCS como PACE (Problema de Agendamento de Cirurgias Eletivas), que consiste em agendar cirurgias previamente conhecidas, desconsiderando casos de cirurgias de emergência. Para a resolução do PACE, é programado um sequenciamento de cirurgias, estabelecendo uma agenda cirúrgica, com o objetivo de minimizar o *makespan*, ou seja, minimizar o horário de término da última cirurgia.

O restante deste trabalho está organizado como segue. Na seção 2 é apresentada a revisão bibliográfica. Na seção 3 desenvolve-se a caracterização do problema. Na seção 4 apresenta-se a metodologia adotada. Na seção 5 apresenta-se as metaheurísticas implementadas. Na seção 6 são expostos os resultados computacionais. E por fim, a seção 7 apresenta a conclusão do trabalho.

2. Revisão Bibliográfica

Segundo [Proença 2010], o processo de planejamento de cirurgias eletivas pode ser dividido em três fases: Planejamento de Casos Mistos *(Case Mix Planning)*; Planejamento Mestre de Cirurgias *(Master Surgery Planning)*; e Agendamento de Casos Eletivos *(Elective Case Scheduling)*.

A fase de Planejamento de Casos Mistos *(Case Mix Planning)* analisa a disponibilidade, em horas das salas cirúrgicas, distribuída pelos diferentes cirurgiões, fase que é realizada anualmente. [Hughes and Soliman 1978], [Robbins and Tuntiwongpiboon 1989] e [Blake and Carter 2002] apresentam diferentes abordagens para esta fase do planejamento.

A fase de Planejamento Mestre de Cirurgias *(Master Surgery Planning)* envolve o desenvolvimento de uma agenda cirúrgica. Trata-se de um documento cíclico, que define o número e o tipo de salas de operações disponíveis, as horas em que as salas estão abertas, definindo, ainda, cirurgiões ou grupos de cirurgias que têm prioridade sobre o tempo das salas cirúrgicas. Esta fase enquadra-se em um nível tático da gestão hospitalar. [Blake and Carter 2002], [Blake and Joan 2002] e [Belien and Demeulemeester 2007] propõem uma série de modelos para a construção de agendamentos de cirurgias para esta fase.

Já na fase de Agendamento de Casos Eletivos *(Elective Case Scheduling)* é estabelecido o agendamento de cada cirurgia em uma base diária. Esta fase situa-se em um nível operacional. Trabalhos relativos a esta fase são encontrados em [Magerlein and Martin 1978], [Przasnyski 1986], [Cardoen et al. 2010], [Ozkarahan 1995], [Gerchak et al. 1996], [Jebali et al. 2006], [Kharrajal et al. 2006].

[Pham and Klinkert 2008] apresentam um modelo em programação linear inteira mista, baseado em uma extensão do *Job Shop Scheduling Problem*, denominada *Multi-Mode Blocking Job Shop*. O modelo define um período de início para cada uma das três fases necessárias na realização de uma cirurgia (pré-operatório, operatório e pós-operatório) e aloca, para cada uma das três fases, um conjunto de recursos necessários. Com técnicas de bloqueio, os autores apresentam uma solução para o problema de restrições de disponibilidade dos equipamentos. É possível, através do algoritmo apresentado, bloquear

os recursos indisponíveis no momento da cirurgia. O objetivo é minimizar o instante de término da última cirurgia.

3. Caracterização do Problema de Agendamento de Cirurgias Eletivas

- Uma abordagem via *Scheduling*

Nesta seção caracteriza-se o PACE como sendo um problema de Programação em Máquinas Paralelas Idênticas com Tempos de Preparação de Máquina dependente da Sequência *(Identical Parallel Machine Scheduling Problem with Sequence Dependent Setup Times)*, este é uma classe particular de problemas de sequenciamento *(Scheduling Problem)*. Logo, será utilizada equivalência entre máquina e sala; e tarefa e cirurgia.

No Problema estudado, tem-se um conjunto $N = \{1, ...n\}$ de n tarefas e um conjunto $M = \{1, ..., m\}$ de m máquinas idênticas, com as seguintes características: (a) Cada tarefa deve ser processada exatamente uma vez por apenas uma máquina; (b) Cada tarefa j possui um tempo de processamento p_j ; (c) Existem tempos de preparação entre as tarefas, s_{jk}, em que as tarefas j e k serão processadas, nesta ordem. O objetivo é encontrar um sequenciamento das n tarefas nas m máquinas de forma a minimizar o tempo de conclusão do sequenciamento, o chamado *makespan* ou C_{max}. Pelas características citadas, este problema é definido como $P|Sjk|C_{max}$, segundo [Pinedo 2008].

3.1. Exemplo ilustrativo de agendamento de cirurgias eletivas

No primeiro momento, é necessário listar todos os recursos necessários para a realização das cirurgias, como a disponibilidade de recursos humanos (cirurgiões, enfermeiros e anestesistas), além dos recursos materiais (salas cirúrgicas, salas de recuperação pós-anestésica - (RPA) e unidade de tratamento intensivo - (UTI).

A Figura 3.1 mostra, por exemplo, que a cirurgia1 é preparada inicialmente pelo enfermeiro E_3, e iniciada na sala S_2, com a participação do cirurgião C_3, enfermeiro E_2 e anestesista A_2 e com a utilização do equipamento Eq_2, e que após a cirurgia o paciente se recupera na sala RPA.

Os blocos cirúrgicos estudados são utilizados por 14 especialidades médicas e para a constituição da duração média dos tempos de cirurgias, foram utilizados dados reais obtidos dois hospitais privados e dois hospitais da rede pública durante o ano de 2011.

São apresentadas na Tabela 1 as distribuições consideradas neste estudo.

Na Figura 2 é apresentada uma solução inicial de um agendamento semanal, segunda à sexta-feira, entre 07hs e 19hs. O período disponível para o agendamento corresponde a 14 horas diárias, o que totaliza

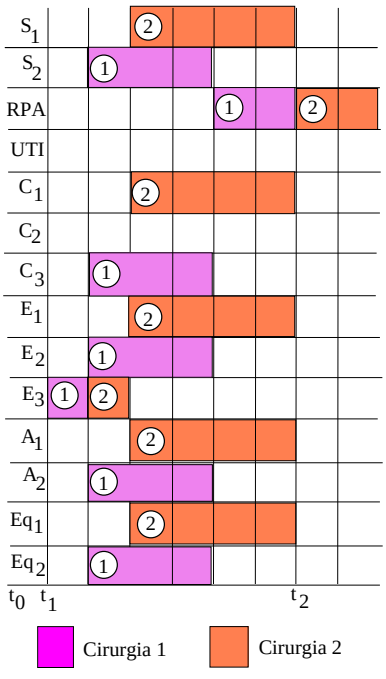

Figura 1. Gráfico de Gantt com duas cirurgias e os recursos necessários

Tabela 1. Distribuição de tempo de cirurgia por especialidade

Especialidade	Demanda	Duração (min)
Cirurgia Plástica	17,8%	120
Cirurgia Geral	16,6%	100
Ortopedia e Traumatologia	13,3%	120
Coloproctologia	12,7%	60
Cirurgia Torácica	9,3%	270
Urologia	6,7%	90
Cirurgia Cardíaca	6,5%	240
Neurocirurgia	5,4%	270
Otorrinolaringologia	4,5%	110
Angiologia	4,1%	80
Mastologista	1,4%	90
Neurologia	0,7%	250
Ginecológica/Obstetrícia	0,3%	90
Dermatologia	0,3%	50

70hs semanais. Estas horas de agendamento são divididas em *slots* de 30min, ou seja, uma cirurgia de 2hs de duração possui 4 *slots*. Considerando que para cada dia são disponibilizados 28 *slots*, tem-se então um total de 140 *slots* para o agendamento semanal de cirurgias. Os *slots* hachurados na cor verde, representam a duração de higienização das salas, que podem variar de um a dois *slots*, de acordo com a duração e especialização da cirurgia. O *makespan* corresponde ao instante de término da última cirurgia, no caso, a cirurgia de número 15 que foi concluída no *slot* 139.

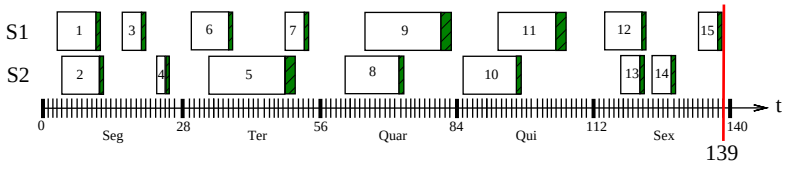

Figura 2. Solução inicial do PACE com 15 cirurgias realizadas em 2 salas e com um *makespan* igual a 139

A Figura 3 apresenta a solução final para o agendamento semanal. Observa-se que todas as cirurgias agendadas para a sala S1 durante a sexta-feira foram realocadas para outros dias da semana; por exemplo, é possível realocar a cirurgia de número 15, agendada para a sala S1 durante a sexta-feira, para a sala S2 na segunda-feira; e as cirurgias 13 e 14, agendadas para sala S2, podem ser agendadas para quarta-feira nesta mesma sala. Com estas realocações, o *makespan* diminui para 106 slots.

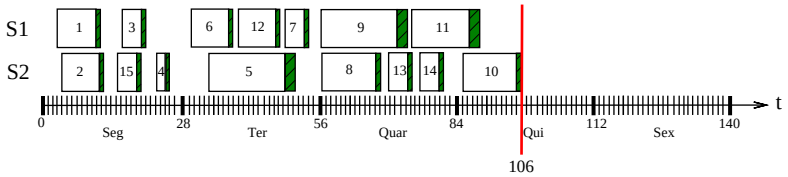

Figura 3. Solução final do PACE após as realocações das cirurgias o *makespan* diminui para 106

4. Metodologia adotada para resolução do problema PACE

Esta seção apresenta os procedimentos propostos para a solução do Problema de Agendamento de Cirurgias Eletivas (PACE). São apresentadas as quatro metaheurísticas implementadas, a saber, *Greedy Randomized Adaptative Search Procedure* (GRASP), *Iterated Local Search* (ILS), *Variable Neighborhood Search* (VNS) e *Iterated Greedy* (IG). Os Algoritmos 5, 6, 7, 8, 9 e 10 mostram os seus pseudocódigos.

4.1. Representação da Solução

A solução é representada através de dois vetores. O primeiro vetor (seq) representa a sequência em que as cirurgias devem ser realizadas, enquanto que o segundo vetor (sala) representa em quais das salas a cirurgia será realizada, visto que cada cirurgia pode ser realizada em mais de uma sala. A Figura 4 representa a solução do problema.

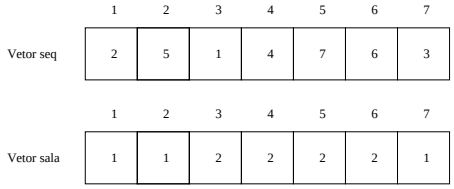

Figura 4. Representação da solução através de dois vetores

4.2. Estruturas de Vizinhança

Neste artigo, definiu-se oito tipos de vizinhanças. São elas:

- **Troca de cirurgias na mesma sala**: troca a posição de duas cirurgias na mesma sala;
- **Realocação de cirurgias na mesma sala**: realoca uma cirurgia para uma nova posição na mesma sala;
- **Troca de cirurgias em salas distintas**: considera-se duas cirurgias e troca as salas que as executam;
- **Realocação de cirurgias em salas distintas**: considera-se duas cirurgias e realoca-la-ás para uma nova posição em outra sala.
- **Troca em bloco de cirurgias na mesma sala**: troca a posição de seis cirurgias na mesma sala;
- **Realocação em bloco de cirurgias na mesma sala**: realoca três cirurgias para três novas posição na mesma sala;
- **Troca em bloco de cirurgias em salas distintas**: considera-se seis cirurgias e troca as salas que as executam;
- **Realocação em bloco de cirurgias em salas distintas**: considera-se seis cirurgias e realoca-as para três novas posições em outra sala.

5. Metaheurísticas Implementadas

5.1. *Greedy Randomized Adaptative Search Procedure (GRASP)*

A metaheurística *Greedy Randomized Adaptative Search Procedure (GRASP)*, proposta por [Feo and Resende 1995], é um processo iterativo no qual cada iteração consiste de duas fases: (i) fase construtiva, que gera soluções factíveis para o problema; e (ii)

fase de busca local, que busca o ótimo local na vizinhança das soluções iniciais, geradas pela fase de construção.

O pseudocódigo do GRASP é mostrado no Algoritmo 5. Na linha 3, uma solução é construída de forma parcialmente gulosa com o método chamado `construcaoGRASP`, que consiste em escolher aleatóriamente a próxima cirurgia a entrar no sequenciamento, dentre uma lista de melhores candidatos. Essa lista é montada de acordo com a regra de maiores tempos de execução. O parâmetro α define o tamanho da lista. Na linha 4 é feito um refinamento na solução através de busca local. A melhor solução é atualizada se houver melhora (linha 5 a 7). Esses passos são repetidos até que um critério de parada seja atendido

```
Entrada: α, criterioParada
1.  início
2.    repita
3.      s ← construcaoGRASP(α);
4.      s' ← buscaLocal(s);
5.      se fo(s') melhor que fo(s*) então
6.        s* ← s';
7.      fim se
8.    até criterioParada seja satisfeito;
9.  fim
```

Figura 5. Algoritmo GRASP

A busca local utilizada como heurística de refinamento foi o método Descida Aleatória, em que somente parte da vizinhança é analisada. A cada iteração um movimento é escolhido aleatoriamente e aplicado sobre a solução corrente. Se houver melhora a solução cor

```
Entrada: s, maxIter
1.  início
2.    r ← Quantidade de movimentos (no caso, r = 4);
3.    Para i ← 1 até maxIter faça
4.      s' ← seleciona aleatoriamente um vizinho
5.      se s' melhor que s então
6.        s ← s';
7.      fim
8.    fim
9.    retorna s;
10. fim
```

Figura 6. Algoritmo Busca Local Descida Aleatória

5.2. *Iterated Local Search* (ILS)

A meta-heurística ILS *(Iterated Local Search)*, segundo [Lourenço et al. 2003], é um método de busca iterativa que faz uso de pertubações da solução (alterações na solução corrente), tendo como principal objetivo a diversificação da busca, de modo a escapar e

visitar diferentes ótimos locais. Quatro são os principais componentes que definem o método ILS, ilustradas no Algoritmo 7 : geração da solução inicial, conforme linha 2; busca local Descida Aleatória, conforme linha 3; pertubação da solução inicial, conforme linha 6; e por fim novamente a busca local. O ILS baseia-se na ideia de aplicar uma busca local em uma solução inicial qualquer até que se encontre um ótimo local, e então, perturbar a solução encontrada e reiniciar a busca local. A função perturbação deve ser de tal forma que seja suficiente para escapar de um ótimo local e permitir a exploração de outras regiões do espaço de buscas. O método ILS é, portanto, um método de busca local que procura focar a busca não no espaço completo de soluções, mas em um pequeno subespaço definido por soluções que são ótimos locais de determinado procedimento de otimização. Neste método considera-se o nível de perturbação a quantidade de vezes que um dos oito movimentos será aplicado. Após a perturbação, essa nova solução é refinada, conforme Algoritmo 7 (linha 7). Se houver melhora na solução, ela é atualizada e o nível de perturbação reinicializado. Esses passos são repetidos até que um critério de parada seja atendido. Como critério de parada, foi adotado o número de 100 iterações sem melhora, ou seja, o método pára quando a melhor solução se mantém inalterada por 100 iterações.

```
Entrada: α, criterioParada
 1. início
 2.    s ← construcaoGRASP(α);
 3.    s* ← buscaLocal(s);
 4.    nivel ← 1;
 5.    repita
 6.       s ← perturbacao(s*, nivel);
 7.       s' ← buscaLocal(s);
 8.       se fo(s') melhor que fo(s*) então
 9.          s* ← s';
10.          nivel ← 1;
11.       senao
12.          nivel ← nivel + 1;
13.       fim se
14.    até criterioParada seja satisfeito;
15. fim
```

Figura 7. Algoritmo ILS

5.3. *Variable Neighborhood Search* (VNS)

A metaheurística *Variable Neighborhood Search* (VNS), proposta por [Mladenovic 1997], VNS é um método de busca local que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança. Este método não segue uma trajetória, mas sim explora vizinhanças mais distantes da solução corrente e focaliza a busca em torno de uma nova solução se e somente se o movimento de melhora é realizado. O VNS inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. A busca local adotada neste trabalho é a Descida Aleatória.

O pseudocódigo do VNS é mostrado no Algoritmo 8. Na linha 2, é construída uma solução com o método `construcaoGRASP`, o mesmo utilizado na metaheurística GRASP. Na linha 3, a solução construída é refinada através de busca local, utilizando

o médoto de Descida Aleatória, já explicado anteriormente. Na linha 6, um vizinho é escolhido aleatoriamente na vizinhança r. Após este procedimento, é refinada e, se houver melhora na solução, ela é atualizada e a vizinhança reinicializado. Caso não haja melhora a vizinhança é alterada.

```
Entrada: α, criterioParada
 1. início
 2.    s ← construcaoGRASP(α);
 3.    s* ← buscaLocal(s);
 4.    r ← 1;
 5.    repita
 6.       s ← vizinho aleatório com movimento r;
 7.       s' ← buscaLocal(s);
 8.       se fo(s') melhor que fo(s*) então
 9.          s* ← s';
10.          r ← 1;
11.       senao
12.          r ← r + 1;
13.       fim se
14.    até criterioParada seja satisfeito;
15. fim
```

Figura 8. Algoritmo VNS

5.4. *Iterated Greedy*

Iterated Greedy (IG) é um método heurístico bastante novo baseado em um princípio muito simples e que pode mostrar um excelente desempenho. [Ruiz and Stutzle 2007] mostrou seu desempenho para um problema de programação *flowshop* de permutação em que o objetivo é minimizar o *makespan* (FSP-Cmax).

O Algoritmo 9 mostra o pseudocódigo do IG. Na linha 2 é construída a solução inicial com o método `construcaoGRASP`, já detalhada na seção 5.2. Na linha 3 é aplicado o método de busca local, descrita no Algoritmo 10, este método consiste em selecionar no (vetor seq) da solução corrente, uma cirurgia aleatoriamente, e inseri-la na melhor posição possível da sequência. Se houver melhora, a solução corrente é atualizada, senão escolhe-se aleatoriamente outra cirurgia e repete o processo. Esses passos são repetidos até que não seja mais possível melhorar a função objetivo, isto significa que tenho a melhor solução até então. Agora é possível aplicar uma das principais funções da metaheuristica IG. O algoritmo proposto aplica iterativamente duas fases. Na primeira fase chamada de destruição, d (no caso $d = 8$) cirurgias são retiradas da solução corrente e armazenadas no vetor π_R, já na fase de construção as cirurgias retiradas são reinseridas na melhor posição da sequência da solução corrente. Após este procedimento a busca local é aplicada novamente. Se houver melhora na solução, então a solução corrente é atualizada, e se esta solução for a melhor solução encontrada a melhor solução é atualizada. A solução corrente poderá ser atualizada também se o critério de aceitação (linha 21) for atendido.

6. Resultados Computacionais

Os algoritmos GRASP, ILS VNS e IG foram implementados na linguagem C++, utilizando a IDE Borland C++ Builder, e testados em um computador Intel Core 2 Duo

```
Entrada: d, Temperatura, criterioParada
1.  início
2.    π ← construcaoGRASP();
3.    π ← BuscaLocal(π);
4.    π_b ← π;
5.    repita
6.      π' ← π;
7.      para i ← 1 até d faça
8.        Retira uma cirurgia aleatoriamente de π' e insere em π_R;
9.      fim
10.     para i ← 1 até d faça
11.       Retira a cirurgia π_R(i) e insere na melhor posição possivel de π';
12.     fim
13.     π' ← BuscaLocal(π);
14.     se π' melhor que π então
15.       π ← π';
16.       se π' melhor que π_b então
17.         π_b ← π;
18.       fim
19.     senão
20.       pr ← Número aleatório entre 0 e 1;
21.       se pr ¡ exp{−(C_max(π') − C_max(π))/Temperatura} então
22.         π ← π';
23.       fim
24.     fim
25.   até criterioParada ser satisfeito;
26. retorna π_b;
27. fim
```

Figura 9. Algoritmo IG

```
Entrada: π
1.  início
2.    melhora ← Verdadeiro;
3.    enquanto melhora faça
4.      melhora ← Falso;
5.      para i ← 1 até n faça
6.        Retirar uma cirurgia k aleatória de π (sem repetição)
7.        π' ← insere a tarefa k na melhor posição possivel de π;
8.        se π' melhor que π então
9.          π ← π';
10.         melhora ← Verdadeiro;
11.       fim
12.     fim
13.   fim
14.   retorna π
15. fim
```

Figura 10. Algoritmo Busca Local IG

2.10GHz e 3GB de memória RAM.

As instâncias utilizadas foram geradas aleatoriamente, usando como base, dados reais pesquisados no ano de 2011 em 4 hospitais classificados como sendo de grande porte e de atendimento geral. Neste experimento utilizam-se, 30 instâncias contendo até 836 cirurgias a serem agendadas. Detalhes podem ser encontrados na Tabela 2.

Na Tabela 3 são apresentados os resultados obtidos para cada metaheurística, aplicada em 30 instâncias. Sendo estes melhor valor de avaliação encontrado (Melhor) e o valor médio encontrado (Médio). Além das soluções iniciais (Sol. Inicial) obtidas na fase

Tabela 2. Instâncias utilizadas e suas características

	Cirurgias	Salas cirúrgicas	Salas de RPA	Leitos de UTI	Cirurgiões	Enfermeiros	Anestesistas
h1	162	7	6	24	74	18	20
h2	192	12	15	29	112	21	25
h3	216	18	20	40	126	27	31
h4	266	16	55	18	157	34	37
h5	354	19	21	53	186	39	45
h6	378	25	26	64	200	45	51
h7	428	23	61	42	231	52	57
h8	408	30	35	69	238	48	56
h9	458	28	70	47	269	55	62
h10	482	34	75	58	283	61	68
h11	570	37	41	93	312	66	76
h12	620	35	76	71	343	73	82
h13	644	41	81	82	357	79	88
h14	674	46	90	87	395	82	93
h15	836	53	96	111	469	100	113
h16	162	6	6	24	74	18	20
h17	192	11	15	29	112	21	25
h18	216	16	20	40	126	27	31
h19	266	14	55	18	157	34	37
h20	354	17	21	53	186	39	45
h21	378	22	26	64	200	45	51
h22	428	20	61	42	231	52	57
h23	408	27	35	69	238	48	56
h24	458	25	70	47	269	55	62
h25	482	30	75	58	283	61	68
h26	570	33	41	93	312	66	76
h27	620	31	76	71	343	73	82
h28	644	36	81	82	357	79	88
h29	674	41	90	87	395	82	93
h30	836	47	96	111	469	100	113

de construção do algoritmo GRASP e utilizadas neste trabalho para comparar os resultados alcançados. O tempo de execução é de 2min para todas as instâncias. Os melhores valores para cada comparação aparecem em azul na tabela.

Verifica-se o melhor desempenho da recente heurística IG em 24 instâncias, ou seja, encontra o melhor resultado em 80% de todas as instâncias testadas. O melhor desempenho ainda se estende aos valores médio encontrados.

O algoritmo IG consegue alocar de maneira eficiente todos os recursos necessários (salas cirúrgicas, salas de RPA, salas de UTI, cirurgiões, enfermeiros e anestesistas) para a realização das cirurgias. Assim a garantia do agendamento é maior, pois todos os recursos estão devidamente alocados respeitando as restrições de disponibilidade. Com isto as salas disponíveis podem receber cirurgias de emergência ou até mesmo outras cirurgias eletivas.

Para comparar os algoritmos com relação ao tempo necessário para encontrar um valor alvo, foi realizado experimentos segundo a abordagem indicada em ([Aiex et al. 2002]). Por meio de gráficos TTTPlot *(Time-To-Target Plots)*, é possível verificar a probabilidade que cada algoritmo possui de alcançar os valores alvos em função do tempo de execução.

A Figura 11 ilustra, por exemplo o comportamento dos algoritmos GRASP, ILS, VNS e IG para a instância h25 a qual possui como característica principal 482 cirurgias a serem agendadas no sequenciamento em questão.

Tabela 3. Resultados

Instâncias	Sol. Inicial	GRASP		ILS		VNS		IG	
	makespan	*makespan*		*makespan*		*makespan*		*makespan*	
	Melhor	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio
h1	189	135	135,40	125	132,20	132	134,60	125	126,40
h2	139	105	109,20	101	103,20	100	105,80	92	93,20
h3	115	91	92,40	86	88,60	82	90,80	69	70,40
h4	216	175	177,40	175	178,60	172	175,40	177	182,20
h5	166	129	130,60	136	141,80	129	142,60	108	109,80
h6	131	113	115,40	123	131,00	118	130,00	93	95,40
h7	161	126	127,60	125	131,00	116	130,00	110	114,20
h8	123	104	107,20	109	122,20	104	121,60	86	87,60
h9	145	119	120,60	123	131,40	118	136,00	107	110,40
h10	146	112	114,20	110	130,80	116	135,40	95	100,60
h11	156	119	122,80	122	136,00	117	134,00	100	100,80
h12	151	125	128,40	123	130,80	118	133,80	110	113,80
h13	143	117	119,20	115	122,40	108	121,20	105	107,00
h14	144	109	110,80	107	108,80	105	112,60	89	92,60
h15	180	119	121,40	117	119,40	107	116,20	109	114,20
h16	230	145	147,40	144	146,60	142	149,40	141	141,40
h17	152	118	119,00	112	114,80	103	113,20	103	104,60
h18	117	96	97,00	90	97,60	82	93,80	80	82,20
h19	198	133	135,40	130	130,40	122	130,00	126	131,40
h20	161	137	139,20	136	144,40	135	142,60	121	126,00
h21	138	114	115,20	113	129,00	123	134,40	100	104,40
h22	150	135	137,20	142	145,80	140	151,20	130	132,20
h23	139	111	114,00	116	126,20	120	130,20	98	99,80
h24	167	133	136,60	131	146,20	130	141,80	125	126,60
h25	150	119	120,40	117	128,00	118	127,00	104	107,40
h26	146	118	119,20	116	126,40	119	131,80	110	111,80
h27	151	136	136,80	133	133,80	123	132,20	127	130,20
h28	171	119	123,20	120	136,40	116	136,40	117	119,40
h29	149	130	131,60	127	137,00	122	135,60	121	122,40
h30	142	130	132,40	127	133,80	122	132,20	123	125,60

7. Conclusão

Neste trabalho foram propostas vizinhanças de grande porte baseada em heurísticas computacionais, que são exploradas através de oito tipos de movimentos. Os experimentos realizados demonstram que estas vizinhanças são eficientes na obtenção de boas soluções para o problema de programação de tarefas em máquinas paralelas idênticas com tempo de preparação dependente da sequência. Os resultados obtidos com apenas 2min de execução de cada metaheurística mostram a eficiência da metodologia adotada para o conjunto de instâncias consideradas.

Além disso, a metodologia proposta mostrou-se robusta, pois, na resolução de uma instância, o resultado final apresentou pouca dependência com relação à qualidade da solução inicial, ou seja, soluções com boa qualidade foram obtidas independentemente se a solução inicial era bem estruturada ou de baixa qualidade.

Outra importante contribuição deste trabalho é poder analisar o comportamento das quatro metaheurísticas implementadas para este tipo problema. Para testar os desempenho, foi realizada uma comparação dos melhores resultados encontrados com a solução inicial gerada pela fase de construção da metaheuristica GRASP, de eficiência demonstrada.

Como trabalhos futuros pretende-se incluir casos de cirurgias de emergência no agendamento semanal de cirurgias.

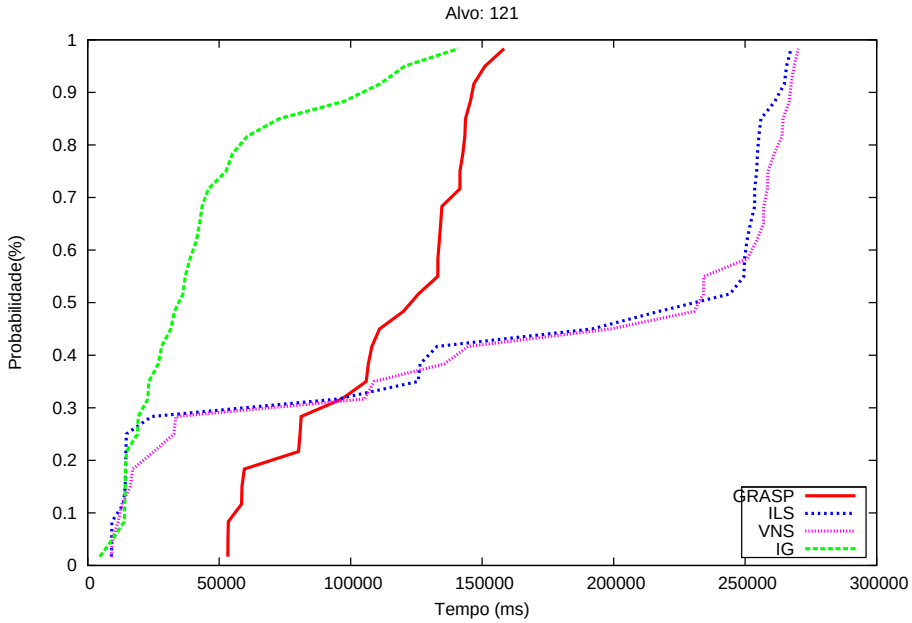

Figura 11. Time-to-target da instância h25 com valor alvo igual a 121.

Agradecimentos: Os autores agradecem ao CEFET-MG, à CAPES, à FAPEMIG e ao CNPq pelo apoio ao desenvolvimento deste trabalho.

Referências

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8:343–373.

Belien, J. and Demeulemeester, E. (2007). Building cyclic master surgery schedules with leveled resulting bed occupancy. *European Journal of Operational Research*, 2(176):1185–1204.

Blake, J. and Carter, M. W. (2002). A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research*, 140:541–561.

Blake, J. and Joan, D. (2002). Mount sinai hospital uses integer programming to allocate operating room time. *Interfaces*, 32(2):63–73.

Cardoen, B., Demeulemeester, E., and Beliën, J. (2010). Operating room planning and scheduling: a literature review. *European Journal of Operational Research*, 3:333–333.

Carter, M. W. and Tovey, C. A. (1992). When is the classroom assignment problem hard? *Operations Research*, 40(1):28–30.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 9:849–859.

Gerchak, Y., Gupta, D., and Mordechai, H. (1996). Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science*, 42:321–334.

Hughes, W. L. and Soliman, S. Y. (1978). Short-term case mix management with linear programming. *Hospital & Health Services Administration*, 30:52–60.

Jebali, A., Atidel, B., Hadj, A., and Pierre, L. (2006). Operating rooms scheduling. *Int. J. Production Economics*, 99:52–62.

Kharrajal, S., Albert, P., and Chaabanel, S. (2006). Bloco de programação para um calendário cirúrgico. *Anais de Conferência Internacional sobre Sistemas de Serviços e Sistemas de Gerenciamento*.

Lourenço, H., Martin, O., and Stützle, T. (2003). Iterated local search. glover,g.f. e kochenberger, editor, handbook of metaheuristics. *Kluwer Academic Publishers, Boston*, pages 321–353.

Macario, A., Vitez, T. S., Dunn, B., and McDonald, T. (1995). Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care. *Anesthesiology*, 83(6):1138–1144.

Magerlein, J. M. and Martin, J. B. (1978). Surgical demand scheduling: A review. *Health Services Research*, pages 418–433.

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

Ozkarahan, I. (1995). Allocation of surgical procedures to operating rooms. *Journal of Medical Systems*, 4(19):333–352.

Pham, D. N. and Klinkert, A. (2008). Surgical case scheduling as a generalized job shopscheduling problem. 185:1011–1025.

Pinedo, M. (2008). *Scheduling: theory, algorithms, and systems*. Springer Verlag.

Proença, I. M. (2010). *Planejamento de Cirurgias Eletivas - Abordagens em Programação Inteira*. PhD thesis, Tese de Doutorado, Departamento de Estatística e Investigação Operacional, Lisboa.

Przasnyski, Z. H. (1986). Operating room scheduling: a literature review. *AORN Journal*, 44:67–79.

Robbins, W. A. and Tuntiwongpiboon, N. (1989). Linear programming is a useful tool in case-mix management. *Healthcare Financial Management*, 6(43):114–116.

Ruiz, R. and Stutzle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049.

Strum, D. P., May, J. H., and Vargas, L. G. (1998). Surgical procedure times are well modeled by the lognormal distribution. *Anesthesia & Analgesia*, 86:47–59.

Torres, N. T. (1999). *Avaliação de desempenho no centro cirúrgico do Hospital Universitário da UFRJ (HUCFF) utilizando a análise envoltória de dados (DEA) e simulação. Diss.* PhD thesis, Universidade Federal do Rio de Janeiro.