DOI: 10.18605/2175-7275/cereus.v10n2p156-169

<<Recebido: 02/06/2018. Aceito: 15/07/2018>>

ARTIGO ORIGINAL

REVISTA CEREUS ISSN: 2175-7275

Algoritmos GRASP e VNS para o Problema de Agendamento de Cirurgias Eletivas em Hospitais de Grande Porte

GRASP and VNS algorithms for the Problem of Scheduling of Elective Surgeries in Large Hospitals

Giselle Paranhos de Andrade[1], Sérgio Ricardo de Souza[2], Adriano C. Machado Pereira[3]

[1] Doutoranda em Modelagem Matemática e Computacional, CEFET-MG

E-mail: giselle.acao@gmail.com

[2] Doutorado em Engenharia Elétrica, UNICAMP.

[3] Doutorado, em Ciência da Computação, UFMG.

RESUMO

Este artigo trata o Problema de Programação de Cirurgia Eletiva (PACE). O PACE será tratado como um Problema de Programação em Máquinas Paralelas Idênticas, no qual o objetivo é minimizar o tempo de conclusão da última cirurgia. Considera-se neste trabalho o período de agendamento de cirurgias como semanal. Há 5 tipos de movimentos, com base em alocações e trocas, para explorar o espaço de soluções. Os algoritmos desenvolvidos com base em GRASP e VNS foram testados usando 75 instâncias com informações reais de hospitais de Minas Gerais, Brasil. Ao final, foi realizado um teste estatístico t-student para comprovar, com 95% de confiança a superioridade do algoritmo VNS e sua capacidade em resolver este tipo de problema para as instâncias testadas.

Palavras-chave: Agendamento de cirurgias. Máquinas paralelas. Otimização. Metaheurísticas.

ABSTRACT

This article deals with the Elective Surgery Programming Problem (PPSE). The PPSE will be treated as a Programming Problem in Identical Parallel Machines, in which the goal is to minimize the time of completion of the last surgery. In this work, it is considered the weekly surgery scheduling period. There are 5 types of movements, based on allocations and exchanges, to explore the space of solutions. The algorithms developed based on GRASP and VNS were tested using 75 instances with real information from hospitals in Minas Gerais, Brazil. At the end, the t-student statistical test was carried out to verify, with 95% confidence, the superiority of the VNS algorithm and its ability to solve this type of problem for the tested instances.

Keywords: Scheduling of surgeries. Parallel machines. Optimization. Metaheuristics.

1. INTRODUÇÃO

O gerenciamento da prestação de serviços de saúde nos hospitais está se tornando cada vez mais importante. Uma unidade de particular interesse é o Centro Cirúrgico (CC), ou melhor, a utilização 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.

Esta questão é tratada no Problema de Agendamento de Salas Cirúrgicas - PASC (*Surgical Case Scheduling - SCS*), problema que consiste em alocar recursos hospitalares para casos cirúrgicos e que também define o melhor instante de realização das cirurgias. Este problema tem um papel decisivo na utilização de recursos hospitalares de forma eficiente, segundo Carter e Tovey (1992). O Problema de Agendamento de Salas Cirúrgicas é considerado, na literatura, um problema clássico de otimização combinatória, pertencente à classe NP-Difícil, segundo Carter e Tovey (1992). Logo, as técnicas heurísticas e as metaheurísticas, de maneira geral, têm sido largamente utilizadas na resolução de problemas desta natureza.

Este artigo se concentra em abordar uma versão do PASC denominada 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, considerando que o período de agendamento tratado é semanal.

O restante deste trabalho está organizado da seguinte forma. Na seção 1.1, é apresentada uma breve revisão bibliográfica para o problema. Na seção 1.2, desenvolve-se a caracterização e definição do problema. Na seção 2, apresenta-se a metodologia adotada para a estrutura de dados utilizada. Na seção 2.1, os algoritmos GRASP e VNS são detalhados. Na seção 2.2, são apresentadas as características das instâncias testadas, bem como alguns exemplos. Na seção 3, são expostas as características computacionais para testar as instâncias do problema. Na seção 4, apresenta-se o teste estatístico realizado para validar os resultados obtidos. Por fim, na seção 5, apresenta-se a conclusão do trabalho e a proposta de alguns trabalhos futuros.

1.1. Revisão Bibliográfica

Segundo Ines (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 ou equipes cirúrgicas. Está situada em um nível estratégico de decisão e, geralmente, é realizada anualmente. A distribuição do tempo considera a capacidade operativa de cada cirurgião ou de cada grupo cirúrgico e a quantidade esperada de pacientes ao longo do correspondente horizonte temporal. Hughes e Soliman (1978), Robbins e Tuntiwongpiboon (1989) e Blake e 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. O horizonte temporal nesta fase do planejamento é mais reduzido do que na primeira fase. Blake e Carter (2002), Blake e Joan (2002) e Belien e Demeulemeester (2007) propõem uma série de modelos para a construção de agendamentos de cirurgias para esta fase. 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 e Martin (1978), Przasnyski (1986), Cardoen et al. (2010), Ozkarahan (1995) e Kharrajal et al. (2006), dentre outros.

Pham e Klinkert (2008) apresentam um modelo em otimização 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 período de

início da última cirurgia a ser realizada. O problema proposto é resolvido com o programa CPLEX.

1.2. Caracterização do Problema de Agendamento de Cirurgias Eletivas - PACE

Nesta seção é apresentada uma descrição do Problema de Agendamento de Cirurgias Eletivas (**PACE**), tratado no presente artigo como um Problema de Programação em Máquinas Paralelas Idênticas (*Identical Parallel Machine Scheduling Problem*), ou seja, um caso particular de problemas de sequenciamento (*Scheduling Problem*).

O Problema de Programação em Máquinas Paralelas Idênticas caracteriza-se por um conjunto $N = \{1, \dots, n\}$ de tarefas, a serem processadas por um conjunto $M = \{M_1, \dots, M_n\}$ de máquinas idênticas, com as seguintes características: (a) cada tarefa deve ser processada exatamente uma vez e por apenas uma máquina; (b) cada tarefa i possui um tempo de processamento p_i; (c) existem tempos de preparação s_{ik} entre as tarefas i e k, considerando que as tarefas n e k serão processadas nesta ordem. Estes tempos de preparação são independentes da sequência, pois é um parâmetro conhecido. O objetivo é encontrar um sequenciamento das n tarefas nas m máquinas de forma a minimizar o tempo de conclusão do sequenciamento, ou seja, o chamado *makespan* ou C_{max}. Pelas características citadas, este problema é definido como $P||C_{max}$, segundo Pinedo (2008).

Com o objetivo de solucionar o PACE usando as características do problema de programação em máquinas paralelas idênticas, considera-se a equivalência entre máquina e sala cirúrgica; e entre tarefa e cirurgia. , Como exemplo, considere, conforme mostra a Tabela 1, os tempos de processamento de sete cirurgias realizadas em duas salas cirúrgicas. A Figura 1 ilustra um possível sequenciamento para este exemplo.

Na Figura 1, observa-se que a cirurgia 6 é alocada na terceira posição da sala S_2, tendo a cirurgia 4 como predecessora e a cirurgia 3 como sucessora. As partes em preto da figura representam os tempos de preparação ou higienização das salas e equipamentos, entre uma cirurgia e outra. O tempo de conclusão das cirurgias na sala S_1 é 60 e o da sala S_2 é 65, o que resulta em um *makespan* de 65 unidades de tempo.

Tabela 1. Tempos de Processamento das cirurgias

n	1	2	3	4	5	6	7
p_n	20	25	15	32	38	23	65

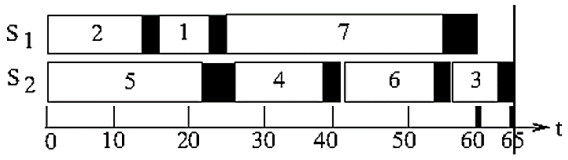

Figura 1. Exemplo de um possível sequenciamento

2. METODOLOGIA

Esta seção apresenta os procedimentos propostos para a solução do Problema de Agendamento de Cirurgias Eletivas (PACE). Inicialmente, é mostrada a estrutura de dados utilizada para a representação de uma solução. Em seguida, apresenta-se os cinco tipos de movimentos utilizados na vizinhança da solução, bem como a busca local utilizada como heurística de refinamento. Após, apresenta-se as metaheurísticas implementadas, a saber, *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Variable Neighborhood Search* (VNS).

Representação da Solução

A solução do PACE é representada por dois vetores. O primeiro vetor, intitulado `vetorseq`, representa a sequência em que as cirurgias devem ser realizadas, ao passo que o segundo vetor, chamado aqui de `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 2 representa a solução para o exemplo apresentado na seção anterior. Verifica-se que as cirurgias de número 1, 2 e 7 são realizadas na sala 1 e as demais na sala 2.

	1	2	3	4	5	6	7
vetor seq	2	5	1	4	7	6	3

	1	2	3	4	5	6	7
vetor sala	1	1	2	2	2	2	1

Figura 2. Representação da solução através de dois vetores

Vizinhança

Para exploração do espaço de soluções, foram utilizadas combinações de dois tipos diferentes de movimentos: troca e realocação, conforme descrito a seguir.

Movimento1 - TrocaOrdemCirurgias: consiste em realizar as trocas de posições entre duas cirurgias realizadas em quaisquer salas;

Movimento2 - RealocaSala: consiste em realocar uma cirurgia para uma outra sala;

Movimento3 - TrocaOrdemCirurgias2: consiste em realizar o Movimento1 duas vezes

no vetor de ordem das cirurgias;

Movimento4 - RealocaSala2: consiste em realizar o Movimento2 duas vezes no vetor de salas;

Movimento5 - TrocaRealoca: consiste em realizar o Movimento1 seguido do movimento2.

Busca Local

Nesta implementação, é utilizado, como heurística de refinamento, o Método Randômico de Descida. A utilização desse método se justifica pela dimensão do espaço de busca do problema tratado e pelo número de combinações e movimentos possíveis na exploração do mesmo. O método implementado consiste em analisar um conjunto de vizinhos gerados a partir de movimentos aleatórios, em que o melhor vizinho encontrado é comparado com a solução corrente. A solução vizinha somente será a nova solução corrente caso seja melhor que a solução corrente. Para a geração da vizinhança, a cada iteração, todos os movimentos descritos na seção anterior têm a mesma probabilidade de ocorrerem. No Algoritmo 1, é apresentado o pseudocódigo da busca local utilizada.

Algoritmo 1. Busca Local – Método *Randômico* de Descida

```
Entrada: s, maxIter
Saída: Melhor vizinho encontrado
início
    r ← Quantidade de movimentos (neste caso, r=5)
    para i ← 1 até maxIter faça
        s' ← seleciona aleatoriamente um vizinho (dentre as r vizinhanças)
        se s' melhor que s então
            s ← s'
        fim
    fim
fim
retorna s
```

2.1. Metaheurísticas Implementadas

Greedy Randomized Adaptive Search Procedure (GRASP)

A metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), proposta por Feo e Resende (1995), é um processo iterativo no qual cada iteração consiste em 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.

Na fase construtiva, uma função denominada `construçãoGRASP` é empregada, gerando uma Lista de Candidatos (LC). Tais candidatos são avaliados segundo um critério em que o tempo necessário para a realização de uma cirurgia seja maior ou igual à diferença entre T_{max} e $\alpha(T_{max} - T_{min})$, sendo α um parâmetro real variando entre 0 e 1. Para $\alpha = 1$, tem-se uma solução totalmente aleatória; para $\alpha = 0$, tem-se uma solução gulosa. Os candidatos que atendem a esta condição compõem a Lista Restrita de Candidatos (LRC). Logo, o algoritmo irá escolher aleatoriamente um dos candidatos da LRC, sequenciando-o e retirando-o da LC. Este passo se repete até que todas as tarefas sejam sequenciadas.

Para a segunda fase ou fase de busca local foi utilizado o Método Randômico de Descida, já descrita na seção anterior. Como já dito anteriormente, este método analisa parte da vizinhança da solução corrente. Os movimentos e vizinhos são escolhidos aleatoriamente. No final, efetua-se uma comparação entre a melhor solução encontrada e a solução corrente. Havendo melhora, a solução corrente é atualizada, senão a solução permanece inalterada.

O pseudocódigo do GRASP é apresentado no Algoritmo 2. Uma solução é construída de forma parcialmente gulosa com o método chamado `construçãoGRASP`, que consiste em escolher aleatoriamente a próxima cirurgia a entrar no sequenciamento, dentre uma lista de melhores candidatos. Essa lista é montada de acordo com a regra de menores tempos de execução. O parâmetro α define o tamanho da lista. Em seguida é feito um refinamento na solução através de busca local. A melhor solução é atualizada se houver melhora. Esses passos são repetidos até que um critério de parada seja atendido.

***Variable Neighborhood Search* (VNS)**

A metaheurística *Variable Neighborhood Search* (VNS), proposta por Mladenovic e Hansen (1997), é uma metaheurística que se baseia em mudanças sistemáticas de vizinhança das soluções para resolver problemas de otimização. O método VNS explora vizinhanças gradativamente mais distantes da solução corrente e focaliza a busca em torno de uma nova solução se e somente se um movimento de melhora é realizado. O VNS inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente.

O pseudocódigo do VNS é apresentado no Algoritmo 3. É construída uma solução aleatória. Em seguida, a solução construída é refinada através de busca local, utilizando o método de Descida Aleatória, já explicado anteriormente. Depois 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 reinicializada.

Algoritmo 2. Pseudocódigo GRASP

```
Entrada: α, criterioParada
início
    repita
        s ← construcaoGRASP(α)
        s' ← buscaLocal(s)
        se fo(s') melhor que fo(s*) então
            s* ← s'
        fim
    até criterioParada seja satisfeito;
fim
```

Algoritmo 3. Pseudocódigo VNS

```
Entrada: α, criterioParada
início
    s ← construcaoAleatoria(α)
    s* ← buscaLocal(s)
    r ← 1
    repita
        s ← vizinho aleatório com movimento r
        s' ← buscaLocal(s)
        se fo(s') melhor que fo(s*) então
            s* ← s'
            r ← 1
        senão
            r ← r + 1
        fim
    até criterioParada seja satisfeito;
fim
```

2.2. Instâncias Propostas para o PACE

As instâncias utilizadas para efetuar os testes computacionais nos algoritmos desenvolvidos, apresentados na seção 2.1, foram geradas a partir de dados reais obtidos em quatro hospitais de grande porte localizados na região metropolitana de Belo Horizonte, pesquisados no ano de 2017, sendo três hospitais da rede privada e um hospital da rede pública. Na Tabela 2, é apresentado um exemplo de como são esquematizadas as informações contidas nas instâncias geradas e testadas.

Tabela 2. Informações das Instâncias

Instância	Cirurgias	Salas Cirúrgicas	Salas de RPA	Leitos de UTI	Cirurgiões	Enfermeiros	Anestesistas

Tabela 3. Instâncias Testadas

Instância	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
h31	162	7	6	24	67	18	20
h32	192	12	15	29	100	21	25
h33	216	18	20	40	114	27	31
h34	266	16	55	18	140	34	37
h35	354	19	21	53	167	39	45
h36	378	25	26	64	181	45	51
h37	428	23	61	42	207	52	57
h38	408	30	35	69	214	48	56
h39	458	28	70	47	240	55	62
h40	482	34	75	58	254	61	68
h41	570	37	41	93	281	66	76
h42	620	35	76	71	307	73	82
h43	644	41	81	82	321	79	88
h44	674	46	90	87	354	82	93
h45	836	53	96	111	421	100	113
h46	162	8	6	24	74	18	20
h47	192	13	15	29	112	21	25
h48	216	20	20	40	126	27	31
h49	266	18	55	18	157	34	37
h50	354	21	21	53	186	39	45
h51	378	28	26	64	200	45	51
h52	428	26	61	42	231	52	57
h53	408	33	35	69	238	48	56
h54	458	31	70	47	269	55	62
h55	482	38	75	58	283	61	68
h56	570	41	41	93	312	66	76
h57	620	39	76	71	343	73	82
h58	644	46	81	82	357	79	88
h59	674	51	90	87	395	82	93
h60	836	59	96	111	469	100	113
h61	162	7	6	24	81	18	20
h62	192	12	15	29	124	21	25
h63	216	18	20	40	138	27	31
h64	266	16	55	18	174	34	37
h65	354	19	21	53	205	39	45
h66	378	25	26	64	219	45	51
h67	428	23	61	42	255	52	57
h68	408	30	35	69	262	48	56
h69	458	28	70	47	298	55	62
h70	482	34	75	58	312	61	68
h71	570	37	41	93	343	66	76
h72	620	35	76	71	379	73	82
h73	644	41	81	82	393	79	88
h74	674	46	90	87	436	82	93
h75	836	53	96	111	517	100	113

Na Tabela 2, tem-se que:
- Instância: representa a instância, denotada pelo prefixo *h;*
- Cirurgias: quantidade de cirurgias previamente conhecida para o agendamento semanal;
- Salas Cirúrgicas: quantidade de salas cirúrgicas disponíveis;
- Salas de RPA: quantidade de salas de recuperação pós-anestésica disponíveis;
- Leitos de UTI: quantidade de leitos de UTI disponíveis;
- Cirurgiões: quantidade de cirurgiões disponíveis;
- Enfermeiros: quantidade de enfermeiros disponíveis;
- Anestesistas: quantidade de anestesistas disponíveis.

Os detalhes de cada instância podem ser encontrados na Tabela 3.

3. RESULTADOS

Os algoritmos GRASP e VNS foram implementados em linguagem C++. As instâncias têm os tempos de cirurgias gerados aleatoriamente entre 2 e 16 *slots* de tempo, com distribuição binomial mais um fator constante 1, se o tempo for menor que 8, e 2, caso contrário.

Os testes computacionais foram realizados em um computador com processador Intel Core i3-M330, 2,13 GHz, 3GB de RAM e sistema operacional Ubuntu 16.04 de 32 bits usando compilador G++ versão 5.4. Foram realizadas 30 execuções de cada algoritmo sobre cada instância, tendo-se como critério de parada o número de iterações, no caso, 100 iterações. Na Tabela 4 são apresentados os resultados obtidos, sendo estes melhores valores de avaliação encontrados (Média (GRASP - VNS)), (Desvio Padrão), (teste-t (H_1:f_VNS<f_GRASP; 95%)), (Quantidade), onde consta a superioridade do algoritmo VNS, o qual reduz o valor de *makespan* em 66 das 75 instâncias testadas.

4. DISCUSSÃO

Para análise estatística e comprovação da existência de diferenças significativas entre os algoritmos implementados, foram utilizados vários testes *t-Student* de uma amostra unicaudal à esquerda. Considerando-se $\mu_0 = \{0{,}8, 10{,}12\}$ e $H_1: \mu > \mu_0$. Seja f_A o valor de função objetivo alcançado pelo algoritmo A e t_A o tempo de execução do algoritmo A. Seguem os seguintes modelos dos testes de hipóteses:

$$\begin{cases} H_1: t_{GRASP} < t_{VNS} \\ H_0: t_{GRASP} \geq t_{VNS} \end{cases} \quad \begin{cases} H_1: f_{GRASP} - f_{VNS} > \alpha \\ H_0: f_{GRASP} - f_{VNS} \leq \alpha \end{cases}, \forall \alpha \in \{0{,}8{,}10{,}12\}$$

Após analisar 30 execuções de cada algoritmo, em cada uma das 75 instâncias, conforme resultado na Tabela 4, percebe-se que:
- É possível afirmar com significância de 95% que a resposta do algoritmo VNS tem um valor de função objetivo menor que as do algoritmo GRASP para 66 das 75 instâncias testadas, lembrando que o problema é de minimização.
- Com 95% de confiança, o algoritmo VNS reduz o *makespan* em mais de 8 *slots* de tempo em 29 das 66 instâncias onde houve melhora.
- Com 95% de confiança, o algoritmo VNS reduz o *makespan* em mais de 10 *slots* de tempo em 22 das 66 instâncias onde houve melhora.
- Com 95% de confiança, o algoritmo VNS reduz o *makespan* em mais de 12 *slots* de tempo em 8 das 66 instâncias onde houve melhora.

- Com relação ao tempo de execução, pode-se afirmar com mais de 99% de confiança que o algoritmo GRASP é cerca de 3,5 vezes mais rápido que o VNS em todos os casos.

Tabela 4. Resultados

	Média (GRASP - VNS)	Desvio Padrão	teste-t (H_1: $f_{VNS} < f_{GRASP}$; 95%)	Quantidade
h1	5,90	9,02239	0,03029	1
h2	3,77	6,19612	0,00308	1
h3	6,17	10,50479	0,00121	1
h4	8,70	8,89847	0,12160	0
h5	10,13	12,05085	0,00992	1
h6	10,60	10,73056	0,01345	1
h7	8,90	11,66294	0,04773	1
h8	7,67	10,71040	0,00045	1
h9	8,87	15,73648	0,02517	1
h10	17,13	14,19503	0,00018	1
h11	16,80	17,89124	0,36330	0
h12	13,57	16,00147	0,01947	1
h13	16,10	11,34217	0,08686	0
h14	16,27	20,52237	0,01994	1
h15	15,73	15,98476	0,19173	0
h16	10,23	12,37262	0,00145	1
h17	5,23	7,26201	0,00001	1
h18	11,50	12,19596	0,00855	1
h19	6,90	8,10002	0,04556	1
h20	6,53	8,50044	0,00433	1
h21	10,20	11,17077	0,00124	1
h22	7,57	11,09422	0,00147	1
h23	12,57	13,41045	0,00506	1
h24	12,40	14,69835	0,00003	1
h25	15,40	17,92244	0,21713	0
h26	11,83	12,80647	0,00017	1
h27	10,10	12,28217	0,02025	1
h28	11,60	13,18201	0,00001	1
h29	16,80	16,19366	0,02605	1
h30	17,57	18,04340	0,00079	1
h31	7,10	10,61343	0,11177	0
h32	6,87	10,13609	0,01812	1
h33	7,07	9,05132	0,00879	1
h34	6,00	9,28105	0,00006	1
h35	11,40	13,04792	0,00067	1
h36	10,07	11,79402	0,00386	1
h37	6,97	8,63227	0,00044	1
h38	10,37	13,52771	0,00046	1
h39	7,90	9,35267	0,03299	1
h40	12,13	12,74615	0,00159	1
h41	19,93	17,44337	0,00359	1
h42	16,63	15,85165	0,00416	1
h43	16,00	18,90265	0,00094	1
h44	21,00	18,13551	0,03129	1
h45	9,10	12,11539	0,00245	1
h46	4,83	7,25441	0,02660	1
h47	5,50	7,17154	0,00277	1
h48	9,17	10,51135	0,01629	1
h49	7,03	9,41196	0,00576	1
h50	9,93	10,32217	0,00462	1
h51	10,00	12,41523	0,01028	1
h52	4,73	7,93914	0,00003	1
h53	8,63	8,89976	0,00277	1
h54	13,50	13,54367	0,18564	0
h55	14,63	13,28853	0,05643	0
h56	16,63	16,78358	0,00922	1
h57	12,53	15,45345	0,00014	1
h58	11,43	13,13244	0,00062	1
h59	14,57	14,75240	0,00683	1
h60	25,30	18,37755	0,02270	1
h61	6,37	9,65074	0,00614	1
h62	6,47	10,44108	0,02758	1
h63	8,07	9,78364	0,13333	0
h64	6,60	8,52420	0,00044	1
h65	6,07	7,58371	0,00185	1
h66	11,97	12,15839	0,00301	1
h67	15,60	19,00744	0,00876	1
h68	11,10	12,39953	0,01894	1
h69	13,27	13,49823	0,00004	1
h70	18,90	16,19994	0,00478	1
h71	18,20	21,12100	0,02872	1
h72	21,20	18,27076	0,00166	1
h73	15,60	18,57993	0,00193	1
h74	15,03	14,49015	0,00282	1
h75	14,43	14,63271	0,01231	1

5. CONSIDERAÇÕES FINAIS

Este trabalho apresentou um estudo acerca da aplicação de metaheurísticas para resolução do Problema de Agendamento de Cirurgias Eletivas em Hospitais de Grande Porte - PACE. Abordou-se aqui a metodologia de Programação de Máquinas Paralelas, uma classe de problemas de agendamento, em que se pretende realizar o agendamento semanal de cirurgias previamente conhecidas, com o objetivo de minimizar o *makespan*, ou instante de término da última cirurgia. São considerados, neste trabalho, a disponibilidade dos recursos de sala cirúrgica, sala de recuperação pós-anestésica (RPA), leitos de UTI, cirurgiões, anestesistas e enfermeiros. Para solucionar o PACE, foram implementadas as metaheurísticas GRASP e VNS, a busca local ou heurística de refinamento utilizada foi o Método Randômico de Descida. A utilização desta busca local se justifica pelo grande espaço de busca que o problema apresenta. Por meio deste trabalho, foi possível testar e desenvolver diferentes técnicas, de forma adaptada ao PACE, permitindo a obtenção de soluções de maior qualidade, quando comparadas com as soluções encontradas atualmente nos hospitais. Com a finalidade de testar a metodologia proposta, foram geradas instâncias com dados reais de quatro hospitais de grande porte pesquisados durante o ano de 2017. Foram geradas 75 instâncias, contendo até 836 cirurgias semanais, a serem agendadas em 53 salas cirúrgicas. De maneira geral, utilizando as instâncias geradas, os algoritmos desenvolvidos apresentaram resultados muito promissores, sendo capazes de encontrar soluções melhores do que as soluções presentes, atualmente, nos hospitais em 100% das instâncias testadas, como é o caso da metaheurística VNS que se destaca para este grupo de instâncias.

Conclui-se com este trabalho e com os resultados apresentados que o algoritmo VNS contribui para a resolução do Problema de Agendamento de Cirurgias Eletivas. Demonstrando assim a grande importância de explorar várias vizinhanças e manter movimentos mais robustos.

Para análise estatística e comprovação da existência de diferenças significativas entre os algoritmos implementados, foram utilizados vários testes *t-Student* de uma amostra unicaudal à esquerda.

Como proposta de trabalhos futuros, os autores consideram:

-Testar e propor um agendamento mensal para o PACE;

-Testar um método exato, usando *Optimization Programming Language* (OPL);

-Implementar a inclusão de agendamento de cirurgias de emergência a atual agenda;

-Implementar com a linguagem de programação Python usando bibliotecas de heurísticas.

REFERÊNCIAS

AIEX, R.M.; RESENDE, M.G.C.; RIBEIRO, C.C. Probability distribution of solution time in grasp: an experimental investigation. **Journal of Heuristics**, v. 8, 343–373, 2002.

BELIEN, J.; DEMEULEMEESTER, E. Building cyclic master surgery schedules with leveled resulting bed occupancy. **European Journal of Operational Research**, v. 2, n. 176, 1185–1204, 2007.

BLAKE, J.T.; CARTER, M.W. A goal programming approach to strategic resource allocation in acute care hospitals. **European Journal of Operational Research**, v. 140, 541–561, 2002.

BLAKE, J.T.; JOAN, D. Mount sinai hospital uses integer programming to allocate operating room time. **Interfaces**, v. 32, n. 2, 63–73, 2002.

CARDOEN, B.; DEMEULEMEESTER, E.; BELIËN, J. Operating room planning and scheduling: a literature review. **European Journal of Operational Research**, v. 3, 333–333, 2010.

CARTER, M.W.; TOVEY, C.A. When is the classroom assignment problem hard? **Operations Research**, v. 40, n. 1, 28–30, 1992.

DEXTER, F.; TRAUB, R.D. How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time. **Anesthesia and Analgesia**, v. 94, n. 4, 933–942, 2002.

FEO, T.A.; RESENDE, M.G.C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, 109–133, 1995.

FIELD, A.P. **Descobrindo a estatística usando o SPSS**. 2 edição, 2009.

HUGHES, W.L.; SOLIMAN, S.Y. Short-term case mix management with linear programming. **Hospital & Health Services Administration**, v. 30,52–60, 1978.

KHARRAJAL, S.; ALBERT, P.; CHAABANEL, S. Bloco de programação para um calendário cirúrgico. In: CONFERÊNCIA INTERNACIONAL SOBRE SISTEMAS DE SERVIÇOS E SISTEMAS DE GERENCIAMENTO. **Anais...**, 2006.

LOURENÇO, H.R.; MARTIN, O.C.; STÜTZLE, T. Iterated local search. **Handbook of Metaheuristics**. Kluwer Academic Publishers, Boston, 321–353, 2003.

MACARIO, A.; VITEZ, T.S.; DUNN, B.; MCDONALD, T. Where are the costs in perioperative care?: analysis of hospital costs and charges for inpatient surgical care. **Anesthesiology**, v. 83, n. 6, 1138–1144, 1995.

MAGERLEIN, J.M.; MARTIN, J.B. Surgical demand scheduling: a review. **Health Services Research**, 418–433, 1978.

MLADENOVIC, N; HANSEN, P. Variable neighborhood search. **Computers and Operations Research**, v. 24, 1097–1110, 1997.

OZKARAHAN, I. Allocation of surgical procedures to operating rooms. **Journal of Medical Systems**, v. 4, n. 19, 333–352, 1995.

PHAM, D.N.; KLINKERT, A. **Surgical case scheduling as a generalized job shop scheduling problem**. v. 185,1011–1025, 2008.

PINEDO, M. **Scheduling**: theory, algorithms, and systems. Springer Verlag, 2008.

PROENÇA, I.M. **Planejamento de cirurgias eletivas**: abordagens em programação inteira. Tese de Doutorado, Departamento de Estatística e Investigação Operacional, Universidade de Lisboa, 2010.

PRZASNYSKI, Z.H. Operating room scheduling: a literature review. **AORN Journal**, v. 44, 67–79, 1986.

ROBBINS, W.A.; TUNTIWONGPIBOON, N. Linear programming is a useful tool in case-mix management. **Healthcare Financial Management**, v. 6, n. 43, 114–116, 1989.