

SCAP - Software web para o problema de alocação de salas

Rafael Dutra Jardim

Centro Universitário Una

Av. João César de Oliveira, 5.775, Beatriz, Contagem, Minas Gerais

rafaeldjardim@gmail.com

Rodrigo de Carvalho

Centro Universitário Una

Av. João César de Oliveira, 5.775, Beatriz, Contagem, Minas Gerais

carvalho.rodrigo@prof.una.br

RESUMO

Este trabalho apresenta um software Web utilizando ferramentas e métodos da Pesquisa Operacional para resolver o problema conhecido como *Classroom Assignment Problem* (CAP). Este problema tem sua devida importância dentre os vários problemas de alocação de recursos sob restrições e tem sido muito estudado nas últimas décadas. Foram utilizados dois *solvers*, Gurobi e SCIP, além de uma heurística paralela para resolver o CAP. Os experimentos computacionais demonstraram que, para o conjunto de instâncias consideradas, o CAP pode ser tratado eficientemente com ambos os recursos.

PALAVRAS CHAVE. Heurística, Software, Otimização.

ABSTRACT

This paper presents a Web-based software that embeds Operations Research methods to solve the problem known as *Classroom Assignment Problem* (CAP). This problem has its due importance among the various resource allocation problems and it has been much studied in the last decades. We use two solvers, Gurobi and SCIP, as well as a parallel heuristic to solve the CAP. Computational experiments show that, for the instances set considered, CAP can be handled efficiently with both resources.

KEYWORDS. Heuristic, Software, Optimization.

1. Introdução

Nos últimos anos a comunidade acadêmica vem se esforçando para buscar soluções automatizadas para o processo de alocação de recursos sob restrições. Tais problemas surgem em diversos contextos, tais como: eventos esportivos (Schonberger. et al. [2004]; Mine et al. [2006]), transporte (Schoenauer [2005]), escalonamento de enfermeiras em hospitais (Croce e Salassa [2014]) e horários educacionais (Aubin e Ferland [1989]; Petrovic e Burke [2004]; Burke et al. [2007]; Achá e Nieuwenhuis [2010]). Dentre os problemas desta natureza, um muito estudado é conhecido como *Classroom Assignment Problem* (CAP). Este problema consiste em definir em quais salas determinadas aulas serão alocadas, respeitando um conjunto de restrições e um horário predefinido. De forma geral o problema é NP-Difícil, embora exista casos que podem ser resolvidos em tempo polinomial (W.Carter e Tovey [1992]).

O CAP tem sua devida importância, isso porque a atribuição das turmas as salas de aula pode influenciar diretamente na vida de funcionários, alunos e professores de uma Instituição de Ensino (IE). Uma vez implementada a atribuição, essa normalmente é utilizada durante todo o período letivo, fazendo com que as pessoas tenham que se adaptar a ela, pois as soluções geralmente não atendem à todos os interesses e disponibilidade dos recursos envolvidos. Não menos importante, uma solução ruim para este problema pode implicar em aumento de custos para a IE, isso porque pode gerar a necessidade de novas salas, recursos (computadores, datashow etc) ou períodos extras para suprir uma demanda.

Além disso, a procura de soluções de forma manual pode ser uma tarefa árdua, em alguns casos impraticável, pois demandaria serviço de diversas pessoas durante vários dias e, mesmo assim, ao fim do processo a solução encontrada poderia não ser satisfatória, ou mesmo viável. Assim sendo, sistemas computacionais que auxiliem na boa administração dos interesses das IE, expressos em uma atribuição adequada, passa a ser um problema de grande interesse prático, ao lado de sua importância teórica.

É possível encontrar diversos trabalhos na literatura para tratar este tipo de problema. Parte dos trabalhos utilizam metodologias baseadas em heurísticas (Dammak et al. [2006]; Freire e Melo [2006]; Elloumi et al. [2014]; Burke et al. [2010]). Entretanto, também é possível encontrar propostas baseadas em métodos exatos (Gosselin e Truchon [1986]; Glassey e Mizrach [1986]; Qualizza e Serafini [2005]; Lach e Lübbeck [2012]). Embora exista um grande número de métodos eficientes para tratar este problema, vale salientar que uma característica importante no caso dos problemas de alocação de salas é sua unicidade, pois cada IE tem diferentes estruturas físicas, recursos e valores, necessitando assim de modelos específicos (Wendt e Mullher [2017]). Por outro lado, o grande número de trabalhos na área permite que sejam propostos modelos e algoritmos cada vez mais eficientes para resolver este problema, uma vez que características importantes já foram estudadas, como pode ser visto no trabalho apresentado por W.Carter e Tovey [1992].

Neste trabalho é proposto um software que faz uso de técnicas de otimização para resolver o CAP de uma IE situada no estado de Minas Gerais, Brasil. Além disso, é proposto um modelo matemático baseado no conceito de programação por metas. As técnicas de modelagem matemática, em grande parte, estão condicionadas à solução de um único objetivo e assumem que as restrições não podem ser violadas. Entretanto, inúmeras problemas existentes no mundo real exigem que as decisões do dia a dia sejam flexíveis. Dessa forma, os gestores procuram satisfazer ou pelo menos se aproximar dos objetivos estabelecidos, em vez de considerá-los como rígidos. A programação por metas é uma técnica que permite a modelagem e a busca de soluções para os problemas com múltiplos objetivos ou metas a serem otimizados, flexibilizando algumas restrições. Uma das características que diferem esse tipo de programação é a função objetivo, onde se tem variáveis de desvio, que têm como propósito medir os desvios entre as metas estabelecidos. Nessas condições, a solução "ideal" seria aquela na qual todas as variáveis de desvio tivessem valor igual a zero, ou seja, em que todos os objetivos tivessem sido atingidos em seus parâmetros estipulados (Martins et al. [2007]).

Para obter soluções para o modelo proposto foram utilizados os solvers Gurobi e SCIP. Além disso, foi utilizada uma eficiente heurística paralela. Os resultados obtidos neste trabalho mostram que ambas as abordagens são aptas a fornecerem respostas eficientes para os tomadores de decisão.

2. Classroom assignment problem

O CAP tratado neste trabalho é baseado nas seguintes definições: Seja $I = \{i_1, i_2, \dots, i_i\}$ o conjunto de disciplinas oferecidas, $J = \{j_1, j_2, \dots, j_j\}$ o conjunto de horários disponibilizados durante a semana e $K = \{k_1, k_2, \dots, k_k\}$ o conjunto de salas. Considere $a_{ij} \in \{0, 1\}$ para indicar se uma disciplina $i \in I$ é alocada a um horário $j \in J$. Se a disciplina foi alocada $a_{ij} = 1$, caso contrário $a_{ij} = 0$. Além disso, considere d_i o número de alunos matriculados na disciplina $i \in I$, c_k a capacidade da sala $k \in K$ e q_k um parâmetro igual a 1 caso a sala $k \in K$ possui ar condicionado e 0 caso contrário. g é o máximo de alunos que podem ser designados a uma sala, sem que esta tenha ar condicionado.

Ainda, para representar uma solução e mensurar a qualidade da mesma, temos as seguintes variáveis: $x_{ijk} \in \{0, 1\}$ para indicar se uma disciplina $i \in I$ alocada ao horário $j \in J$ foi atribuída a uma sala $k \in K$. Se a disciplina foi alocada $x_{ijk} = 1$, caso contrário $x_{ijk} = 0$; r_{ijk} indicando a quantidade de alunos que foram alocados em uma sala sem ar condicionado (além do limite g), onde $i \in I, j \in J$ e $k \in K$; s_{ijk}^+ e s_{ijk}^- , sendo, respectivamente, o número de alunos acima e abaixo da capacidade de uma sala $k \in K$, onde $i \in I, j \in J$; $v_{ij} \in \{0, 1\}$ para indicar se a disciplina $i \in I$ foi atribuída, em algum momento, para a sala $j \in J$. Se houve a alocação $v_{ij} = 1$, caso contrário $v_{ij} = 0$; t_i sendo a quantidade de salas que uma disciplina $i \in I$ utilizou na atribuição. As equações de 1 a 9 modelam o problema.

$$\min z = \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} \left[\alpha_1 r_{ijk} + \alpha_2 s_{ijk}^+ + \alpha_3 s_{ijk}^- \right] + \alpha_4 \sum_{i \in I} t_i \quad (1)$$

$$(d_i x_{ijk})(1 - q_k) - g - r_{ijk} \leq 0 \quad \forall i \in I, j \in J, k \in K \quad (2)$$

$$(c_k - d_i)x_{ijk} + s_{ijk}^+ - s_{ijk}^- = 0 \quad \forall i \in I, j \in J, k \in K \quad (3)$$

$$\sum_{i \in I} x_{ijk} \leq 1 \quad \forall j \in J, k \in K \quad (4)$$

$$\sum_{k \in K} x_{ijk} = a_{ij} \quad \forall i \in I, j \in J \quad (5)$$

$$x_{ijk} - v_{ik} \leq 0 \quad \forall i \in I, j \in J, k \in K \quad (6)$$

$$t_i \geq \sum_{k \in K} v_{ik} - 1 \quad \forall i \in I \quad (7)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in I, j \in J, k \in K \quad (8)$$

$$s_{ijk}^+, s_{ijk}^-, r_{ijk}, t_i \geq 0 \quad \forall i \in I, j \in J, k \in K \quad (9)$$

A equação 1 representa a função objetivo, que visa a minimização dos desvios referentes a utilização dos recursos em relação as metas. A equação 2 diz respeito ao atendimento da meta de lotação máxima de alunos em uma sala em relação ao ar condicionado. A equação 3 obtém os o número de cadeiras vazias ou número de alunos além da capacidade de uma sala. A equação 6

obtem as salas utilizadas por cada disciplina. A equação 7 obtém o número de salas utilizadas por cada disciplina. Estas restrições são denominadas flexíveis e auxiliam no cálculo da qualidade de uma solução. Já as restrições representadas pelas equações 4 e 5 representam as restrições rígidas. Estas precisam ser atendidas para que a solução seja viável. A equação 4 assegura que disciplinas que são lecionadas em um mesmo horário não sejam alocadas em uma mesma sala. A equação 5 assegura que se houver aula haverá uma alocação para ela em uma sala. As restrições 8 e 9 dizem respeito a integralidade e não negatividade das variáveis.

Além disso, os parâmetros α consistem nas penalidades atribuídas a cada uma das metas representados pelas restrições flexíveis. Esses parâmetros são configurados de acordo com a priorização das metas do utilizador do sistema.

No caso de já existir uma solução que deve ser modificada, por indisponibilidade de uma ou mais salas, é necessário inserir a restrição representada pela equação 10. Essa equação define o número máximo de mudanças que podem ocorrer em relação a alocação disciplinas $\times$ salas. Onde $\bar{x}_{ijk}$ representa a solução atual, considerando somente as salas disponíveis. Além disso, temos $S_0 = \{(i, j, k) : \bar{x}_{ijk} = 0\}$, $S_1 = \{(i, j, k) : \bar{x}_{ijk} = 1\}$ e M representando o número máximo de realocações que podem ser feitas. Isso tenta minimizar os impactos que uma mudança poderia gerar em toda a alocação, diminuindo os transtornos na IE.

$$\sum_{(i,j,k) \in S_0} x_{ijk} + \sum_{(i,j,k) \in S_1} (1 - x_{ijk}) \leq M \quad (10)$$

3. Software

Desenvolver softwares que sejam capazes de fornecerem dados para as técnicas da Pesquisa Operacional, bem como manipular as soluções obtidas é muito importante. Isso permite que pessoas leigas nesta área façam o uso dos benefícios que tais metodologias podem fornecer, de forma simples e direta.

Ao desenvolver o software, denominado aqui de SCAP, buscamos fornecer um ambiente amigável para se utilizar o modelo desenvolvido neste trabalho, ou qualquer outro que possa ser acoplado ao sistema desde que obedeça a interface de comunicação adotada. O projeto do SCAP é baseado na arquitetura cliente-servidor, conforme ilustrado na Figura 1. Foram criados três módulos principais, sendo eles: armazenamento de dados, servidor WEB e solver. Isso permite que se altere um módulo sem impacto significativo nos outros, desde que as interfaces sejam respeitadas.

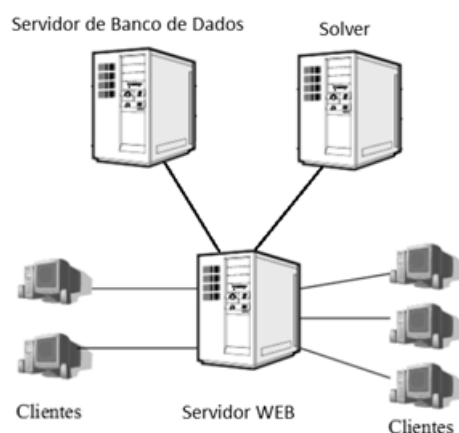


Figura 1: Organização do SCAP

Para o módulo do servidor WEB foi desenvolvido um sistema WEB utilizando o servidor Apache e a linguagem PHP, dentre outros recursos para manipulação e apresentação de dados,

tais como JavaScript, HTML, CSS etc. O módulo Servidor de Banco de Dados foi construído utilizando o banco de dados MySQL via REST sobre HTTP. Por fim, no servidor responsável pelo solver foi configurado com os *solvers* Gurobi, SCIP e um método heurístico. O objetivo é deixar o sistema disponível para uso com um resolvidor sem custos. Por outro lado, aferimos a diferença de desempenho entre os *solvers* para auxiliar o tomador de decisão da IE em relação a qual tecnologia utilizar.

A figura 2 mostra parte da interface gráfica responsável em mostrar uma solução para o PAC. É possível que o usuário altere a solução proposta pelo solver, ou seja, aloque aulas em salas diferentes daquelas propostas pelo mecanismo de otimização. Isso é interessante porque alguns autores acreditam que os problemas de alocação não podem ser totalmente automatizados. Entre as justificativas para isso, uma é que existem razões que não podem ser facilmente expressas em um sistema automatizado, de forma que torne um alocação melhor que o outra.

		SEGUNDA	TERÇA	QUARTA
101	1	ENG1AN-BRA MATEMÁTICA DESCRITIVA	ENG1AN-BRA REDAÇÃO DE DOCUMENTOS	SIN2AN-BRA MATEMÁTICA DESCRITIVA
	2		ENG1AN-BRA REDAÇÃO DE DOCUMENTOS	ADM2AN-CON INTERFACE HOMEM-MÁQUINA
102	1	ADM2AN-CON MATEMÁTICA DESCRITIVA		ADM2AN-CON REDAÇÃO DE DOCUMENTOS
	2	SIN2AN-BRA PROGRAMAÇÃO ORIENTADA A OBJETOS	ADM2AN-CON INTERFACE HOMEM-MÁQUINA	ADM2AN-CON REDAÇÃO DE DOCUMENTOS

Figura 2: Interface Gráfica SCAP

Ainda é possível cadastrar disciplinas, turmas, salas e criar atribuições específicas. Entre os cadastros, o de turma merece atenção. Após o cadastro inicial, onde são requeridos somente o nome da turma e seu código, é necessário associar quais disciplinas a turma está vinculada, o número esperado de alunos, bem como o seu horário semanal. Parte dessa interface é mostrada na Figura 3.

A versão beta do software esta disponível em <http://ensalamentoapp.herokuapp.com>. Vale ressaltar que esta versão atende uma IE específica, esforços estão sendo concentrados para que a próxima versão possa atender uma vasta quantidade de instituições. A ideia é modelar as principais restrições encontradas na literatura e ativa-las via software quando houver necessidade.

4. Heurística

Nesta seção a heurística proposta é apresentada.

4.1. Método de Construção

A ideia geral do método de construção é escolher uma sala para atender uma disciplina onde a diferença entre a demanda da disciplina e a capacidade da sala seja a menor possível. Assim sendo, as salas são ordenadas pela sua capacidade de forma crescente. Já as disciplinas, são ordenadas de acordo com a sua demanda de forma decrescente. A primeira sala que estiver disponível e que tenha capacidade para atender a disciplina é escolhida para sediar a aula. A Figura 4 ilustra os principais passos deste método.

No caso onde uma solução existente é inviabilizada pela indisponibilização de alguma sala, o procedimento sofre uma pequena modificação. São consideradas somente as aulas que

TURMAS		VISUALIZAR DISCIPLINAS (SIN2AN-BRA)	
←	📅	< ANO 2018 / 1º SEMESTRE >	→
DISCIPLINA	QUANTIDADE DE ALUNOS		
MATEMÁTICA COMPUTACIONAL	50	🗑️	✎️
MATEMÁTICA DESCRITIVA	50	🗑️	✎️
INTERFACE HOMEM-MÁQUINA	50	🗑️	✎️
PROGRAMAÇÃO ORIENTADA À OBJETOS	50	🗑️	✎️

Figura 3: Interface Gráfica SCAP

```

Algoritmo Método de Construção 1
1Início
2   Para  $i \leftarrow 0$  até  $|I|$ 
3     Para  $j \leftarrow 0$  até  $|J|$ 
4        $f \leftarrow 0$ 
5       Se  $(a_{ij} = 1)$  então
6          $k \leftarrow 1$ 
7         Faça
8           Se  $(d_i \leq c_k)$  e  $(x_{ijk} = 0)$  então
9              $x_{ijk} \leftarrow 1$ 
10             $f \leftarrow 1$ 
11          Fim-Se
12           $k \leftarrow k + 1$ 
13        Enquanto  $(k < |K|)$  e  $(f = 0)$ 
14          Se  $f = 0$ 
15             $x_{ijk} \leftarrow 1$ 
16          Fim-Se
17        Fim-Se
18      Fim-Para
19    Fim-Para
20FIM

```

Figura 4: Método de construção.

foram desalocadas, ou seja, aquelas onde a_{ij} foram alocadas as salas k que foram indisponibilizadas. Neste caso, considerando Q o conjunto representando estas aulas, onde cada elemento é representado pela tupla (i, j) , o método ilustrado na Figura 5 mostra o passo a passo de como construir uma solução.

4.2. Estruturas de Vizinhança e Perturbação

Existe uma grande variedade de estruturas de vizinhança que podem ser exploradas pelos métodos de busca local, para o problema tratado neste trabalho foram utilizadas 3 estruturas, denominadas: "Troca de sala", "Muda de Sala" e "Onda de Realocação". Na estrutura "Troca de Sala" as aulas são trocadas de sala com aulas de outra disciplina. Nesta estrutura são trocadas o maior número de aulas possível de uma mesma disciplina, que melhore a função objetivo, respeitando a disponibilidade da sala. A estrutura "Muda de Sala" troca aulas de uma disciplina para salas que estão ociosas. Já a estrutura "Onda de Realocação" troca aulas de uma disciplina de sala, não respeitando a disponibilidade da sala para o qual serão redirecionadas as aulas. Se a troca é direcionada para uma sala que não está ociosa, um procedimento para viabilizar a solução é executado. Este procedimento consiste em fazer a realocação das aulas que estavam anteriormente alocadas a

```

Algoritmo Método de Construção 2
1 Início
2   Para cada  $i, j \in Q$ 
3      $f \leftarrow 0$ 
4      $k \leftarrow 1$ 
5     Faça
6       Se  $(d_i \leq c_k)$  e  $(x_{ijk} = 0)$  então
7          $x_{ijk} \leftarrow 1$ 
8          $f \leftarrow 1$ 
9       Fim-Se
10       $k \leftarrow k + 1$ 
11      Enquanto  $(k < |K|)$  e  $(f = 0)$ 
12        Se  $f = 0$ 
13           $x_{ijk} \leftarrow 1$ 
14        Fim-Se
15      Fim-Para
16 FIM

```

Figura 5: Método de construção parcial.

sala para uma ociosa.

Para perturbar uma solução foram utilizadas as estruturas de vizinhança "Troca de Sala" e "Muda de Sala".

4.3. ILS-RVND

O método proposto neste trabalho é baseado na meta-heurística *Iterated Local Search* (ILS, Lourenco et al. [2003]), tendo sua busca local guiada pelo método *Random Variable Neighborhood Descent* (RVND). Este método segue os passos descritos a seguir. Inicialmente, uma solução é gerada através do método de construção descrito anteriormente, então n -threads são criadas para executar o restante do algoritmo. Para cada *thread* criada, o método RVND executa até que encontre uma solução ótima local. Este método explora, sistematicamente, diferentes estruturas de vizinhança. A cada iteração todas as estruturas são utilizadas, porém em diferentes sequências. Isso reduz os riscos do algoritmo convergir prematuramente. Além disso, é possível explorar diferentes áreas no espaço de busca, uma vez que a ordem das estruturas de vizinhança pode levar a diferentes soluções. O critério utilizado para a busca local é conhecido como Primeiro Aprimorante.

Posteriormente, cada *thread* verifica se sua solução é melhor que a melhor solução encontrada até o momento e atualiza esta solução caso necessário. A penúltima etapa é decidir se a próxima solução corrente será a melhor encontrada até aquele momento ou a solução corrente da *thread*. Essa decisão é tomada aleatoriamente. Após esta escolha, uma perturbação é aplicada a solução corrente e novamente é aplicada a busca local. O nível de perturbação em cada *thread* é independente e aumenta a cada vez que uma busca local é executada e não melhora a melhor solução encontrada até o momento. O objetivo é tentar explorar melhor o espaço de soluções e não precisar calibrar nenhum tipo de parâmetro. O algoritmo termina quando uma solução alvo é encontrada ou o tempo de processamento é esgotado.

O nível de perturbação varia entre 5 a 50% (UB e LB), sendo aplicada, obrigatoriamente, a pelo menos uma disciplina. Quando alcança o UB , a perturbação automaticamente volta assumir o LB . Além disso, o valor é atualizado somando 2% ao valor corrente. A regra de atualização da perturbação, para cada *thread*, segue os seguintes passos: se a melhor solução conhecida não foi melhorada, aumente a perturbação; caso contrário, envie uma informação para todas as outras *threads* para reiniciar o valor das suas perturbações com LB .

5. Resultados

Esta seção mostra o desempenho dos três diferentes mecanismos de resolução para o problema tratado. O principal objetivo é mostrar que as três opções podem ser utilizadas, cada

uma com a sua vantagem. Se por um lado o método heurístico obtém soluções de qualidade em um tempo relativamente pequeno, por outro não existe nenhuma garantia de otimalidade ou qualidade da solução. Já os *solvers* são capazes de informar o quanto uma solução está próxima ao ótimo, porém para instâncias maiores podem demandar um tempo superior. Outra questão que deve ser considerada é o custo envolvido para a utilização de cada mecanismo. A heurística proposta é disponibilizada para utilização sem nenhum custo, enquanto a utilização do solver depende de aquisição de uma licença.

Para aferir o desempenho destes mecanismos foram utilizadas 15 instâncias criadas para este trabalho. Elas variam o número de horários e salas disponíveis, bem como o número de disciplinas e aulas. A menor instância possui 25 disciplinas, totalizando 118 aulas a serem alocadas. Já a maior conta com 600 disciplinas e 3288 aulas. Todas as instâncias utilizadas neste trabalho, bem como os códigos do software podem ser encontrados no site www.rclink.com.br. Todo o experimento foi executado em um computador com dois processadores, cada processador contando com 6 núcleos (24 *threads*), modelo Intel Xeon E5-2620 de 2.1 GHz com 64 Gb RAM, com sistema operacional Ubuntu 14.

A Tabela 1 mostra a solução encontrada por cada *solver*, SCIP e Gurobi, após um limite de 30 minutos de processamento. É mostrado o valor médio da função objetivo, bem como o *gap* para a solução ótima. Além disso, a última coluna mostra o *gap* obtido pela heurística em relação a melhor solução encontrada entre os *solvers*. O critério de parada utilizada pela heurística foi 3 minutos de processamento ou a obtenção da solução alvo. O SCIP, Gurobi e a heurística foram executados 10 vezes para cada instância. Os *solvers* foram executados em modo *default*, enquanto a heurística utilizou 15 *threads* para a busca local.

Os valores de α foram configurados com 100. Estes valores devem refletir a necessidade e valores de cada IE, aqui não demos prioridade a nenhuma das metas. Em testes práticos, onde a instituição prioriza cadeiras sobrando dentro da sala em relação a superlotação, além da manutenção de aulas de uma mesma disciplina em uma mesma sala e a não alocação de disciplinas com mais de 50 alunos em salas sem ar condicionado, os valores de α_1 , α_2 , α_3 e α_4 foram configurados, respectivamente, com 40, 40, 2, 18. Valores definidos pelo tomador de decisão.

Tabela 1: Resultados obtidos pelos mecanismos de resolução utilizado no SCAP

Instância	SCIP		Gurobi		ILS-RVND
	z	gap	z	gap	gap
1	74000	0	74000	0	0
2	215800	0	215800	0	0
3	177000	0.54	176700	0.06	0
4	414700	1.19	412400	0.53	0
5	220100	0.59	220000	0.12	0
6	560000	3.07	554900	2.04	0
7	491400	2.06	490000	1.7	0
8	478500	2.59	475700	1.76	0
9	1146500	Inf	1062300	23.3	0
10	76300	0	76300	0	0
11	86000	0	86000	0	0
12	861700	Inf	894100	29.1	0
13	409900	4.58	400500	1.87	0
14	154300	4.01	150400	0	0
15	170400	6.03	165000	1.81	0

Como pode ser observado, a heurística alcançou a melhor solução, entre as obtidas pelos

solvers, para todas as instâncias. Vale ressaltar que foi utilizado somente 1/10 do tempo disponibilizado para os *solvers*. O Gurobi mostrou ser mais robusto que o SCIP, obtendo 14 melhores soluções dentre as 15 instâncias testadas.

Embora estes resultados forneçam um bom indicador de desempenho, não podem ser utilizados para obter conclusões mais gerais. A avaliação do desempenho foi realizada utilizando o teste estatístico de Friedman [1937], que objetiva identificar se há diferença estatisticamente significativa entre os algoritmos. Testes estatísticos são usados para determinar se as diferenças observadas são reais ou são devido ao acaso. Considerando o nível de significância de 0,05, verificou-se que há uma diferença significativa entre os algoritmos ($p = 0.007$). Além disso, constatou-se a partir do método multcompare (MATLAB) que o Gurobi e a Heurística superam o SCIP. A Figura 6 ilustra o resultado deste procedimento.

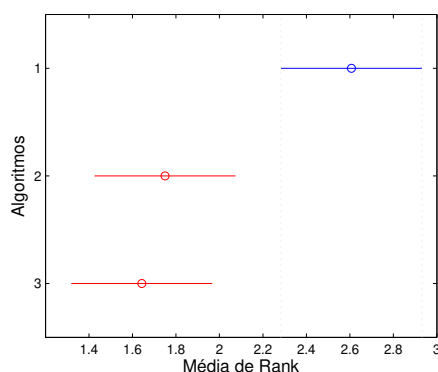


Figura 6: Gráfico de comparações múltiplas

Neste gráfico o eixo das ordenadas determinam os diferentes mecanismos de resolução, no caso 1, 2 e 3 são, respectivamente, o SCIP, Gurobi e a Heurística. Além disso, ele deve ser interpretado da seguinte forma: duas médias são significativamente diferentes se os intervalos são disjuntos, e não são significativamente diferentes, se os seus intervalos se sobrepõem. Ainda, levando em consideração que na atribuição de ranks o maior valor é dado para quem tem o maior valor de função objetivo, é possível chegar a conclusão que o SCIP e o Gurobi, em média, são superiores ao SCIP.

Dentre os resultados, os valores obtidos para o *gap* de duas instâncias, 9 e 12, merecem atenção. O SCIP não foi capaz de obter um limite inferior para estas instâncias, enquanto a solução "incumbente" do Gurobi deixou a desejar na qualidade. Nestes casos, a heurística foi mais eficiente, por exemplo, na instância 12 o valor de função objetivo obtido após 3 minutos de processamento foi igual a 648700, 24% melhor que o Gurobi em 30 minutos. Este mesmo valor só foi obtido pelo Gurobi após 9 horas de processamento. Ainda é necessário investigar as circunstâncias que levaram os *Solvers* a terem dificuldade para resolver estas instâncias.

De forma geral, os resultados obtidos apontam um desempenho satisfatório para todos os mecanismos. Percentualmente, o Gurobi e a Heurística foram superiores, em média, apenas 1% considerando a função objetivo obtida no tempo estipulado.

6. Conclusão

O objetivo deste estudo foi apresentar o SCAP, um software Web para gerenciamento e criação do "ensalamento" em uma IE. Este software utiliza métodos de otimização para criar soluções eficientes. Foi verificado o comportamento de três diferentes mecanismos para obtenção destas soluções. Os resultados apontaram um desempenho satisfatório para ambos os mecanismos.

Considerando que as decisões que envolvem a atribuição de aulas a determinadas salas, na instituição para o qual o software foi desenvolvido, pode demorar semanas e mesmo assim, ao

final do processo, deixar a desejar em muitos critérios. O software apresentado neste trabalho é de grande préstimo, possibilitando obter soluções rápidas e de qualidade. No pior caso, apenas 30 minutos foi necessário para obter soluções de boa qualidade. Como tempo não é um requisito crítico para planejar um "ensalamento", o tempo disponibilizado para os mecanismos de resolução poderia ser ainda maior. Por outro lado, em situações onde salas ficam indisponíveis de forma inesperada, obter uma solução rapidamente pode ser essencial. Neste caso, a heurística proposta seria uma boa opção, por obter boas soluções rapidamente.

Como trabalho futuro, pretende-se deixar o software genérico, atendendo um número maior de IE. Para isso, será necessário implementar as diversas restrições encontradas na literatura e ativa-las sob demanda via software. Além disso, em um futuro próximo o SCAP abordará a parte de gerar automaticamente o quadro de horário, tornando-o uma solução completa para gerenciamento de quadro de horários de uma IE. Assim sendo, metodologias mais eficientes serão utilizados para resolver o problema, tais como o método de geração de colunas e sistemas multi-agentes.

References

- Achá, R. A. e Nieuwenhuis, R. (2010). Curriculum-based course timetabling with sat and maxsat. In *8th International Conference on the Practice and Theory of Automated Timetabling*.
- Aubin, J. e Ferland, J. A. (1989). A large scale timetabling problem. *Computers and Operations Research*, 16:67–77.
- Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., e Qu., R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206:46–53.
- Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., e Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operation Research*, 2007: 176–192.
- Croce, F. D. e Salassa, F. (2014). A variable neighborhood search based matheuristic for nurse rostering problems. In *Annals of Operations Research*, volume 218, p. 185–199.
- Dammak, A., Elloumi, A., e Kamoun, H. (2006). Classroom assignment for exam timetabling. *Advances in Engineering Software*, 37:569–666.
- Elloumi, A., Kamoun, H., Jarboui, B., e Dammak, A. (2014). The classroom assignment problem: complexity, size reduction and heuristics. *Applied Soft Computing*, 14:677–686.
- Freire, J. e Melo, R. (2006). Formulações, heurísticas e um limite combinatório para o problema de alocação de salas de aula com demandas flexíveis. In *Ansi do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, volume 722-729.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701.
- Glassey, C. R. e Mizrach, M. (1986). A decision support system for assigning classes to rooms. *Interfaces*, 16:92–100.
- Gosselin, K. e Truchon, M. (1986). Allocation of classrooms by linear programming. *Journal of the Operational Research Society*, 37:561–569.
- Lach, G. e Lübbecke, M. E. (2012). Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operation Research*, 194:255–272.

- Lourenco, H. R., Martin, O., e Stutzle, T. (2003). *Handbook of Metaheuristics: Iterated Local Search*. Kluwer Academic.
- Martins, A. X., Souza, M. J. F., Carvalho, R., e Silva, T. C. B. (2007). Modelo de otimização do processo de lavra de curto prazo baseado em programação linear por metas aplicada a uma empresa. In *XIV Simpósio de Engenharia de Produção*, volume 1, p. 1–8.
- Mine, M. T., Silva, M. S. A., Souza, M. J. F., P., S. G., e Ochi, L. S. (2006). Iterated local search aplicado à programação de jogos de competições esportivas realizadas em dois turnos completos e espelhados: um estudo de caso. *XIII Congreso Latino-Iberoamericano de Investigación Operativa*, 1:1–6.
- Petrovic, S. e Burke, E. (2004). University timetabling. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, 45.
- Qualizza, A. e Serafini, P. (2005). A column generation scheme for faculty timetabling. *Practice and theory of automated timetabling*, 3616:161–173.
- Schoenauer, M. (2005). An efficient memetic permutation based evolutionary algorithm for real-world train timetabling. In *IEEE CEC05: Congresso n Evolutionary Computation*, volume 1, p. 2752–2759.
- Schonberger, J. D., Mattfeld, D. C., e Kopfer, H. (2004). Algorithm timetabling for sport leagues. *European Journal of Operation Research*, 153:102–116.
- W.Carter, M. e Tovey, A. C. (1992). When is the classroom assignment problem hard? *Operations Research*, 40:28–39.
- Wendt, J. F. e Mullher, F. M. (2017). Problema de alocação de salas no centro de tecnologia - ufsm com um modelo matemático multi-índice. In *Simpósio Brasileiro de Pesquisa Operacional*, p. 1–8.