ANÁLISE DO IMPACTO DA ALEATORIEDADE E MOVIMENTOS NA META-HEURÍSTICA GRASP COM SIMULATED ANNEALING PARA O PROBLEMA DE TABELA-HORÁRIO DO DCOMP-CCENS-UFES

Edmar Hell Kampke, Tulio Machado Müller
Universidade Federal do Espírito Santo
Alto Universitário, s/n, Campus Universitário, Alegre – ES, 29500-000
edmar.kampke@ufes.br, tulio.muller@edu.ufes.br

RESUMO

O Problema de Tabela-Horário em Universidades (PTHU) possui grande relevância no âmbito acadêmico, por ser um processo complexo e desgastante que deve ser realizado todo semestre. Sendo assim, este trabalho analisa o impacto do fator de aleatoriedade e de movimentos na meta-heurística híbrida *Greedy Randomized Adaptive Search Procedure* (GRASP) com *Simulated Annealing* (SA), que já mostrou ser eficiente na resolução do PTHU do Departamento de Computação (DCOMP) do Centro de Ciências Exatas, Naturais e da Saúde (CCENS) da Universidade Federal do Espírito Santo (UFES). Dessa maneira, o movimento *Lecture Move* foi implementado e o PTHU do DCOMP-CCENS-UFES foi resolvido combinando diferentes fatores de aleatoriedade e movimentos. Os resultados obtidos são comparados e indicam que o fator de aleatoriedade e os movimentos usados impactam diretamente na qualidade das soluções.

PALAVRAS CHAVE. Problema de Tabela-Horário em Universidades, Fator de Aleatoriedade, Lecture Move.

13. MH - Meta-heurísticas; 14. OC - Otimização Combinatória

ABSTRACT

The University Timetabling Problem (PTHU) has great relevance in the academic field, being a complex and exhausting process that must be done every semester. Therefore, this work analyzes the impact of the factor of randomness and movements in the hybrid metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP) with Simulated Annealing (SA), which has already proved to be efficient in solving the PTHU of the Computing Department (DCOMP) of the Center for Exact, Natural and Health Sciences (CCENS) of the Federal University of Espirito Santo (UFES). In this way, the movement Lecture Move, was implemented and the PTHU of the DCOMP-CCENS-UFES, was solved by combining different factors of randomness and movements. The results obtained are compared and indicate that the randomness factor and the movements used directly impact the quality of the solutions.

KEYWORDS. University timetabling problem. Factor of randomness. Lecture Move.

13. MH - Metaheuristics; 14. OC - Combinatorial Optimization

https://proceedings.science/p/157428?lang=pt-br

1. Introdução

O Problema de Tabela-Horário (PTH) é um problema de Otimização Combinatória (OC), que deriva do problema de escalonamento e foi definido por Wren [1996] como a alocação, submetida a restrições, de eventos em um número limitado de períodos de tempo e locais, de forma a satisfazer, tanto quanto possível, um conjunto de objetivos estabelecidos.

O PTH pode ser aplicado a diversos tipos de situações, entre os quais pode-se citar: escalas de trabalhadores, escalas de condutores de veículos de transporte, escalas de competições esportivas, e tabelas de horário educacionais, sendo este último o foco deste trabalho, por ser um dos mais estudados na área, conforme pode ser observado em Babaei et al. [2015].

Os PTH's educacionais abordam, por exemplo, a geração de tabela-horário para escolas de ensino médio e universidades. O Problema de Tabela-Horário de Universidades (PTHU) possui diversas variantes. Isso ocorre pois cada instituição de ensino possui diferentes restrições do problema.

Segundo Schaerf [1999], esse problema é classificado como NP-completo para grande parte das variantes, isso significa que uma solução ótima, geralmente, só pode ser encontrada rapidamente para instâncias muito pequenas, o que não é a realidade da maioria das universidades brasileiras. Desta forma, busca-se minimizar, através de uma solução automática, o esforço manual na geração de tabela-horário, bem como isentar o processo de um possível viés na alocação dos horários, por parte dos docentes envolvidos.

Apesar das diferentes formulações, segundo Santos e Souza [2007], os PTH's educacionais possuem uma característica em comum: a separação das restrições em dois grupos, denominados de restrições fortes e restrições fracas. Isso é feito dessa maneira, pois geralmente não é possível encontrar uma solução que atenda todas as restrições impostas.

As restrições fortes são aquelas que não podem ser violadas. Este grupo restringe o conjunto de soluções para impedir situações irreais. Se uma tabela-horário não viola nenhuma restrição forte, ela é considerada uma solução viável. Já as restrições fracas são aquelas cuja satisfação é desejável, mas caso não seja possível atendê-las, a solução não é inviabilizada. Essas restrições possuem pesos para refletir sua importância na qualidade da solução.

Este trabalho leva em consideração o PTHU do caso real do Departamento de Computação (DCOMP) do Centro de Ciências Exatas, Naturais e da Saúde (CCENS) da Universidade Federal do Espírito Santo (UFES), no qual foram inicialmente identificadas 15 restrições por Mariano [2014], e posteriormente adicionadas mais duas restrições por Moreira et al. [2016]. A seguir, as 17 restrições consideradas neste trabalho são apresentadas.

- **Restrições Fortes**:

1. Conflitos de professor: um professor não poderá ministrar mais de uma disciplina no mesmo dia e horário;
2. Conflitos de turmas: uma turma não poderá assistir a mais de uma aula no mesmo dia e horário;
3. Conflitos de salas: uma sala de aula não poderá estar reservada para mais de uma disciplina no mesmo dia e horário;
4. Aulas fora do turno: uma aula não poderá ser alocada fora do turno da oferta (diurno ou noturno).
5. Capacidade da sala: uma turma não poderá ser alocada em uma sala cuja capacidade seja inferior ao número de alunos da turma;

6. Tipo incompatível de sala: as aulas não poderão ser alocadas em uma determinada sala que não é compatível ao tipo solicitado, por exemplo, aulas que deveriam ser realizadas em laboratórios e foram alocadas em salas de aula;
7. "Disciplinas especiais": disciplinas com 3 horas aulas semanais deverão ser alocadas nos três primeiros horários do turno diurno, e nos três primeiros ou três últimos horários do turno noturno, permitindo assim que outras disciplinas possam ser alocadas;

- **Restrições Fracas**:

8. Intervalo de trabalho do professor: o intervalo entre o primeiro e o último dia da semana em que um professor ministrará as aulas deverá ser minimizado;
9. Janelas de horário: intervalos na grade de horários de cada turma, entre duas aulas, deverão ser reduzidos;
10. Período preferencial: as turmas diurnas deverão ter suas disciplinas concentradas no período da manhã ou da tarde. Assim, a quantidade de disciplinas ofertadas fora do turno "preferencial" de cada turma deverá ser minimizada;
11. Aulas seguidas: aulas repetidas de uma disciplina ministradas para uma turma no mesmo dia devem ser evitadas;
12. Intervalo entre períodos: a ocorrência de professores que ministram aula em um dia à noite e no dia seguinte pela manhã deverá ser minimizada;
13. Aulas seguidas de nível "difícil": as aulas de complexidade "difícil" ministradas em horários sequenciais devem ser evitadas;
14. Aulas de nível "difícil" no último horário: aulas de complexidade "difícil" ministradas no último horário de cada dia deverão ser evitadas.
15. Aulas de carga horária par: aulas com 2 ou 4 horas do turno diurno deverão ser alocadas fora do primeiro horário do dia.
16. Aulas alocadas imediatamente antes, ou imediatamente depois, do horário de almoço devem ser evitadas. Ex: Uma aula com carga horária de 2 horas, sendo alocada de 11:00 às 13:00 ou 12:00 às 14:00
17. Aulas com alocação iniciada fora do horário padrão. Ex: Uma aula sendo alocada de 13:00 às 14:00 ou 15:00 às 16:00

A partir desse contexto, o objetivo deste trabalho é dar continuidade ao trabalho de Moreira et al. [2016], e estudar o impacto da escolha do fator de aleatoriedade e de movimentos, entre eles o *Lecture Move*, na meta-heurística híbrida *Greedy Randomized Adaptive Search Procedure* (GRASP) com *Simulated Annealing* (SA), quando aplicado ao PTHU do DCOMP-CCENS-UFES.

Portanto, na Seção 2 é apresentada uma breve revisão da literatura sobre o PTHU. Na Seção 3 a metodologia desenvolvida neste trabalho é descrita. Já a Seção 4 é dedicada à apresentação e discussão dos resultados obtidos nos experimentos computacionais. Por fim, a Seção 5 apresenta as conclusões deste trabalho.

2. Revisão de Literatura

Na década de 60, Gotlieb [1962], através de resoluções a partir de análise combinatória, iniciou estudos relacionados ao problema de tabela-horário de instituições de ensino. Desde então, o tema ganhou atenção de pesquisadores [Schaerf, 1999; Lewis, 2008].

Devido à grande variedade de restrições específicas a cada instituição de ensino, o problema de tabela-horário não possui uma variante única. De acordo com Souza [2000], os primeiros trabalhos utilizavam heurísticas construtivas. Logo em seguida os pesquisadores passaram a utilizar outras técnicas para a resolução do problema, como exemplo, representando o problema com grafos e resolvendo com algoritmos de fluxo [Ostermann e Werra, 1982], ou coloração [Čangalović e Schreuder, 1991]. Outros trabalhos também usaram programação inteira mista [Ferland e Roy, 1985] e programação por restrições [Alcantara et al., 2012].

Em Lewis [2008] pode ser observado que grande parte dos trabalhos tem utilizado meta-heurísticas, tanto pela simplicidade, quanto pelos bons resultados alcançados. SA [Mariano, 2014], Busca Tabu [Machado e Boeres, 2009] e Algoritmos Genéticos [Burke et al., 1994] são as meta-heurísticas mais utilizadas. Em alguns trabalhos se observa também combinação de meta-heurísticas [Moreira et al., 2016].

Na primeira edição do campeonato internacional de tabela-horário (*International Timetabling Competition* - ITC), realizado em 2002 (ITC-2002), Kostuch [2004] desenvolve um algoritmo que constrói a tabela-horário em três etapas. Na primeira etapa é usado um algoritmo de coloração de grafos, que obtém uma solução inicial viável. Na segunda e terceira etapas aplica-se o SA, sendo que em cada etapa é utilizada uma estrutura diferente de vizinhança.

Müller [2009] resolve o problema de tabela-horário de universidades da terceira formulação da segunda edição do ITC, realizado em 2007 (ITC-2007), usando *Conflict-based Statistics* para gerar a solução inicial e *Hill Climbing* combinado com *Great Deluge* e SA para refinamento da solução. Rocha [2013] trata do mesmo problema aplicando a meta-heurística GRASP, sendo testados os métodos *Hill Climbing* e SA como métodos de busca local, e o método *Path-Relinking* também é aplicado, mas para intensificar a busca por soluções de boa qualidade. Segatto [2017] expande o trabalho de Rocha [2013] e implementa novas vizinhanças. Dessa forma, proporcionou um maior entendimento do processo de busca no espaço de soluções e conseguiu melhorar o desempenho do algoritmo.

Ceschia et al. [2012] usam SA para também resolver a terceira formulação do ITC-2007. Os autores conseguiram boas respostas para as instâncias usadas no ITC-2007, e em algumas instâncias foram obtidas melhores soluções que as conhecidas até aquele momento.

Mariano [2014] utilizou a meta-heurística *Adaptive Large Neighborhood Search* (ALNS), aplicada ao PTHU de um caso real do DCOMP-CCENS-UFES. Nesse mesmo problema Moreira et al. [2016] apresentou uma proposta de utilizar a meta-heurística GRASP, com o SA na fase de busca local, tendo encontrado soluções melhores do que as feitas manualmente pelos coordenadores de curso. Carvalho et al. [2015] resolve o mesmo problema que Mariano [2014] e Moreira et al. [2016], porém usa um método guloso para construir uma solução inicial que é refinada através da meta-heurística SA.

3. Metodologia

Nessa seção é apresentada a metodologia proposta para o desenvolvimento deste trabalho. Além disso, é apresentada também a forma como o problema foi representado computacionalmente e a função objetivo usada para calcular a qualidade de uma solução do PTHU abordado.

3.1. Modelagem do Problema

A modelagem usada para representar o PTHU do DCOMP-CCENS-UFES foi baseada nos trabalhos de Mariano [2014] e Moreira et al. [2016], bem como as instâncias que estes trabalhos utilizaram (2013/2 e 2016/1). O problema é modelado com estruturas definidas pelo identificador de cada elemento relevante e das características seguintes ao identificador.

As estruturas são armazenadas em vetores que possibilitam estabelecer as relações entre os conjuntos de elementos que são trabalhados. Desse modo, as estruturas mais simples são: salas, professores, disciplinas, turmas e horários. Além dessas, é estabelecida uma estrutura denominada oferta, que promove a relação entre todas as outras.

Com todos os dados armazenados nas estruturas previamente definidas, o algoritmo inicia o processo de criação de uma solução, que é representada com uma matriz tridimensional. A primeira dimensão representa as salas de aula, a segunda representa os horários disponíveis em um dia e a terceira os dias da semana.

3.2. Função Objetivo

Cada tabela-horário gerada recebe uma nota que reflete sua qualidade, a função f que calcula essa nota é chamada de Função Objetivo (FO). Cada restrição violada aumenta o valor da função objetivo de acordo com o peso da restrição. A melhor solução para um problema de tabela-horário é aquela que seja viável e que minimiza o valor da função objetivo.

Seja *S* uma solução do PTHU abordado, e considerando que $f_{RFt}(S)$ é o número total de violações das restrições fortes de *S*, então *S* é uma solução viável, se todas as restrições fortes são satisfeitas, ou seja, $f_{RFt}(S) = 0$. De acordo com as restrições propostas por Mariano [2014] e posteriormente expandidas por Moreira et al. [2016], a função objetivo do PTHU do DCOMP-CCENS-UFES é calculada pela seguinte fórmula:

$$f(S) = \sum_{i=8}^{17} \omega_i \cdot \beta_i(S)$$

sendo que $\beta_i(S)$ representa o número total de violações de cada restrição fraca i. O vetor $\omega = [\omega_8, \omega_9, ..., \omega_{17}]$ contém os pesos das 10 restrições fracas consideradas por Moreira et al. [2016] e apresentadas na Seção 1 deste trabalho.

3.3. GRASP

A meta-heurística GRASP, apresentada por Feo e Resende [1989] foi inicialmente utilizada para tratar o problema de cobertura de conjuntos. De acordo com Rocha [2013], o GRASP já foi aplicado com sucesso em diversos problemas de otimização desde sua proposta inicial.

O algoritmo GRASP é um procedimento iterativo dividido em duas fases, que são detalhadas nas próximas subseções. Na primeira fase constrói-se uma solução inicial, e na segunda fase aplica-se uma busca local para melhorá-la. A resposta final é a melhor obtida após a execução de todas as iterações [Feo e Resende, 1995]. Para essa meta-heurística são utilizados dois parâmetros principais: o número máximo de iterações (Max_{Iter}) e o fator de aleatoriedade (α).

3.3.1. Construção da solução inicial

Inicialmente, uma solução vazia (sem ofertas alocadas) para o PTHU do DCOMP-CCENS-UFES é criada. Nessa solução inicial, todas as células de cada tabela-horário terão valor negativo, sendo -1 se o horário estiver disponível e -2 se naquele horário já há uma oferta de outro departamento da UFES alocada.

A partir disso, uma lista de ofertas, contendo todas as ofertas, é criada. Essa lista é ordenada de forma decrescente pelo número de vagas. A partir da lista inicial, são feitas simulações de alocação em todas as posições da matriz de salas, com a oferta que possui o maior número de vagas. Cada simulação tem seu custo calculado, e cada uma dessas simulações é inserida em uma lista denominada Lista de Candidatos (LC).

A LC possui a posição em que a simulação de alocação foi efetuada e seu custo de alocação. A partir disso, a LC serve de base para a formação de uma outra lista, denominada Lista Restrita

de Candidatos (LRC), que é composta por um intervalo da LC, depois dela ter sido ordenada crescentemente pelo custo de alocação das simulações. Esse intervalo é definido de acordo com o fator de aleatoriedade α da seguinte forma: $[c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$ em que c^{min} representa o menor custo de alocação, c^{max} representa o maior custo de alocação, e $\alpha \in [0, 1]$.

Quando $\alpha = 1$, LC tem seus elementos totalmente selecionados para LRC. Se $\alpha = 0$, LC terá apenas o primeiro elemento selecionado para LRC. Em outra palavras, quando $\alpha = 1$ a escolha do horário é feita de maneira totalmente aleatória, caso $\alpha = 0$, o horário de LC que possui o menor custo será sempre escolhido, o que representa uma escolha totalmente gulosa.

LC da oferta com maior nº de vagas

Posição	[1][2][0]	[3][3][2]	[1][2][2]	[3][2][3]	[7][5][1]	[1][4][4]
Custo	400	500	566	700	885	990

⇓

LRC definida de acordo com α

Posição	[1][2][0]	[3][3][2]	[1][2][2]
Custo	400	500	566

⇓

Escolha aleatória: candidato[1], posição[3][3][2]

Figura 1: Exemplo de uma iteração do construtor de soluções

	Sala X				
Horário	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**
08:00-10:00	-1	-1	-1	-1	-1
10:00-12:00	-1	-1	92	-1	-1
12:00-14:00	-1	-1	-1	-1	-1
14:00-16:00	-1	-1	-1	-1	-1
16:00-18:00	-1	-1	-1	-1	-1
18:00-20:00	-1	-1	-1	-1	-1
20:00-22:00	-1	-1	-1	-1	-1

Tabela 1: Tabela-horário depois de uma iteração do procedimento de criar soluções.

Na Figura 1 é exibido um exemplo de uma iteração da fase de construção de uma solução inicial no método GRASP. Nesse exemplo, a LC da oferta ainda não alocada e que possui maior número de vagas é criada com seis possíveis horários para alocação ordenados crescentemente pelo custo de alocação, calculado durante cada simulação. Em seguida, é formada a LRC, com apenas três horários, supondo o fator de aleatoriedade $\alpha = 0,5$. Dentre esses três horários é feita uma escolha aleatória para definir em qual horário a oferta é alocada. Essa alocação é representada na Tabela 1 com a alocação da oferta cujo identificador é 92 (destaque em laranja).

Dessa maneira, o valor de α define o tamanho da LRC, sendo que após essa definição, um horário de LRC é escolhido aleatoriamente e a oferta é acrescentada à solução e consequentemente removida da lista de ofertas. O procedimento é repetido até que todas as ofertas sejam alocadas. Portanto, a construção da solução inicial termina quando todas as ofertas estiverem inseridas na tabela-horário.

3.3.2. Busca Local

O objetivo da busca local é melhorar a solução inicialmente construída. A estratégia de busca local utilizada foi a meta-heurística SA, pois ela se mostrou satisfatória em outros trabalhos, como de Rocha [2013], Mariano [2014] e Carvalho et al. [2015].

A meta-heurística SA tem como inspiração o processo da metalurgia conhecido como recozimento, em que derrete-se um metal e ele é esfriado lentamente até a solidificação, de forma controlada, para que sua estrutura se mantenha organizada e equilibrada [Kirkpatrick, 1984].

Os cinco principais parâmetros que o SA possui são: a solução inicial (S), a temperatura inicial (T_i), a temperatura final (T_f), a taxa de resfriamento (β) e o número de soluções vizinhas (N_v) que deverão ser geradas a cada iteração.

O algoritmo começa com uma temperatura inicial que vai sendo resfriada até chegar na temperatura final. Em cada temperatura (iteração), começando com a temperatura inicial T_i, são geradas N_v soluções vizinhas. Se a solução vizinha gerada tem valor de f menor que a da solução atual, esta é atualizada. Se a solução vizinha tiver valor de f maior, ela pode ser aceita com uma probabilidade $e^{-\Delta f/T}$, em que Δf é a diferença entre os valores das FO's da solução vizinha e da solução atual, e T é a temperatura atual. Quanto maior Δf, e menor a temperatura atual T, menor será a chance de aceitar uma solução vizinha. Tipicamente, o algoritmo aceita grande diversificação de soluções no início, quando a temperatura está alta. À medida que a temperatura decresce, poucas soluções piores são aceitas e assim é intensificada a busca em uma determinada vizinhança. Ao final de cada iteração, a temperatura T é multiplicada pelo valor β, sendo $\beta \in [0, 1]$ a taxa de resfriamento. O Algoritmo termina quando $T < T_f$.

Para gerar soluções vizinhas da solução atual, neste trabalho foram utilizados três movimentos. Os movimentos clássicos *Move* e *Swap* implementados por Moreira et al. [2016], e o movimento *Lecture Move*, baseado neles e proposto por Müller [2009]. O movimento *Lecture Move* apresentou bom desempenho na análise de movimentos feita por Kampke et al. [2017], surgindo assim a motivação para sua implementação.

De acordo com Schaerf [1999], os movimentos *Move* e *Swap* são clássicos e estão entre os mais utilizados para PTH's. O movimento *Move* consiste em mover uma oferta escolhida aleatoriamente para uma posição não ocupada na tabela-horário, também escolhida aleatoriamente. Nota-se que no *Move*, a oferta selecionada pode ser movida para qualquer posição vazia da tabela-horário, desde que essa mudança não cause uma inviabilidade na solução, ou seja, desde que não infrinja nenhuma restrição forte.

Já o movimento *Swap* consiste na troca entre duas ofertas escolhidas aleatoriamente na tabela-horário. Da mesma forma que no *Move*, as ofertas selecionadas podem ser trocadas, desde que essa mudança não cause uma inviabilidade na solução.

Por fim, o movimento *Lecture Move* consiste na troca entre duas ofertas ou entre uma oferta e um horário vazio. Em outras palavras, o *Lecture Move* corresponde a uma combinação de *Move* e *Swap*, em que são escolhidos aleatoriamente dois horários na tabela-horário. Caso apenas um deles esteja vazio é realizado um *Move*, mas se ambos possuírem ofertas alocadas é realizado um *Swap*.

4. Resultados Computacionais

Esta seção apresenta os experimentos computacionais e os resultados alcançados neste trabalho, constando as análises comparativas com outro trabalho, bem como as soluções manuais elaboradas pelos coordenadores de curso do DCOMP-CCENS-UFES. Mostra também as escolhas dos parâmetros do GRASP, os parâmetros específicos da busca local *SA* e os movimentos utilizados para a geração de soluções vizinhas.

4.1. Escolha de Parâmetros

O algoritmo GRASP possui dois parâmetros: o número máximo de iterações Max_{Iter} e o valor α, que define a forma como a construção da solução inicial será conduzida. Conforme dito na Subseção 3.3.1, se o parâmetro α tem seu valor muito próximo a 0 (zero), o algoritmo de construção inicial tem comportamento predominantemente guloso, ou seja, possui a tendência de produzir soluções de boa qualidade, porém pouco diversificadas. Caso α seja mais próximo de 1 (um), as soluções possuem características mais aleatórias, porém com a desvantagem de geralmente possuirem um valor de função objetivo mais alto. Para o fator de aleatoriedade α foram escolhidos 5 valores distintos: 0; 0,25; 0,5; 0,75 e 1, para serem usados nos experimentos. Para efeitos de comparação com o trabalho de Moreira et al. [2016], o valor de $\alpha = 0.15$ também foi testado e o parâmetro Max_{Iter} foi definido pelo tempo de execução: 500 segundos.

Para o *SA*, os parâmetros foram calibrados de forma a permitir um certo grau de diversificação no início da busca e maior intensificação no final do processo. Para cada valor de α e para cada instância (2013/2 e 2016/1) o *SA* foi testado com dois conjuntos de movimentos: *Move* e *Swap*, estes, com 50% de probabilidade cada, e *Lecture Move*.

Na tabela 2 são apresentados os parâmetros definidos durante a calibração para o algoritmo *SA*.

Parâmetro	Descrição	Valor
T_i	Temperatura inicial	1000
T_f	Temperatura de congelamento	0.01
β	Taxa de resfriamento	0.975
N_v	Número máximo de iterações	500

Tabela 2: Parâmetros e valores dos algoritmos

Os valores das penalizações (vetor ω) das restrições fracas da função objetivo, usadas neste trabalho, também foram as mesmas adotadas por Moreira et al. [2016] e apresentadas na Tabela 3.

ω_i	Valor
ω_8	10
ω_9	20
ω_{10}	4
ω_{11}	600
ω_{12}	10
ω_{13}	10
ω_{14}	10
ω_{15}	500
ω_{16}	200
ω_{17}	150

Tabela 3: Valores das Penalizações

4.2. Detalhes de Implementação

Os algoritmos descritos neste trabalho foram implementados na linguagem C++ e os testes computacionais foram realizados em uma máquina com CPU Intel Core i5-7200U CPU @ 2.50GHz com 12,0 GB de mémória RAM e sistema operacional Microsoft Windows 10.

4.3. Análise dos Resultados

O algoritmo foi executado 10 vezes para cada instância (2013/2 e 2016/1), para cada valor de α (0; 0,25; 0,5; 0,75 e 1) e para cada conjunto de movimentos da busca local (*Move*/*Swap* e *Lecture Move*). Ao final das execuções, todas as soluções obtidas apresentaram-se viáveis, ou seja, não violaram nenhuma restrição forte para o problema. É importante notar que, apesar da execução ocorrer durante 500 segundos, os valores de tempo apresentados nas tabelas abaixo são os tempos em que foi alcançada a melhor solução.

Nas Tabelas 4 e 5, pode-se verificar que, mantendo $\alpha = 0,15$, mesmo valor usado por Moreira et al. [2016], para ambos os semestres, a solução com o movimento *Lecture Move* apresentou melhores resultados. Para 2013/2, a média dos resultados foi $18,37\%$ melhor do que a solução com os movimentos *Move/Swap*, enquanto para o semestre 2016/1, a melhora foi de $20,08\%$.

	Move/Swap	*Lecture Move*
Melhor Solução	406	306
Média	463,6	331,4
Tempo	267,012	255,710

Tabela 4: Melhores Soluções, médias das soluções e tempo para 2013/2 com α = 0,15

	Move/Swap	*Lecture Move*
Melhor Solução	278	208
Média	298,8	238,0
Tempo	231,525	301,577

Tabela 5: Melhores Soluções, médias das soluções e tempo para 2016/1 com α = 0,15

A melhoria da solução com o *Lecture Move* em relação ao *Move/Swap* pode ser explicada pela eliminação da escolha aleatória de qual movimento seria realizado. Com a introdução do *Lecture Move*, primeiro são escolhidos os horários, se os dois horários escolhidos já possuem aulas alocadas, tenta-se realizar o *Swap*, mas se somente um horário já está alocado, tenta-se fazer o *Move*.

		0	0,15	0,25	0,5	0,75	1
2013/2 *LM*	Melhor Solução	268	306	324	306	296	302
	Média	320,8	331,4	351,8	347,8	347,0	350,0
	Tempo	183,860	255,710	261,245	231,892	363,247	197,625
2016/1 *LM*	Melhor Solução	214	208	200	220	198	214
	Média	247,4	238,0	234,4	242,8	233,4	243,8
	Tempo	342,737	301,577	215,537	271,353	185,639	261,776

Tabela 6: Melhores soluções, média das soluções e tempo para o Movimento *Lecture Move* com variação de α

Na Tabela 6 são mostrados os melhores resultados, a média dos valores e a média de tempo utilizando o movimento de busca local *Lecture Move* (LM) e variando o valor de α. Pode-se observar

que a melhor média de soluções para o semestre 2013/2 foi obtida com $\alpha = 0$, com a média dos resultados de 320,8, enquanto que para 2016/1 foi com $\alpha = 0{,}75$, com a média dos resultados de 233,4.

Para o semestre 2013/2, a melhor solução encontrada foi com $\alpha = 0$, isso significa que uma estratégia totalmente gulosa foi melhor aproveitada para essa instância. Já para o semestre 2016/1, com a melhor solução possuindo $\alpha = 0{,}75$, percebe-se que uma maior aleatoriedade foi mais benéfica para essa instância.

Na Tabela 7 pode-se verificar a melhor solução, os tempos médios de execução e a média de f obtidas neste trabalho, bem como da solução proposta por Moreira et al. [2016] e o valor de f da solução construída manualmente pelos coordenadores de curso.

		Manual	Moreira et al. [2016]	GRASP+SA+LM
2013/2	Melhor Solução	760	406	268
	Média	-	463,6	320,8
	Tempo	-	267,012	206,482
2016/1	Melhor Solução	2374	278	198
	Média	-	297,8	233,4
	Tempo	-	231,525	185,639

Tabela 7: Melhores soluções, média das soluções e tempo para o Movimento *Lecture Move*

Para o semestre de 2013/2 a solução manual obtida pelo DCOMP resultou em $f = 760$, já para o trabalho de Moreira et al. [2016] a melhor solução encontrada foi com $f = 466$. Assim é possível verificar que a solução apresentada por este trabalho alcançou uma melhora percentual de $57,79\%$ em relação à solução manual de 2013/2 e uma melhora de $20,98\%$ quando comparada à solução de Moreira et al. [2016].

Já para o semestre de 2016/1, a solução manual obtida pelo DCOMP teve seu valor de f calculada em 2374. Assim, pode-se verificar que a solução obtida por este trabalho apresenta uma melhora de 90.17% comparada com a solução manual e de $21,63\%$ quando comparada à solução de Moreira et al. [2016].

5. Conclusões

Este trabalho teve como objetivo o estudo do impacto da aleatoriedade e de movimentos na meta-heurística GRASP com SA para a resolução do Problema de Tabela-Horários de Universidades, considerando o caso real do Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo.

Neste trabalho foi implementado o movimento *Lecture Move*, que é a combinação dos clássicos movimentos *Move* e *Swap*, sendo executado para diferentes fatores de aleatoriedade (α).

Os resultados obtidos foram comparados com os resultados do trabalho de Moreira et al. [2016], bem como as soluções manuais elaboradas pelos coordenadores de curso do DCOMP-CCENS-UFES, apresentando soluções melhores para ambos os casos.

Portanto, os resultados obtidos por este trabalho mostraram-se ser mais eficazes que os existentes, podendo ser aplicado para a geração de tabela-horários em semestres futuros, e assim obter tabelas de horários aproximadamente 78,2% melhores, ou seja, com menos violações de restrições, do que as soluções atuais.

Como trabalhos futuros, sugere-se a implementação de novos movimentos para problemas de tabela-horário, como exemplo, o clássico *Cadeia de Kempe*, além de movimentos específicos para

restrições fracas, assim como proposto no trabalho de Müller [2009] para o ITC-2007. Além disso, outros trabalhos futuros poderiam realizar a implementação de outras meta-heurísticas como Busca Tabu e *Iterated Local Search*, que também podem usar o *SA* como busca local.

Referências

Alcantara, M. F., de Sá, C. C., e Heinen, M. R. (2012). Modelagem do problema de balanceamento de currículo acadêmico utilizando programação em lógica por restrições. In *Anais do XXI Seminário de Computação*, p. 75–83, Blumenau-SC. Academia.

Babaei, H., Karimpour, J., e Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43 – 59.

Burke, E. K., Elliman, D., e Weare, R. (1994). A genetic algorithm based university timetabling system. In *Proceedings of the 2nd east-west international conference on computer technologies in education*, p. 35–40, Crimea-Ukraine. Citeseer.

Čangalović, M. e Schreuder, J. A. (1991). Exact coloring algorithms for weighed graphs applied to timetabling problems with lectures of different lenghths. *European Journal of Operational Research*, 51:248–258.

Carvalho, A. S., Mariano, G. P., Kampke, E. H., e Mauri, G. R. (2015). Simulated Annealing aplicado ao problema de programação de horários do CCA-UFES. In *Anais do XVIII SPOLM*, p. 341–352, Rio de Janeiro-RJ. Escola de Guerra Naval.

Ceschia, S., Di Gaspero, L., e Schaerf, A. (2012). Design, Engineering, and Experimental Analysis of a Simulated Annealing Approach to the Post-Enrolment Course Timetabling Problem. *Computers & operations research*, 39(7):1615–1624.

Feo, T. A. e Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Ferland, J. A. e Roy, S. (1985). Timetabling problem for university as assignment of activities to resources. *Computers & operations research*, 12(2):207–218.

Gotlieb, C. C. (1962). The construction of class-teacher time-tables. In *Communications of the ACM*, volume 5, p. 312–313, New York-NY. ACM.

Kampke, E. H., Segatto, E. A., Boeres, M. C. S., Rangel, M. C., e Mauri, G. R. (2017). Neighborhood analysis on the university timetabling problem. In Gervasi, O., Murgante, B., Misra, S., Borruso, G., Torre, C. M., Rocha, A. M. A., Taniar, D., Apduhan, B. O., Stankova, E., e Cuzzocrea, A., editors, *Lecture Notes in Computer Science*, p. 148–164, Cham-China. Springer International Publishing.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5):975–986.

Kostuch, P. (2004). The university course timetabling problem with a three-phase approach. In Burke, E. e Trick, M., editors, *Practice and Theory of Automated Timetabling*, p. 109–125, Berlin–Heidelberg. Springer.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR spectrum*, 30(1):167–190.

Machado, A. M. e Boeres, M. C. S. (2009). Uma proposta de formulação do problema de programação de tabela-horário de exames de toronto via coloração de grafos e sua resolução pelo algoritmo de busca tabu. In *Anais do XLI SBPO*, p. 2910–2921, Rio de Janeiro-RJ. SOBRAPO.

Mariano, G. P. (2014). Resolução do problema de programação de horários de disciplinas do CCA-UFES utilizando a meta-heurística ALNS. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal do Espírito Santo.

Moreira, L. V., Monteiro, R. C., Kampke, E. H., e Mauri, G. R. (2016). Meta-heurística GRASP para o Problema de Tabela-horário de Disciplinas do Departamento de Computação do CCA-UFES. In *Anais do XLVIII SBPO*, p. 2171–2182, Rio de Janeiro-RJ. SOBRAPO.

Müller, T. (2009). ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1):429.

Ostermann, R. e Werra, D. (1982). Some experiments with a timetabling system. *Operations-Research-Spektrum*, 3(4):199–204.

Rocha, W. S. (2013). Algoritmo GRASP para o problema de tabela-horário de universidades. Mestrado em informática, Universidade Federal do Espírito Santo.

Santos, H. G. e Souza, M. J. F. (2007). Programação de horários em instituições educacionais: formulações e algoritmos. In *Anais do XXXIX SBPO*, p. 2827–2882, Rio de Janeiro-RJ. SOBRAPO.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial intelligence review*, 13(2):87–127.

Segatto, E. A. (2017). Um algoritmo GRASP com Cadeia de Kempe Aplicado ao Problema de Tabela-horário para Universidades. Mestrado em informática, Universidade Federal do Espírito Santo.

Souza, M. J. F. (2000). *Programação de horários em escolas: uma aproximação por metaheurísticas*. Doutorado em ciências em engenharia de sistemas e computação, Universidade Federal do Rio de Janeiro.

Wren, A. (1996). Scheduling, timetabling and rostering — a special relationship? In Burke, E. e Ross, P., editors, *Practice and Theory of Automated Timetabling*, p. 46–75, Berlin–Heidelberg. Springer.

https://proceedings.science/p/157428?lang=pt-br