Algoritmo genético para resolução do problema de university course timetabling para instituições de ensino superior privadas: Um estudo de caso na FAGOC

Italo Luis da Silva
Faculdade Governador Ozanam Coelho
Rua Dr. Adjalme da Silva Botelho, 20 – Bairro Seminário, Ubá – MG, 36500-000
italo@maleldil.com

Pedro Henrique Mázala Machado
Faculdade Governador Ozanam Coelho
Rua Dr. Adjalme da Silva Botelho, 20 – Bairro Seminário, Ubá – MG, 36500-000
pedroh.mazala@gmail.com

Saulo Cunha Campos
Faculdade Governador Ozanam Coelho
Rua Dr. Adjalme da Silva Botelho, 20 – Bairro Seminário, Ubá – MG, 36500-000
saulo@fagoc.br

RESUMO

Este artigo aborda o problema de alocação de horários, disciplinas, professores e turmas numa instituição de Ensino Superior privada. O estudo foi baseado em uma instituição real de ensino de grau superior, particular, do estado de Minas Gerais. Foram consideradas diversas restrições aplicáveis ao modelo de negócio de instituições de ensino superior particular. A resolução do problema é feita através da aplicação combinada de um Algoritmo Genético (AG) e uma meta-heurística GRASP em duas etapas. A primeira etapa consiste na geração de uma matriz de horários (viável) que combina professores, disciplinas e *slots* de tempo, através da aplicação de um AG. Já na segunda etapa é realizada a alocação de disciplinas para todos os alunos da instituição com a meta-heurística GRASP. O objetivo em questão é maximizar o número de aulas cursadas pelos alunos. Avalia-se a qualidade das soluções através da soma das horas de aula semanais de cada aluno da instituição. Foi realizado um experimento computacional para extrair a melhor configuração do algoritmo proposto. Os dados utilizados são reais e fornecidos pela própria instituição. Os resultados mostram que a aplicação da solução proposta é viável na instituição em questão, pois os resultados gerados são superiores aos que são atualmente encontrados, em termos de qualidade e tempo.

PALAVRAS CHAVE. *University Course Timetable*, Algoritmo Genético, GRASP.

ABSTRACT

This article approaches the problem of allocating time slots, courses, teachers and classes at a private higher education institution. The study was based on a real private higher education institution located on Minas Gerais state. Several restrictions applicable to the private higher education institution business model were considered. The solution of the problem is achieved by the combined application of a Genetic Algorithm (GA) and a GRASP meta heuristic in two phases. The first phase consists in the generation of a viable schedule matrix that combines teachers, courses and time slots, created by the GA. The objective of the second phase is to allocate courses for every student on the institution with the GRASP meta heuristic. The goal is to maximize the sum of the weekly studied hours for every student on the institution. Different computational experiments were performed for configuring the proposed algorithm. The data used are real and provided by the institution. The results show that applying the proposed solution is viable in the studied institution, as the generated results are superior to the currently experienced.

KEYWORDS. University Course Timetable, Genetic Algorithm, GRASP.

1. Introdução

Segundo Schaerf (1999), o problema de *timetabling* consiste em agendar uma sequência de aulas entre professores e estudantes em um período de tempo delimitado, satisfazendo um conjunto de restrições. O problema vem sendo estudado desde 1963, por Gotlieb, e se aplica a instituições de ensino tanto superior quanto básico, assim como se aplica a instituições privadas e públicas.

Esse problema existe em qualquer instituição de ensino, mas se torna cada vez mais complexo conforme ela cresce, uma vez que ele pertence à classe NP-Difícil, para qual não se conhecem algoritmos de tempo de polinomial para a obtenção de soluções ótimas [Levitin 2003].

O problema de *timetabling* é pesquisado há mais de 50 anos, começando por Gotlieb em 1963. Técnicas pioneiras foram desenvolvidas com base na forma como uma pessoa desenvolvia um *timetable* [Schmidt e Strohlein 1979], utilizando melhorias sucessivas. Também existem técnicas baseadas em coloração de grafos para a resolução do problema [Welsh e Powel 1967]. Even, em 1976, aplica uma técnica de encontrar uma sequência maximal de emparelhamentos no grafo bipartido composto por professores e turmas.

Esse trabalho aborda um subproblema do *Timetabling*, o *University Course Timetabling* introduzido por [De Werra 1985]. O *University Course Timetabling* resolve o problema de criar atribuições de *slots* de tempo, salas de aula, professores e disciplinas, de forma a impedir que uma sala seja ocupada por duas disciplinas simultaneamente, ou então que dois professores lecionem a mesma aula, ou que um professor lecione duas disciplinas no mesmo *slot* de tempo [Schaerf 1999]. O indicador de qualidade de uma solução varia. [De Werra 1985] utiliza o somatório das preferências de uma determinada alocação disciplina – professor acontecer. Autores como [Eiselt e Laporte 1987] e [Aubin e Ferland 1989] adotam restrições *soft* e as utilizam para avaliar a qualidade da solução, utilizando critérios de preferências dos alunos ou espaçamento entre aulas. [Tripathy 1984] considera uma matriz de conflitos entre estudantes cursando dois cursos i e j simultaneamente. A função objetivo considera a minimização do número de casos onde i e j são agendados ao mesmo tempo.

[Tripathy 1984] utiliza programação inteira, aplicando uma técnica de Relaxamento Lagrangiano. [Ferland e Roy 1985] formulam o problema como um de alocação, resolvendo-o por uma redução a um problema de alocação quadrático. [Chahal e De Werra 1989] utilizaram técnicas de resolução de fluxo em redes. Muitos autores após [Eiselt e Laporte 1987] dividiram as restrições entre *hard* e *soft*, onde as restrições *hard* definem as soluções factíveis, e as restrições *soft* são inclusas na função objetivo, diferenciando a qualidade das soluções. [Hertz 1991] aplica Busca Tabu, e no ano seguinte [Hertz 1992] estende o algoritmo para blocos de durações diferentes. [Paecther et al. 1994] utilizam algoritmos genéticos. [Carrasco e Pato 2000] utilizam um algoritmo genético multiobjetivo, combinando dois pontos de vista: o orientado aos estudantes e o orientado aos professores. [Gaspero e Schaerf 2002] utilizam Busca Tabu, explorando a utilização de uma busca com várias vizinhanças. [Casey e Thompson 2002] utilizam um algoritmo GRASP em dois estágios. [Daskalaki e Birbas 2005] formulam um outro modelo para o problema, utilizando também de restrições *soft* na função objetivo. [Burke e Qu 2009] exploram um *framework* de hiper-heurísticas baseadas em grafos, pesquisando algoritmos de buscas locais baseadas em algoritmos de coloração de baixo nível. [Abudllah e Turabieh 2012] também aplicam uma Busca Tabu, utilizando estruturas de vizinhança independentes de domínio, penalizando vizinhanças incapazes de gerar soluções melhores. [Fong et al. 2014] propõem uma combinação de uma meta-heurística colônia de abelhas com uma política de assimilação para guiar o processo de busca.

É importante levar em consideração o contexto do problema da instituição de ensino em questão, pois o problema pode variar de acordo com a instituição e o seu modelo de negócio, como, por exemplo, em instituições de ensino públicas e privadas. As públicas normalmente possuem um modelo aberto, em que disciplinas são oferecidas e os cursos são construídos em volta delas, enquanto os alunos têm liberdade para escolher quais disciplinas querem cursar, e em quais horários. Como as disciplinas são livres, não há nenhuma obrigatoriedade de salas onde ela deva ser lecionada, e essa alocação também faz parte do processo como um todo; é possível que uma disciplina seja lecionada em várias salas diferentes ao longo da semana. Já em algumas instituições

privadas (como a estudada nesse trabalho) adota-se um modelo fechado, em que as disciplinas que o aluno irá cursar são alocadas pelo coordenador do curso. Esse conjunto de disciplinas alocados para um aluno é conhecido pelo nome "grade" e, posteriormente, é aprovado pelo aluno. Além disso, as disciplinas estão ligadas a turmas dos cursos, não sendo completamente livres como as das instituições públicas. Como resultado, as disciplinas não devem ser alocadas em salas, pois elas possuem uma localização fixa: a sala da turma à qual é atribuída. Sendo assim, o professor fica encarregado de trocar de sala e não o aluno. Os professores, peculiarmente nas instituições privadas, podem ministrar diversas disciplinas e não há obrigatoriedade de que uma disciplina seja ministrada sempre pelo mesmo professor, enquanto nas instituições públicas é comum um professor ser contratado para uma ou mais disciplinas especificas.

Assim, este estudo é voltado para pequenas e médias universidades privadas e sua importância se dá devido ao tempo gasto para gerar manualmente o horário do curso e a alocação de disciplinas no semestre de cada aluno, tarefa normalmente atribuída ao coordenador de cada curso. Uma forma automatizada da resolução de tal problema não só reduz drasticamente o tempo de tal tarefa, reduzindo seu custo, mas também pode obter soluções (melhores) que uma pessoa deixaria passar despercebido [Schaerf 1999].

Este trabalho é um estudo de caso aplicado diretamente à realidade da instituição FAGOC – Faculdade Governador Ozanam Coelho. A FAGOC está localizada na cidade de Ubá, Zona da Mata de Minas Gerais e foi fundada em 13 de setembro de 1999 com a missão de participar ativamente do crescimento e desenvolvimento local e regional através da oferta de cursos de formação superior. Atualmente conta com 1860 alunos e 12 cursos [FAGOC 2016].

Foi realizada uma entrevista com os coordenadores dos cursos da faculdade estudada, e descobriu-se que a criação manual de uma matriz de horários demanda em média 24 dias de trabalho de cada coordenador, tempo que só tende a aumentar com o crescimento da instituição. Considerando-se que 24 dias representam um mês de trabalho, que o salário médio dos coordenadores é de R$ 6.500,00 e que a instituição possui 12 cursos, a alocação automatizada economizaria cerca de R$ 78.000,00 por semestre. Atualmente a faculdade não conta com um sistema automatizado para estas tarefas, o que torna a implementação de um ainda mais impactante, pois a redução de uma tarefa tão dispendiosa possibilitaria ao coordenador um aproveitamento de seu tempo em atividades mais nobres.

Esse estudo propõe resolver esse problema utilizando um algoritmo genético (GA) [Holland 1975], que evolui possíveis soluções – matrizes de horários completos – tendo como sua função objetivo o somatório das horas que cada aluno possui em sua grade de disciplinas (obtidas a partir de um horário em questão), geradas pela aplicação da meta heurística GRASP [Feo e Resende 1995]. A seção 2 define formalmente o problema. A seção 3 discute os detalhes de implementação dos algoritmos e a seção 4 expõe os resultados dos experimentos computacionais.

2. Definição formal do problema

O problema investigado nesse trabalho é a alocação de professores e disciplinas a horários nos cursos de uma instituição de ensino superior privada, tendo como base de estudo a instituição FAGOC. Esse problema é resolvido em duas etapas:

1) Geração de uma matriz de horários viável com alocação de professores e disciplinas;
2) Geração das grades de disciplinas para todos os alunos da instituição.

A primeira etapa consiste em atribuir a cada *slot* de tempo, para cada período de um curso, uma disciplina, e, atribuir a cada disciplina um professor para lecioná-la, de forma que toda a carga horária de todas as disciplinas sejam consideradas, respeitando as diversas restrições do problema. As restrições são descritas formalmente na seção 2.1.

A segunda etapa consiste em gerar, para todos os alunos da instituição, uma grade de disciplinas no semestre. Esse processo consiste em escolher a melhor combinação dentre as disciplinas que estão disponíveis na matriz de horários e que o aluno está apto a cursar. O termo "melhor", aqui utilizado, é considerado em função da maximização da carga horária. A condição para que o aluno esteja apto compreende que não tenha que cursar duas disciplinas que

compartilham um mesmo *slot*, nem que sejam alocadas disciplinas para as quais as condições de pré-requisitos e co-requisitos sejam desrespeitadas.

Como função objetivo é utilizada a soma da quantidade de horas semanais das alocações de disciplinas de cada aluno da instituição. Assim, neste trabalho, uma solução é a matriz de horários gerada e sua qualidade se dá pelo somatório de grades de disciplinas que podem ser alocadas para todos os alunos da instituição. A solução é representada por uma estrutura de dados formada por uma matriz tridimensional, onde o eixo Z são os períodos dos cursos (por exemplo, o 5º período de Administração ocupa uma linha, o 6º ocupa outra, o 3º período de Educação Física outro etc.), o eixo X é o dia semana (no caso testado, de segunda a sábado) e o eixo Y é o horário do dia (por exemplo, 19:00, 19:50, etc.). O valor da célula da matriz é um par Professor–Disciplina. A Figura 1 mostra um exemplo de uma seção da matriz vista de forma bidimensional.

Figura 1: Exemplo da representação da solução problema da alocação do horário

Z	X	Y	Valor
2ºCOMP	Segunda	19:00 – 19:50	Kennedy - Cálculo I
		19:50 – 20:40	Kennedy - Cálculo I
	...		
	Terça	19:00 – 19:50	Saulo - Sistema de informação
		19:50 – 20:40	Saulo - Sistema de informação
	...		
...			
4ºCOMP	Segunda	19:00 – 19:50	Cleverson - Probabilidade estatística
		19:50 – 20:40	Cleverson - Probabilidade estatística
	...		
	Terça	19:00 – 19:50	Daibert - Redes de comunicação de dados I
		19:50 – 20:40	Daibert - Redes de comunicação de dados I
	...		
...			

2.1. Modelagem do problema de geração de matrizes de horários

Nesta seção é apresentado o modelo desenvolvido para problema através de um conjunto de equações de Programação Matemática. Não há interesses em resolver o modelo proposto, uma vez que é não-linear. É apresentado aqui apenas para fins de descrever formalmente as restrições do problema e facilitar o seu entendimento. Ele é base para a implementação dos algoritmos.

As notações dos dados de entrada do problema de horário são: n que denota o número de alunos; G_z é o valor das horas semanais do aluno z para uma grade de disciplinas gerada a partir da solução atual; P denota o número de professores; T representa o número de *slots* de tempo; C diz respeito ao número de períodos do curso; D corresponde ao número de disciplinas; I representa o número de dias na semana; J representa o número de horários por dia; p denota um professor ($p \in \{0,1,2 \dots P\}$); d denota uma disciplina ($d \in \{0,1,2 \dots D\}$); i denota um dia da semana ($i \in \{0,1,2 \dots I\}$); j denota um horário de um dia ($j \in \{0,1,2 \dots . J\}$); c denota um período do curso ($c \in \{0,1,2 \dots C\}$); o_d indica se a disciplina d está sendo oferecida; $H_{d,c}$ indica se a disciplina d pertence ao período c; $A_{p,i,j}$ indica se o professor p está disponível no *slot* de tempo j no dia i; $L_{p,d}$ indica se o professor p lecionará a disciplina d; $h_{p,d}$ indica se o professor p está habilitado a lecionar a disciplina d; N_p indica o número de horas do contrato do professor p; K_d indica a carga horária da disciplina d; B_m indica o conjunto de tamanhos de blocos para os horários da disciplina m; b_v é um tamanho de bloco para uma disciplina. A variável de decisão desse modelo é $x_{p,d,i,j}$, que indica se o professor p irá lecionar a disciplina d no dia i no horário j.

O objetivo desse modelo é maximizar o número total de horas (T), que corresponde à soma das horas semanais tendo como base matrizes de disciplinas alocadas para cada aluno a partir de uma solução. Tal alocação é obtida através da solução de outro modelo, apresentado na seção 2.2.

Função objetivo: $max\ T = \sum_{z=1}^{n} G_z$
Sujeito a:

$\sum_d x_{p,d,i,j} \leq 1$	$\forall\, p = 0,1,2 \ldots P$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$	Um professor não estará alocado para mais de uma disciplina no mesmo *slot* de tempo
$\sum_d \sum_p x_{p,d,i,j} H_{d,c} \leq 1$	$\forall c = 0,1,2 \ldots C$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$	Uma sala não hospedará duas disciplinas no mesmo *slot* de tempo
$\sum_d x_{p,d,i,j} A_{p,d,c} \leq 1$	$\forall p = 0,1,2 \ldots P$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$	Impede que um professor seja atribuído a um *slot* de tempo em que ele não esteja disponível
$L_{p,d} h_{p,d} \leq 1$	$\forall p = 0,1,2 \ldots P$ $d = 0,1,2 \ldots D$	Um professor só será escolhido para lecionar uma disciplina em que ele for habilitado
$\sum_p L_{p,d} = o_d$	$\forall d = 0,1,2 \ldots D$	Uma disciplina só será lecionada por um professor se estiver sendo oferecida
$x_{p,d,i,j} L_{p,d} = 1$	$\forall p = 0,1,2 \ldots P$ $d = 0,1,2 \ldots D$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$	Um professor só será atribuído a um horário se lecionar a disciplina atribuída a ele
$\sum_d \sum_i \sum_j x_{p,d,i,j} \leq N_p$	$\forall p = 0,1,2 \ldots P$	O professor não excederá o número de horas de seu contrato
$\sum_p \sum_i \sum_j x_{p,d,i,j} = K_d$	$\forall d = 0,1,2 \ldots D$	A disciplina terá toda sua carga horária contemplada no horário
$x_{p,d,i,j} - x_{p,d,i,j+t} = 0$	$\forall p = 0,1,2 \ldots P$ $d = 0,1,2 \ldots D$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$ $b_v \in B_d \wedge b_v \in \{1, 2\}$ $t = 1,2 \ldots b_v - 1$	Se um horário for alocado para uma disciplina, então um número de horários igual ao número de blocos-1 também deverá ser alocado. O tamanho desse bloco deve ser 1 ou 2
$x_{p,d,i,j} \in \{0,1\}$	$\forall p = 0,1,2 \ldots P$ $d = 0,1,2 \ldots D$ $i = 0,1,2 \ldots I$ $j = 0,1,2 \ldots J$	Definem as variáveis de decisão como binárias

Existe uma restrição particular na instituição considerada: os coordenadores procuram alocar disciplinas sempre em pares, de forma que os alunos sempre assistam dois tempos seguidos de uma disciplina, à exceção daquelas que possuem carga horária ímpar.

2.2. Modelagem do problema para alocação de disciplinas aos alunos

A alocação de disciplinas e feita através de uma matriz de duas dimensões, em que cada linha representa um horário do dia (como, por exemplo, 19:00 a 19:50), e cada coluna representa um dia da semana. As células da matriz são apontamentos para os pares de disciplinas-professor de uma solução em questão. Esta matriz é denominada grade de disciplina.

Várias disciplinas podem coexistir no horário do curso para um mesmo espaço na matriz de horário, mas apenas uma delas pode ser escolhida para a alocação do aluno. Também devem ser obedecidas as seguintes restrições pedagógicas:

- Pré-requisitos, onde um aluno precisa ter sido aprovado em todas as disciplinas que são pré-requisito daquela que está sendo considerada (ou em suas equivalentes);
- Co-requisitos, onde o aluno precisa estar cursando ou ter cursado (aprovado ou não) as disciplinas co-requisitos daquela que está sendo considerada (ou equivalentes);

Além disso, disciplinas que o aluno já tenha sido aprovado também são consideradas infactíveis. Duas disciplinas são consideradas equivalentes se forem consideradas iguais do ponto de vista pedagógico.

As notações dos dados de entrada são: n denota o número de disciplinas da matriz curricular do curso; m denota o número de horários totais por semana do curso; i, j denotam as disciplinas ($i, j \in \{1,2, \dots, n\}$); k denota os horários ($k \in \{1,2, \dots, m\}$); G_i denota a carga horária da disciplina i; a_j identifica se o aluno foi aprovado na disciplina j, ou em alguma de suas equivalentes; u_j identifica se o aluno já cursou a disciplina j ou não, ou alguma de suas equivalentes; P_i corresponde ao número de pré-requisitos da disciplina i; $p_{i,j}$ identifica se j é pré-requisito de i; C_i corresponde ao número de co-requisitos da disciplina i; $c_{i,j}$ identifica se j é co-requisito de i; $h_{i,k}$ identifica se i possui uma aula no horário k; r_i identifica se a disciplina i é da preferência do aluno; $e_{i,j}$ identifica se as disciplinas i e j são equivalentes. O objetivo é encontrar a combinação de disciplinas de forma que todas as restrições sejam satisfeitas e maximizando o número de horas-aula semanais, calculado por:

$$\sum_{i=1}^{n} G_i y_i + 0.1 r_i y_i$$

Onde o peso 0.1 do segundo membro da função se deve à intenção de utilizar as preferências do aluno apenas como critério de desempate. O conceito de preferências varia de acordo com a instituição, podendo ser relacionado ao o horário ou ao dia em que as aulas estão agendadas, ou as disciplinas que farão parte da grade horária, entre outros. Neste trabalho, a pedido da própria instituição avaliada no caso de estudo, considera-se a preferência do aluno ser associado a disciplinas de sua turma e período iniciais.

Considerando que o número de disciplinas em uma alocação para o aluno no semestre nunca será igual a 10 (ou seja, $\sum_i^n r_i y_i < 1$ é sempre verdadeiro), a decisão sobre qual dentre duas alocações com número de horas igual será feita pela quantidade de disciplinas que atendem a preferência. Porém, uma alocação com maior número de horas sempre será melhor, mesmo que com menor número de preferências atendidas.

Após a geração do resultado, um truncamento é aplicado sobre o valor para desprezar as casas decimais. Fazendo com o que o valor retornado seja fiel ao número de horas computadas para a grade de disciplinas do aluno. Um modelo matemático de programação inteira binária (BIP) para o problema de alocações é proposto. A variável de decisão é a binária y_i, que indica se a disciplina i será cursada ou não. A seguir é apresentado o modelo para o problema.

Função objetivo: $maxT = \sum_{i=1}^{n} G_i y_i * 0.1 r_i y_i$
Sujeito a:

$a_i \leq \sum_{j=1}^{n} e_{i,j}$	$\forall i = 1,2, \dots, n$	Uma disciplina é considerada aprovada se, pelo menos, uma de suas equivalentes tiver sido aprovada
$c_i \leq \sum_{j=1}^{n} e_{i,j}$	$\forall i = 1,2, \dots, n$	Uma disciplina é considerada cursada se, pelo menos, uma de suas equivalentes tiver sido cursada
$P_i y_i \leq \sum_{j=1}^{n} a_j p_{i,j}$	$\forall i = 1,2, \dots, n$	Um aluno só cursará uma disciplina se seus pré-requisitos estiverem cumpridos
$y_i \leq 1 - a_i$	$\forall i = 1,2, \dots, n$	Uma disciplina em que o aluno já tenha sido aprovado seja escolhida para sua alocação
$\sum_{i=1}^{n} y_i h_{i,k} \leq 1$	$\forall k = 1,2, \dots, m$	Duas disciplinas que concorrem em algum horário não podem ser escolhidas

$C_i y_i \leq \sum_{j=1}^{n} c_{i,j}.(u_j + y_j)$	$\forall i = 1,2,\dots,n$	Um aluno só pode cursar uma disciplina se cumprir todos os seus co-requisitos, seja porque é uma disciplina que já cursou (mesmo sem ter sido aprovado) ou porque é uma disciplina que já foi escolhida para a alocação
$y_{i,} a_i, c_i \in \{0,1\}$	$\forall i = 1,2,\dots,n$	Define as variáveis de restrição e a de decisão como binárias

3. Resolução do problema

Para problemas de otimização combinatória da classe NP-Difícil não se conhecem algoritmos que os resolvam em tempo polinomial [Levitin 2003]. Como alternativas são utilizadas heurísticas, que se propõe a encontrar uma solução de alta qualidade em tempo polinomial [Talbi 2009]. Este trabalho propõe a utilização de um algoritmo genético, para resolução da primeira etapa do problema, aliado a meta-heurística GRASP para a resolução da segunda etapa.

3.1 Algoritmo Genético (AG-MaxHorario) para geração de matrizes de horários

O GA é um algoritmo evolutivo que busca melhorar uma população de soluções através da aplicação de operadores de seleção, combinação e mutação [Srinivas e Patnaik 1994]. O Algoritmo 1 mostra como o *template* do GA foi adaptado para resolução do problema em questão.

Algoritmo 1: AG-MaxHorario

```
Entrada: disciplinas da instituição d[];
         períodos p[];
         professores habilitados a ministrarem disciplinas profD[];
         tamanho da população PopTam;
         tamanho do torneio tor_tam;
         probabilidade de mutação probMutacao;
         número de tentativas de mutação ntMut;
         porcentagem de cruzamentos por iteração pCruz;
         operador de cruzamento op_cruz;
         número máximo de iterações sem melhoria ag_max_it;
  Início
1.      Pop = popInicial(d, p, profD, popTam);
2.      num_cruz = pCruz * PopTam;
3.      fo_best = Pop[0].fo;
4.      qtde_iteracoes_sem_melhora = 0;
5.      Repita
6.        pTor[] = torneio(Pop, tor_tam);
7.        filhos[] = cruzamento(pTor, num_cruz, op_cruz);
8.        Pop.add(filhos);
9.        Para individuo em Pop
10.         Se (rand(0..1) < probMutacao) então
11.           mut = mutacao(individuo, ntMut);
12.           Pop.add(mut);
13.         Fim Se
14.       Fim Para
15.       Pop = elitismo(Pop, PopTam);
16.       Se Pop[0].fo > fo_best então
17.         fo_best = Pop[0].fo;
18.         qtde_iteracoes_sem_melhora = 0;
19.       Senão
20.         qtde_iteracoes_sem_melhora++;
21.       Fim se
22.     Enquanto qtde_iteracoes_sem_melhora < ag_max_it
23.     H = Pop[0];
24. Fim

Saída: H /*Melhor matriz de horários encontrado*/
```

A população inicial é gerada aleatoriamente (linha 1), a partir de um processo que gera uma combinação de período, dia, horário, professor e disciplina e constrói uma matriz de horário. Isso ocorre até que a matriz de horários esteja completa, ou o algoritmo encontrar-se numa situação em que não consegue mais gerar uma solução factível (momento em que a atual é abandonada).

A população é uma coleção ordenada de indivíduos, comparados de forma decrescente a partir de sua função objetivo. Portanto, o melhor indivíduo da população é sempre o primeiro elemento da coleção (*Pop[0]*). A linha 2 define o número de cruzamentos como a porcentagem *pCruz* de *PopTam*. A linha 3 inicializa a variável *fo_best* com a FO do melhor indivíduo da população inicial. Essa variável será utilizada na condição de parada do algoritmo (linha 16).

O operador de seleção utilizado foi o torneio [Goldberg e Deb 1991], onde um número *tor_tam* de indivíduos é escolhido aleatoriamente da população e o melhor deles é selecionado. A linha 7 aplica o operador de cruzamento *op_cruz*. Três operadores foram testados: Order Crossover (OX) [Davis 1985], Cycle Crossover (CX) [Oliver et al., 1987] e Partial-Mapped Crossover (PMX) [Goldberg e Lingle]. Todos eles são efetuados considerando apenas uma camada, para evitar que um operador transporte uma disciplina para uma camada (período) a que ela não pertence.

A viabilidade da solução após a aplicação desses operadores é verificada, pois alguma das diversas restrições do problema podem ser desrespeitadas. Neste caso, as operações são reaplicadas sucessivamente até uma solução factível ser gerada ou todas as camadas terem sido testadas. Neste último caso, o operador não retorna uma nova solução.

O operador de mutação (linha 11) utilizado foi o de troca, onde uma camada é escolhida aleatoriamente e dois blocos, cada um contendo duas disciplinas, são trocados de lugar. Tal operação acontece aleatoriamente, a partir de um parâmetro (*probMutacao*) que determina as chances de uma determinada matriz de horários ser escolhida para ser modificada (linha 7). Como o processo pode gerar uma matriz de horários infactível, ele é executado até *ntMut* vezes, na tentativa de gerar uma matriz factível.

A instrução da linha 16 avalia se a solução corrente é melhor que a melhor conhecida (*fo_best*) e, se verdadeiro, faz a atualização (linha 17). O número de iterações sem melhora é controlado através da variável *qtde_iteracoes_sem_melhora*, sendo que ela é zerada quando uma solução melhor que a *fo_best* é encontrada (linha 15), caso contrário, é incrementada (linhas 19 e 20). O algoritmo para quando executar *ag_max_it* iterações e não for constatado nenhuma melhoria (linha 22). Em seguida, o melhor indivíduo da população é retornado (linha 23).

3.2. Algoritmo GRASP (GRASP-MaxAlocação) para alocação de disciplinas aos alunos

O GRASP é uma meta-heurística que, basicamente, executa dois passos de forma iterativa: construção de uma nova solução e busca local. O Algoritmo 2 (denominado GRASP-MaxAlocacao) mostra como esta meta-heurística foi implementada.

Algoritmo 2: Algoritmo GRASP-MaxAlocação

```
Entrada: matriz de horário H[][][];
         aluno A;
         alfa α;
         número máximo de iterações sem melhoria grasp_max_it;
         número de vizinhos gerados pela busca local num_vizinhos;
   Início
1.    sb = Ø;
2.    qtde_iteracoes_sem_melhora = 0;
3.    Repita
4.      s  = Construção(A, H);
5.      s' = BuscaLocal(s, A, α, num_vizinhos);
6.      Se (s'.fo > sb.fo) Então
7.        sb = s;
8.        qtde_iteracoes_sem_melhora = 0;
9.      Senão
10.     qtde_iteracoes_sem_melhora = qtde_iteracoes_sem_melhora + 1;
11.     Fim se
12.   Enquanto qtde_iteracoes_sem_melhora < grasp_max_it
   Fim
   Saída: sb /* Melhor matriz de disciplinas encontrada */
```

O algoritmo recebe como entrada uma matriz *H* de horários completa, um aluno *A* para quem irá gerar uma alocação, um α que regula seu comportamento, o número máximo de iterações sem melhoria *grasp_max_it* e o número de vizinhos (*num_vizinhos*) que a busca local irá gerar.

O método de construção é realizado na linha 4 e gera uma grade de horários *s* para o aluno *A* com base na matriz de horários *H*. Na linha 5 é aplicada busca local *s*. Estes procedimentos se repetem até que não haja melhoria na grade de disciplinas s_b (linhas 6 a 12).

O Algoritmo 3 apresenta como é realizado o procedimento de construção do GRASP.

Algoritmo 3: Fase de construção do GRASP

```
Entrada: matriz de horário H[][][];
         aluno A;
  Início
1.     d[] = A.disciplinas_restantes;
2.     s = Ø; /* matriz de disciplinas do aluno */
3.     Repita
4.       ds = remove_aleatorio(d);
5.       Se s.pode_adicionar(ds) Então
6.         s.adicionar(ds);
7.       Fim Se
8.     Até s.completa ou d = Ø
   Fim
Saída: s /* Matriz de disciplinas montada */
```

A linha 1 copia a lista de disciplinas restantes do aluno para a variável *d*. Das linhas 3 a 8 tem-se um laço de repetição: obtém-se uma disciplina da lista de forma aleatória (linha 4), e tenta-se inseri-la na matriz de disciplinas do aluno (linhas 5 e 6). Nesse momento são verificadas as restrições (linha 5). Se nenhuma for quebrada, a inserção é realizada (linha 6). O laço termina quando não houverem mais candidatas ou a alocação esteja completa (linha 8).

A seguir, no Algoritmo 4, é apresentado o pseudocódigo do procedimento de busca local.

Algoritmo 4: Fase de busca local do GRASP

```
Entrada: matriz de disciplinas montada s;
         Aluno A;
         alfa α;
         número de vizinhos desejados num_vizinhos;
  Início
1.     sbest = s;
2.     Para i de 0..num_vizinhos
3.       s' = remove_disciplina_aleatoria(s);
4.       Repita
5.         d'[] = lista_candidata(A.disciplinas_restantes, α);
6.         ds = escolhe_aleatorio(d');
7.         Se s'.pode_adicionar(ds) Então
8.           s'.adicionar(ds);
9.         Fim Se
10.      Até s'.completo ou d' = Ø
11.      Se s'.fo > sbest.fo Então
12.        sbest = s';
13.      Fim Se
14.    Fim Para
   Fim
Saída: sbest /* Melhor matriz de disciplinas vizinha encontrada */
```

O procedimento de busca local realiza a remoção de uma das disciplinas atuais da alocação e tenta a inserção de uma outra disciplina, aleatoriamente obtida da lista de candidatas *d'*. Tal busca local é executada *num_vizinhos* de vezes, cada uma gerando um vizinho e ao final é tomado o melhor dentre eles.

O procedimento recebe uma matriz de disciplinas pronta *s*, um aluno *A* e os parâmetros *α* e o número de vizinhos *num_vizinhos*. Inicializa-se a melhor alocação (s_{best}) como sendo a de entrada (linha 1). As iterações são executadas *num_vizinhos* vezes (linha 2). A cada iteração uma disciplina é removida aleatoriamente da matriz de horários de entrada, gerando uma matriz incompleta *s'* (linha 3). As linhas 4 a 10 compõem um laço de repetição, que executa os seguintes passos: é gerada uma lista de disciplinas candidatas *d'* a partir da lista de disciplinas restantes do aluno e do parâmetro *α* (linha 5). Tal *d'* é formada pelas *α*% melhores disciplinas da lista de restantes do aluno, comparadas de forma decrescente em relação a suas cargas horárias. Uma disciplina (d_s) é então escolhida aleatoriamente de *d'* (linha 6). As linhas 7 a 9 tentam adicionar d_s a *s'*. Se nenhuma restrição for quebrada, a disciplina é inserida com sucesso. Tal laço é executado até *s'* estar completa (é impossível adicionar mais disciplinas à matriz de disciplinas do aluno) ou até que *d'* esteja vazia, ou seja, todas as disciplinas já tenham sido testadas (linha 10).

Compara-se então o vizinho que acabou de ser gerado (*s'*) com o melhor até agora (s_{best}). Se ele for melhor, passa a ocupar seu lugar (linhas 11 a 13). Ao final do algoritmo, a melhor matriz de disciplinas obtida é retornada.

4. Experimento computacional

Pelo fato do algoritmo possuir muito parâmetros, foi executado um experimento computacional para descobrir a melhor configuração de parâmetros para o problema em questão. Os algoritmos foram implementados em C++11 e compilados com Visual C++ 12, configurado para aplicar todas as otimizações. As execuções foram distribuídas entre 39 computadores iguais, possuindo o processador AMD A8-5600K com *clock* de 3.6Ghz, 8.0 Gb de memória RAM e sistema operacional Windows 7 64-bits.

Os dados de entrada[1] foram cedidos pela FAGOC e obtidos diretamente do sistema de gerenciamento acadêmico da instituição, através de uma consulta SQL em um *snapshot* do banco de dados referente ao segundo semestre de 2015. No experimento foram utilizados todos os dados recebidos, que consiste em 1392 alunos, 11 cursos, 2248 disciplinas, 68 professores e 24 horários e 33 períodos.

Os valores testados para cada parâmetro do algoritmo foram: número de indivíduos (*PopTam*), com possíveis valores 20 e 40; porcentagem de cruzamentos por iteração (*pCruz*), com valores 20% e 30%; número de tentativas de mutação (*ntMut*), com 2 e 4; número de indivíduos por torneio (*tor_tam*), com 2 e 4; número de iterações sem melhoria do GRASP (*grasp_max_it*), com 15 e 25; número de vizinhos do GRASP com 2 e 4 (*num_vizinhos*); o *α* do GRASP, com 20%, 40% e 60%; o número de iterações sem melhoria do AG (*ag_max_it*), com valores possíveis 20 e 40; operadores de cruzamento (*op_cruz*) com CX, OX e PMX; taxa de mutação (*probMutacao*) com 15% e 30%. Ao todo, devido ao produto cartesiano dos valores dos parâmetros, obteve-se 2304 combinações. Cada combinação foi executada 10 vezes, totalizando 23040 rodadas.

Para avaliação dos resultados foi utilizada a métrica de qualidade Desvio Percentual Relativo, denotado por RPD e obtido através da fórmula:

$$RPD = \frac{f_{method} - f_{best}}{f_{best}} \; x \; 100$$

onde f_{method} representa a função objetivo daquela solução e f_{best} representa a melhor função objetivo observada no experimento.

Foi observado que 1536 combinações alcançaram a mesma função objetivo, a saber 22.502 horas, que é a maior observada no experimento, o que as rendeu um RPD de 0. Esse conjunto é composto por todas as combinações que utilizam os operadores de cruzamento CX e OX. Sendo assim, concluiu-se que o operador PMX é o menos eficiente para este problema. Em termos de tempo de execução, as configurações que utilizam o operador de cruzamento CX foram, em média, mais rápidas. Obtendo o tempo médio de 94,8 segundos. Dessa forma, concluímos que a melhor configuração de parâmetros para aplicação neste problema (ou seja, aquela que gera os melhores resultados médios com menor tempo de execução) é que utiliza os seguintes valores: número de

[1] Dados de entrada e resultado da execução do algoritmo calibrado disponíveis: https://goo.gl/og1kL3

indivíduos (*popTam*) = 20, número de tentativas de mutação (*ntMut*) = 4, número de indivíduos por torneio (*tor_tam*) = 4, número de iterações sem melhoria do GRASP (*grasp_max_it*) = 15, número de vizinhos do GRASP (*num_vizinhos*) = 2, α do GRASP = 20%, número de iterações sem melhoria do AG (*ag_max_it*) = 20, operador de cruzamento (*op_cruz*) = CX, taxa de mutação (*probMutacao*) = 15%. Para este conjunto de parâmetros e valores o resultado obtido (em média) foi de 22.502 horas (RPD médio 0) e tempo 22 segundos. O operador OX também conseguiu um RPD médio de 0, mas foi mais lento que o CX.

5. Conclusão

Neste trabalho foi abordado o problema de alocação de horários, disciplinas, professores e turmas numa instituição de Ensino Superior privada. A proposta do estudo era encontrar um algoritmo que resolvesse o problema para o contexto da instituição FAGOC, que envolve tanto a montagem de uma matriz de horário para os cursos, quanto a alocação de disciplinas a serem estudadas no semestre para cada aluno. O objetivo em questão foi encontrar uma matriz de horário que possibilitasse a maximização do número total de horas estudadas na instituição pelos alunos. Para isso foi desenvolvido um algoritmo que combina as meta-heurísticas Algoritmo Genético e GRASP, com utilização de técnicas conhecidas da literatura. Foram realizados experimentos computacionais para extrair o melhor desempenho do algoritmo. O resultado obtido pelo algoritmo proposto foi superior ao atual processo de trabalho da FAGOC. Enquanto o primeiro obtém um resultado de 22502 horas estudadas, o segundo é de 22377, o que corresponde a uma melhoria de 0.56%. Quanto ao tempo gasto, o algoritmo leva apenas 22 segundos para resolver todo o problema, enquanto o processo manual é necessário 24 dias úteis (em média). É de interesse da instituição que o algoritmo seja incorporado ao seu sistema computacional de gestão acadêmica para automatização do processo de montagem de horários e grade de disciplinas dos alunos.

Referências

Abdullah, S., e Turabieh, H. (2012). On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems. *information sciences*, 191, p. 146-168.

Aubin, J., e Ferland, J. A. (1989). A large scale timetabling problem. *Computers & Operations Research*, 16(1), p. 67-77.

Carrasco, M. P., e Pato, M. V. (2000). A multiobjective genetic algorithm for the class/teacher timetabling problem. *In Practice and Theory of Automated Timetabling III*, p. 3-17. Editora Springer Berlin Heidelberg.

Casey, S., & Thompson, J. (2002). GRASPing the examination scheduling problem. In *Practice and Theory of Automated Timetabling IV*, p. 232-244. Editora Springer Berlin Heidelberg.

Chahal, N. e De Werra, D. (1989). An interactive system for constructing timetables on a PC. *European Journal of Operational Research*, 40(1):32-37.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In *Proceedings of an international conference on genetic algorithms and their applications (Vol. 140)*. Carnegie-Mellon University Pittsburgh, PA.

Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160(1):106-120.

De Werra, D. An Introduction to Timetabling. (1985). *European Journal of Operational Research*, 19:151-162.

Di Gaspero, L., e Schaerf, A. (2002). Multi-neighbourhood local search with application to course timetabling. In *Practice and theory of automated timetabling IV*, p. 262-275. Editora Springer Berlin Heidelberg.

Eiselt, H. A. e Laporte, G. (1987). Combinatorial optimization problems with soft and hard requirements. *Journal of the Operational Research Society*, 785-795.

FAGOC. (2016). Web Page. http://fagoc.br/institucional/apresentacao. Acessado: 2016-05-11.

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

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

Fong, C. W., Asmuni, H., McCollum, B., McMullan, P., e Omatu, S. (2014). A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems. *Information Sciences*, 283, 1-21.

Goldberg, D. E. e Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, p. 154-159. Lawrence Erlbaum Associates, Publishers.

Goldberg, D. E. e Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69-93.

Gotlieb, C. C. (1963). The construction of class-teacher timetables. In *C. M. Popplewell, IFIP congress 62*, p. 73–77. North-Holland.

Hertz, A. (1991). Tabu search for large scale timetabling problems. *European journal of operational research*, 54(1):39-47.

Hertz, A. (1992). Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics*, 35(3):255-270.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Editora U Michigan Press.

Larranaga, P., Kuijpers, C. M., Murga, R. H. e Yurramendi, Y. (1996). Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 26(4):487-493.

Oliver, I. M., Smith, D., e Holland, J. R. (1987). Study of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987.

Paechter, B., Cumming, A., Luchian, H. e Petriuc, M. (1994). Two solutions to the general timetable problem using evolutionary methods. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence, Proceedings of the First IEEE Conference*, p. 300-305. IEEE.

Qu, R., e Burke, E. K. (2009). Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems. *Journal of the Operational Research Society*, 60(9), p. 1273-1285.

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

Schmidt, G e Strohlein, T. (1979). Timetable construction – an annotated bibliography. *The Computer Journal*, 23(43):07-316.

Srinivas, M., e Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer*, 27(6), p.17-26.

Tripathy, Arabinda. (1984). School timetabling—a case in large binary integer linear programming. *Management science*, 30(12):1473-1489.

Welsh, J. A. e Powell, M. B. (1967). An upper bound to the chromatic number of a graph and its applications to timetabling problems. *The Computer Journal*, 10:85-86.