Um Algoritmo GRASP com Cadeia de Kempe Aplicado ao Problema de Tabela-horário para Universidades

Erika de Almeida Segatto
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras - 29075-910 - Vitória - ES
erika.ccomp@gmail.com

Maria Claudia Silva Boeres, Maria Cristina Rangel
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras - 29075-910 - Vitória - ES
{boeres,crangel}@inf.ufes.br

Edmar Hell Kampke
Universidade Federal do Espírito Santo
Alto Universitário, s/n, Guararema, Cx. postal 16 - 29500-000 - Alegre - ES
edmar.kampke@ufes.br

RESUMO

O problema de Tabela-horário educacional consiste em alocar um conjunto de aulas em salas disponíveis e períodos de tempo pré-determinados, considerando alunos e professores e satisfazendo algumas restrições. Existem na literatura diversos modelos matemáticos para esse problema. Adotamos o modelo voltado para universidades proposto no segundo campeonato internacional de tabela-horário (ITC-2007), que é baseado em currículos de disciplinas. Para a solução desse problema, utilizamos a meta-heurística GRASP com *Path-Relinking*, já investigada em trabalhos anteriores. Naquela versão do GRASP, os dois tipos de movimento empregados na busca local são conhecidos na literatura como *MOVE* e *SWAP*. Neste trabalho, agregamos à busca local do GRASP mais um tipo de movimento: Cadeia de Kempe. Testes computacionais com instâncias disponibilizadas pelo ITC-2007 foram realizados para o GRASP com os três tipos de movimentos e os resultados foram comparados com as melhores respostas obtidas no ITC-2007.

PALAVRAS CHAVE. Tabela-horário para Universidades, GRASP, Cadeia de Kempe

Área Principal: Otimização Combinatória, Metaheurísticas

ABSTRACT

Educational timetabling problem is about allocate a set of classes available rooms and predetermined periods considering students and teachers and satisfying some restrictions. We adopt the model proposed on the competition track 3 of the International Timetabling Competition 2007 (ITC-2007). For the solution of this problem, we use the GRASP meta-heuristic with Path-Relinking, already investigated in previous work by the authors. In that version of GRASP, the two types of movement used in the local search are known in the literature as MOVE and SWAP. In this work, we added the local search GRASP another type of movement: Kempe chain. Computational tests with instances provided by the ITC-2007 were performed and results were compared with the best responses from the ITC-2007.

KEYWORDS. University timetabling, GRASP, Kempe Chain

Main Area: Combinatorial Optimization, Metaheuristics

1. Introdução

O problema de tabela-horário é um problema de alocação de recursos, no qual o objetivo é alocar entidades (como tarefas, eventos ou pessoas) a um recurso limitado no tempo, e possui aplicação em diversos domínios, como por exemplo, na elaboração de escala para funcionários, agendamento de partidas para campeonatos esportivos, ou na área educacional [Schaerf(1999)]. Tabela-horário educacional engloba escolas, universidades e exames. Nesse trabalho abordamos o problema de tabela-horário para universidades.

Na tabela-horário para universidades, o problema consiste em alocar um conjunto de aulas em um número pré-determinado de horários, satisfazendo restrições de professores, alunos e espaço físico. A formulação do problema pode variar de uma universidade para outra, pois cada uma possui seus próprios requisitos. Por exemplo, em uma universidade pode ser desejado a compactação do horário do professor, enquanto que em outra, o professor trabalha em horário integral, não se fazendo necessário o cumprimento desse requisito. Apesar de existirem diferentes formulações, as restrições são comumente divididas em fracas e fortes [Burke and Petrovic(2002)]. As restrições fortes são aquelas que devem obrigatoriamente ser satisfeitas para que a tabela-horário seja válida. Por exemplo, deve ser respeitado o fato de que um professor não pode dar mais de uma aula em um mesmo horário. As restrições fracas são aquelas que, idealmente, deseja-se satisfazer, mas não são necessárias para a viabilidade da solução. Por exemplo, que uma mesma turma não tenha períodos vagos entre aulas.

Com o intuito de gerar pesquisas com diferentes abordagens e preencher o espaço entre a pesquisa e prática neste tema, o PATAT - *The International Series of Conferences on the Practice and Theory of Automated Timetabling* promoveu três competições de tabela-horário educacional, chamados de ITC - *International Timetabling Competition*. A primeira competição do ITC foi realizada em 2003 e tinha como foco a tabela-horário para universidades. A segunda competição, em 2007, abordou três formulações: tabela-horário para universidades baseada em exames, informações de matrícula dos alunos e currículo dos cursos [McCollum et al.(2010)]. A terceira, e mais recente competição, foi a ITC-2011, cujo foco foi o problema de tabela-horário para escola de ensino médio (*high school*). Neste trabalho utilizamos a ITC-2007 na sua terceira formulação como referência, pois utiliza a formulação baseada no currículo dos cursos, semelhante ao modelo das universidades brasileiras, em especial, a Universidade Federal do Espírito Santo - UFES.

Os primeiros algoritmos para o problema foram baseados em métodos de resolução manual, isto é, a tabela-horário é construída aula por aula, da mais conflitante para a menos conflitante, até que todas tenham sido inseridas [Schaerf(1999)]. Contudo, na literatura existem diversas abordagens mais inteligentes para solucionar este problema. Em [Carter et al.(1996)], os autores propõem heurísticas baseadas em coloração de grafos para tabela-horário de exames, em [Daskalaki et al.(2004)], recorrem à programação inteira para modelar tabela-horário de universidades e em [Mulvey(1982)], os autores modelam o problema de espaço físico como fluxo de redes em grafos.

As meta-heurísticas também são utilizadas para o problema de tabela-horário, entre as quais destacamos os Algoritmos Genéticos [Burke et al.(1994)] e VNS (*Variable Neighborhood Search*) aplicados ao problema de tabela-horário de exames, Busca Tabu [Hertz(1991)] proposto para resolver problemas de tabela-horário de grande porte, *Great Deluge* [Shaker and Abdullah(2009), Mcmullan(2007)] para o problema de tabela-horário universitário baseado em cursos conforme a definição do ITC-2007.

Ainda no contexto das meta-heurísticas aplicadas a este problema, recentemente foi realizado um trabalho na Universidade Federal do Espírito Santo utilizando o GRASP [Rocha(2013)]. Esta meta-heurística é um procedimento com iterações independentes, onde cada iteração constrói uma solução inicial que, em seguida, é submetida a uma busca local [Feo and Resende(1995)]. A resposta do algoritmo é a melhor solução obtida dentre todas as iterações. A versão apresentada

em [Rocha(2013)] utiliza *Hill Climbing* e *Simulated Annealing* como estratégias de busca local, com dois movimentos conhecidos na literatura: *MOVE* - deslocamento de apenas uma aula para algum horário vago, e *SWAP* - troca uma aula por por outra, além do *Path Relinking* para refinamento das soluções locais. Neste trabalho propomos uma nova versão do método apresentado em [Rocha(2013)], fixando como busca local a meta-heurística *Simulated Annealing* e agregamos um outro movimento, denominado **Cadeia de Kempe**. Tal movimento foi estudado cuidadosamente e adaptado para ser utilizado como uma alternativa de diversificação para o GRASP aplicado ao problema de tabela-horário. A Cadeia de Kempe envolve a troca de subconjuntos de aulas, ao contrário do *MOVE* e *SWAP*. Essa estrutura de movimentos é um conceito que tem origem na Teoria dos Grafos, a partir da tentativa frustrada do matemático Kempe, em 1879, de provar que qualquer grafo planar pode ser colorido com quatro cores.

No problema de tabela-horário, a Cadeia de Kempe pode ser interpretada como um *SWAP* de blocos de aulas, de acordo com um grafo de conflitos conexo construído respeitando as restrições existentes. No trabalho de [Thompson and Dowsland(1998)], a Cadeia de Kempe é utilizada de forma a permitir movimentos que ajudam a escapar de ótimos locais, mostrando-se uma boa técnica para diversificação.

Neste trabalho desenvolvemos um algoritmo GRASP híbrido com *Path Relinking* tendo como busca local a meta-heurÃstica SA com três movimentos, a citar, *MOVE*, *SWAP* e Cadeia de Kempe. Os experimentos computacionais são efetuados para o mesmo conjunto de 21 instâncias adotado pelo ITC-2007, considerando a sua terceira formulação. Tais instâncias são advindas de exemplos reais da Universidade de Udine, Itália. Os resultados são comparados com os obtidos pelos algoritmos submetidos na competição e se mostraram competitivos.

Na seção 2 é apresentada a formulação do problema adotada neste trabalho. Na seção 3 o algoritmo desenvolvido para o problema de tabela-horário de universidades é descrito, com destaque para a Cadeia de Kempe em problemas de tabela-horário na subseção 3.4. Na seção 4 são apresentados os resultados computacionais obtidos e confrontados com as melhores respostas para o problema da competição ITC-2007. Por fim, a seção 5 apresenta as conclusões obtidas com o trabalho e lista algumas melhorias futuras para o algoritmo GRASP proposto neste trabalho.

2. Tabela-horário para Universidades

O problema de tabela-horário para universidades consiste de um horário semanal de aulas para várias disciplinas, considerando um número de salas e de períodos de aulas, no qual os conflitos entre disciplinas são determinados de acordo com o currículo, e não pelas matrículas. Um período é um horário em um dia de aula. A solução desse problema é chamada de **tabela-horário**. Cada universidade possui necessidades específicas, e por isso existem várias formulações diferentes para o problema. Utilizamos a terceira formulação do ITC-2007 (baseada no currículo dos cursos) [Di Gaspero et al.(2007)], por se adequarem ao padrão de universidades brasileiras e por possibilitar comparações com resultados obtidos naquela competição. Essa formulação possui restrições divididas em fortes e fracas.

As restrições fortes (RFt) são aquelas que devem, obrigatoriamente, ser satisfeitas. A tabela-horário que não satisfaz alguma restrição forte é inviável. O ITC-2007 considera as seguintes restrições fortes: (i) todas as aulas das disciplinas devem ser alocadas e em períodos diferentes. Uma violação ocorre se uma aula não é alocada ($aula$); (ii) aulas de disciplinas do mesmo currículo ou lecionadas pelo mesmo professor devem ser alocadas em períodos diferentes ($conflito$); (iii) duas aulas não podem ocupar uma sala no mesmo horário ($sala$) e (iv) uma aula não pode ser alocada num horário em que a disciplina é indisponível ($disponibilidade$).

As restrições fracas (RFc) são aquelas que devem, idealmente, ser satisfeitas. Quanto menos violações às restrições fracas existirem, melhor é a tabela-horário. São elas: (i) Dias Mínimos de Trabalho: as aulas de cada disciplina devem ser espalhadas por uma quantidade mínima de dias. Cada dia abaixo do mínimo é contado como uma violação ($dmin$); (ii) Aulas Isoladas:

aulas do mesmo currículo, não necessariamente as aulas de uma mesma disciplina, devem ser alocadas em períodos adjacentes. Cada aula isolada é contada como uma violação ($isolada$); (iii) Capacidade da Sala: o número de alunos da disciplina deve ser menor ou igual ao número de assentos da sala em que a aula for alocada. Cada aluno excedente contabiliza uma violação ($capacidade$) e (iv) Estabilidade de Sala: todas as aulas de uma disciplina devem ser alocadas na mesma sala. Cada sala distinta é contada como uma violação ($estabilidade$).

Essa formulação considera também um valor calculado para uma tabela-horário que reflete sua qualidade. A função que calcula esse valor é chamada de função objetivo f_o. Cada restrição infringida aumenta a pontuação de f_o de acordo com o seu peso. A melhor tabela-horário para um problema é a tabela-horário que minimiza o valor desta função dada pela fórmula

$$f_o = \text{Violações}_{RFt} + \text{Violações}_{RFc}$$

em que Violações$_{RFt}$ = $|aula|+|conflito|+|sala|+|disponibilidade|$ e Violações$_{RFc}$ = $5|dmin|+2|isolada| + |capacidade| + |estabilidade|$. O símbolo $|\cdot|$ representa o número de violações de cada restrição. Na contagem total das violações fracas são considerados pesos diferentes para cada tipo de violação. A restrição $dmin$ possui peso 5 (cinco), $isolada$, peso 2 (dois) e as demais, peso 1 (um).

3. GRASPK - um algoritmo hÃbrido GRASP e SA com Path Relinking e Cadeia de Kempe

Nesta seção é apresentado o algoritmo híbrido GRASP com Path Relinking utilizando SA como estratégia de busca local. Os movimentos adotados para geração dos vizinhos são MOVE, SWAP e Cadeia de Kempe. Para melhor entendimento, as etapas foram descritas em subseções distintas.

A meta-heurística *Greedy Randomized Adaptive Search Procedures* (GRASP) foi proposta por [Feo and Resende(1989), Feo and Resende(1995)] e se baseia na estratégia *multi-start*, ou seja, a cada iteração são criadas novas soluções iniciais. Assim é possível explorar o espaço de busca de forma mais ampla. Em cada iteração da meta-heurística, após a criação da solução inicial, é aplicada uma busca local na tentativa de melhorar a qualidade da solução [Nascimento et al.(2010)]. A resposta do algoritmo é a melhor solução alcançada dentre todas as iterações.

O algoritmo GRASP com *path-relinking* é apresentado no pseudo-código do Algoritmo 1.

Algoritmo 1: Algoritmo Híbrido GRASPK com Path-Relinking para o ITC-2007

Entrada: MaxIter, MaxElite
Saída: Melhor Solução S^*

```
1  f* ← ∞ ;
2  Elite ← ∅ ;
3  para i ← 1 até MaxIter faça
4  |  S ← GeraSolucaoInicial() ;
5  |  S ← SA(S) // Com Move, Swap e Kempe
6  |  se i ≥ 2 então
7  |  |  Selecione aleatoriamente S_elite ∈ Elite ;
8  |  |  S ← PathRelinking(S, S_elite) ;
9  |  fim se
10 |  se f(S) < f* então
11 |  |  S* ← S ;
12 |  |  f* ← f(S) ;
13 |  fim se
14 |  AtualizaElite(S, Elite) ;
15 fim para
```

3.1. Construção da solução inicial

A construção de uma solução inicial no GRASP é realizada de forma gulosa (para produzir soluções de melhor qualidade) e também aleatória (para produzir soluções diferentes a cada iteração). O objetivo da construção é produzir uma tabela-horário válida tentando minimizar as restrições fracas. O GRASP não requer que a solução inicial seja válida, mas esse recurso foi adotado para que as fases seguintes do algoritmo se concentrem apenas na minimização das violações das restrições fracas.

Neste trabalho, a construção da solução inicial começa com uma tabela-horário vazia. Em seguida as aulas são alocadas uma a uma até que ao final, estejam todas alocadas. Em outras palavras, a cada iteração, a aula mais difícil (a que possui menos horários viáveis) é escolhida para ser alocada. O nível de dificuldade de uma aula é medido pelo número de conflitos na sua alocação. Esse valor é calculado contando-se quantos horários ainda disponíveis na tabela-horário atual são viáveis para alocar a aula da disciplina.

A partir disso, existem diferentes combinações de horários e salas para a alocação de uma aula. Os custos de todas essas combinações são calculados levando-se em conta as penalizações das restrições fracas. Dentre as combinações viáveis, ou seja, aquelas em que é possível realizar a alocação da aula em um horário disponível, é escolhido aleatoriamente um horário e sala que esteja na Lista Restrita de Candidatos (LRC), definida pelo intervalo: $[c^{min}; c^{min} + \alpha(c^{max} - c^{min})]$, sendo c^{min} o menor custo dentre as combinações de horários e salas disponíveis, c^{max} o maior custo dentre as combinações de horários e salas disponíveis, e $\alpha \in [0; 1]$. Caso $\alpha = 1$, o conjunto LRC possuirá todas as combinações de horários e salas disponíveis, fazendo com que o algoritmo de construção se torne completamente aleatório. Caso $\alpha = 0$, o conjunto LRC possuirá apenas a combinação de horário e sala disponível de menor custo, fazendo com que o algoritmo de construção seja completamente guloso.

É possível que em alguns casos, ao escolher uma aula para alocar, não haja um horário que mantenha a viabilidade da solução. Por isso foi aplicado um procedimento denominado **explosão**, que permite que uma aula alocada anteriormente seja retirada da tabela-horário para abrir espaço para a aula que não está sendo possível alocar. A aula retirada volta para o conjunto de aulas não alocadas. A aula a ser retirada da tabela-horário é escolhida aleatoriamente, de forma que, ao repetir o procedimento de explosão, aulas diferentes sejam escolhidas para serem removidas.

3.2. Busca local

Dois algoritmos de busca local foram adotados por no GRASP: *Hill Climbing* (HC) e *Simulated Annealing* (SA). O GRASP com SA gerou resultados melhores do que aqueles obtidos usando HC como busca local. Por esse motivo, neste trabalho será utilizada apenas a meta-heurística SA como busca local do GRASP, sendo detalhada na seção 3.2.1.

Os algoritmos de busca local utilizam movimentos de geração de vizinhos a partir de uma solução corrente para a geração de novas soluções. Neste trabalho foram implementados dois movimentos simples denominados de *MOVE* e *SWAP*. O movimento *MOVE* significa mover uma aula para uma posição desocupada na tabela-horário e o *SWAP* é a troca das posições de duas aulas na tabela-horário. Além disso, neste trabalho também foi implementado um terceiro tipo de movimento denominado de Cadeia de Kempe, que será detalhado na seção 3.4.

Na busca SA a escolha do tipo de movimento é feita de forma aleatória, sendo que cada movimento efetuado possui 50% de chance de ser do tipo *MOVE* e 50% de chance de ser do tipo *SWAP*.

3.2.1. Simulated Annealing

O algoritmo *Simulated Annealing* [Kirkpatrick(1984)] foi concebido numa analogia ao processo de resfriamento do aço. Motivados pela possibilidade de acrescentar inteligência ao algoritmo GRASP, o SA foi utilizado neste trabalho como busca local com o objetivo de tentar escapar de ótimos locais aceitando soluções piores.

O algoritmo *Simulated Annealing* (SA) possui três parâmetros principais: temperatura inicial, final e taxa de resfriamento. A partir de uma temperatura inicial mais alta, são gerados N_v vizinhos a cada iteração. A cada vizinho gerado, é calculado o valor da função objetivo. Caso o vizinho gerado seja melhor que a solução atual, esta é atualizada. Caso contrário, ele é aceito com uma probabilidade igual a $P = e^{-\Delta f/T}$, em que Δf é a diferença de valor da função objetivo do vizinho e da solução atual e T é a temperatura atual. Quanto maior for o Δf, e menor a temperatura T, menor é a chance de aceitar um vizinho pior do que a melhor solução atual. Após a geração dos vizinhos, recalcula-se a temperatura atual aplicando a taxa de resfriamento(β) através da fórmula $T = T * \beta$. O algoritmo termina quando a temperatura atual é menor ou igual à temperatura final. Na prática, o algoritmo aceita mais soluções piores nas primeiras iterações, obtendo grande diversificação, e tem menos chance de aceitar soluções piores à medida que a temperatura diminui, realizando intensificação da busca.

3.3. Path-Relinking

O *path-relinking* foi proposto originalmente por [Glover(1998)] no contexto dos algoritmos Busca Tabu e *Scatter Search*. Em [Laguna and Marti(1999)], o *path-relinking* foi incorporado ao GRASP. A hibridização do *path-relinking* com o GRASP consiste em armazenar um conjunto de soluções de alta qualidade, denominadas de soluções elites, encontradas durante a busca. Novas soluções são desenvolvidas explorando a trajetória que conecta as soluções elites e as soluções alcançadas pelas iterações do GRASP.

A cada iteração da meta-heurística, o *path-relinking* é executado tendo como entrada, a solução retornada pela busca local, denominada de solução alvo, e uma solução aleatória dentre as soluções elites (conjunto Elite), denominada de solução inicial. O objetivo é gerar modificações, ou variações, na solução inicial até obter a solução alvo, para assim explorar soluções vizinhas de alta qualidade.

3.4. Cadeia de Kempe

Além dos movimentos *MOVE* e *SWAP* já implementados como movimentos nos algoritmos de busca local, agregamos neste trabalho, mais um movimento para a busca local, denominado Cadeia de Kempe. No contexto do problema de tabela-horário, a Cadeia de Kempe é interpretada da seguinte forma: a partir de uma tabela-horário válida, dois horários são escolhidos. Considere no exemplo da figura 1, os horários t_1 e t_2. A aula a_{11} é a aula alocada no horário e dia do horário t_1 na sala 1; a aula a_{12}, alocada no mesmo dia e horário na sala 2, e assim por diante. Em seguida é preciso criar um grafo de conflitos, onde cada aula é um vértice. Se duas aulas conflitam por causa de uma restrição forte, acrescenta-se uma aresta entre elas. Cada grafo conexo formado é uma Cadeia de Kempe.

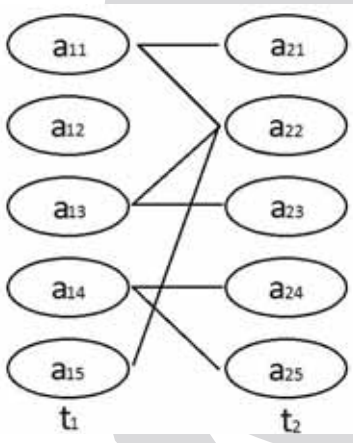

Figure 1: Exemplo de Cadeias de Kempe nos horários t_1 e t_2

A seguir são escolhidas uma ou mais Cadeias de Kempe para que possa ser realizada a troca das aulas pertencentes a essas cadeias. No exemplo da figura 1 existem três Cadeias de Kempe: a primeira é a cadeia que contém as aulas a_{11}, a_{21}, a_{22}, a_{13}, a_{23} e a_{15}, a segunda é a cadeia formada pelas aulas a_{14}, a_{24} e a_{25}, e a terceira é a cadeia formada somente pela aula a_{12}. A tabela-horário obtida após a realização das trocas continuará, obrigatoriamente, sendo válida. Essa garantia ocorre pois todas as aulas, dos horários escolhidos, que conflitam através de uma restrição forte serão trocadas de horário. Um movimento de Kempe válido seria trocar a primeira Cadeia de Kempe a_{11}, a_{21}, a_{22}, a_{13}, a_{23} e a_{15}, obtendo uma nova alocação das aulas nesses dois horários t_1 e t_2 mostrada na figura 2.

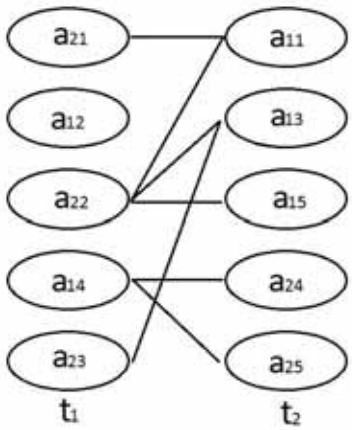

Figure 2: Exemplo de um movimento de Cadeia de Kempe nos horários t_1 e t_2

Dessa forma, a partir de uma solução válida podemos gerar vizinhos utilizando a estrutura de vizinhança Cadeia de Kempe. Os seguintes passos ilustram o procedimento de criação de um vizinho Kempe implementado neste trabalho:

1. Obter dois horários t_1 e t_2 distintos aleatoriamente;
2. Marcar as aulas indisponíveis para troca de horário;
3. Construir a matriz que representa os conflitos existentes nas aulas dos horários t_1 e t_2;
4. Obter a primeira aula da Cadeia de Kempe (aula que possui maior número de conflitos);
5. Construir um vetor contendo as posições da matriz que representam uma Cadeia de Kempe.

Um vizinho Kempe é uma estrutura que contém o conjunto de aulas que serão trocadas. Primeiro, dois horários são escolhidos aleatoriamente (item 1). Para que a tabela-horário permaneça viável após o movimento de Kempe, é feito uma análise nas aulas dos horários t_1 e t_2 (item 2) para verificar se existe alguma aula do horário t_1 que não pode ser movida para o horário t_2, ou vice-versa. Essa análise é realizada para que a restrição $disponibilidade$ não seja violada, condição muito importante na construção de uma Cadeia de Kempe válida.

Vale destacar que neste trabalho a tabela-horário é representada com uma matriz, onde as linhas representam as salas e as colunas representam os horários de todos os dias. As aulas são representadas por números inteiros. Cada aula da disciplina é representada por um número diferente. Se a primeira disciplina da instância possui cinco aulas semanais, então elas são representadas com os números 1, 2, 3, 4 e 5. Uma segunda disciplina com três aulas é representada na tabela-horário com os números 6, 7 e 8, e assim por diante. Horários vagos são representados com o valor -1.

A próxima etapa para gerar um vizinho Kempe é construir uma matriz de conflitos (item 3) para representar os conflitos entre as aulas dos dois horários escolhidos. Para cada par de aulas

$(i, j), i \in t_1$ e $j \in t_2$, a matriz é preenchida com $a_{ij} = 1$ caso as aulas sejam conflitantes, e $a_{ij} = 0$, caso contrário.

A matriz de conflitos exemplificada na tabela 2 é construída a partir da tabela 1, considerando t_1 o horário 1 do dia 0 e t_2, o horário 3 do dia 4.

	Dia 0				Dia 1				Dia 2				Dia 3				Dia 4			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
rA	-1	-1	-1	-1	**15**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**16**	-1
rB	**5**	**14**	**4**	**11**	-1	**6**	-1	-1	-1	-1	-1	-1	-1	-1	**12**	-1	-1	-1	-1	**6**
rC	-1	**2**	-1	-1	**3**	-1	**8**	**7**	-1	-1	-1	-1	**9**	**13**	-1	-1	-1	-1	**1**	-1

Table 1: Tabela-horário codificando as aulas com números inteiros

	16	**-**	**1**
-	0	0	0
14	1	0	1
2	1	0	1

Table 2: Matriz de conflitos

A partir da matriz de conflitos, se faz necessário escolher um conjunto de aulas para que sejam trocadas de horário. A primeira aula escolhida é aquela que possui o maior número de conflitos (item 4). Após a escolha da primeira aula para troca, começamos a construir toda a Cadeia de Kempe (item 5). Neste exemplo, a aula **14** foi escolhida para ser trocada de horário. Observe que a aula **14** conflita com as aulas **16** e **1**, logo, ao mover a aula **14** para o horário 3 do dia 4, devemos mover as aulas **16** e **1** para o horário 1 do dia 0. As aulas **16** e **1** conflitam com a aula **2**, logo, também devemos mover a aula **2** para o horário 3 do dia 4.

Inicialmente a lista de aulas que devem ser trocadas é construída de acordo com a matriz de conflitos (item 5) e o movimento de Kempe é realizado em um segundo momento.

A tabela-horário ilustrada pela tabela 1 se transforma naquela apresentada na tabela 3, após as trocas da Cadeia de Kempe formada pelas aulas $\{14, 16, 1, 2\}$.

	Dia 0				Dia 1				Dia 2				Dia 3				Dia 4			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
rA	-1	-1	-1	-1	**15**	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**14**	-1
rB	**5**	**16**	**4**	**11**	-1	**6**	-1	-1	-1	-1	-1	-1	-1	-1	**12**	-1	-1	-1	-1	**6**
rC	-1	**1**	-1	-1	**3**	-1	**8**	**7**	-1	-1	-1	-1	**9**	**13**	-1	-1	-1	-1	**2**	-1

Table 3: Tabela-horário após o movimento de Kempe

4. Resultados Computacionais

Os testes foram realizados em uma máquina com sistema operacional Ubuntu 14.04 e processador Intel Core2 Duo CPU T5800 @ 2.00GHz x 2. Foram usadas as 21 instâncias do ITC-2007 como base de teste para os algoritmos. Para cada instância, foram realizadas 10 execuções com semente de geração de números aleatórios diferente. Seguindo a diretriz do ITC-2007, o tempo máximo de execução do algoritmo foi definido pelo *software benchmark* [McCollum et al.(2010)] disponibilizado pela organização do campeonato, sendo que para a máquina utilizada nos testes o tempo máximo de execução foi 444 segundos.

4.1. Escolha dos Parâmetros

Foi considerado o subconjunto de instâncias: *comp01*, *comp17* e *comp05* para obter os valores dos parâmetros, pois estas instâncias foram classificadas, de acordo com a quantidade de conflitos, em fácil, intermediária e difícil, respectivamente.

O parâmetro $\alpha \in [0; 1]$ do GRASP influencia diretamente na geração da solução inicial. Para um valor de α próximo a zero, o algoritmo de construção da solução inicial é mais guloso produzindo boas soluções, mas pouco diversificadas. Para um valor de α mais próximo a um, as soluções geradas são mais diversificadas, porém são piores. Testes foram realizados variando α no intervalo de $[0,01; 0,5]$ e analisando os resultados escolhemos utilizar $\alpha = 0,15$. Para o *path-relinking* o tamanho máximo do conjunto de soluções elite foi fixado em 20, e o caminho parte de uma solução escolhida aleatoriamente no conjunto Elite em direção à solução obtida pela busca local do GRASP.

O algoritmo *Simulated Annealing* possui mais parâmetros a serem calibrados, como a temperatura inicial (T_i), a temperatura final (T_f), a taxa de decaimento (β) e o número de vizinhos (N_v) gerados a cada iteração.

O objetivo é escolher parâmetros que permitem um certo grau de diversificação no início da busca e maior intensificação no final do processo. Como a construção da solução inicial já produz uma solução de boa qualidade, a temperatura inicial não precisa ser muito alta. Foi observado que temperaturas iniciais mais altas não proporcionam melhoria no resultado final pois permite aceitar soluções muito ruins e a melhora é observada somente após várias etapas de resfriamento. Para as instâncias do ITC-2007, a busca estabiliza, em média, após a temperatura alcançar o valor $T = 0,01$ e raramente encontra-se uma solução melhor após esse ponto.

Sendo assim, os parâmetros utilizados foram: T_i = 1,5, T_f = 0,005, β = 0,999 e em cada temperatura são gerados N_v = 500 vizinhos.

4.2. Escolha da vizinhança para a geração de vizinhos

Para a escolha da porcentagem da aplicação de cada movimento na busca local, foram realizados testes com as instâncias *comp05*, *comp11* e *comp20* (uma instância difícil, uma fácil e uma intermediária), de acordo com as suas quantidades de conflitos. Desta forma, foram testadas as seguintes proporções de aplicação de movimentos. Em [Rocha(2013)] foi escolhida a proporção 50% para cada movimento (K_00). Para manter a igualdade de chances dos movimentos MOVE e SWAP, as demais proporções (K_10, K_20 e K_30) destacam a porcentagem $p\%$ para Kempe e considera $(100 - p)/2\%$ para os demais.

- (K_00) 0% para Kempe, 50% para *MOVE* e 50% para *SWAP*;
- (K_10) 10% para Kempe, 45% para *MOVE* e 45% para *SWAP*;
- (K_20) 20% para Kempe, 40% para *MOVE* e 40% para *SWAP*;
- (K_30) 30% para Kempe, 35% para *MOVE* e 35% para *SWAP*.

Os melhores resultados obtidos para cada teste nas três instâncias escolhidas são apresentados na tabela 4.

Instâncias	K_00	K_10	K_20	K_30
comp05	446	475	412	406
comp11	0	0	0	0
comp20	122	120	127	136
média	189	198	**180**	181

Table 4: Melhores resultados do GRASPK para diferentes porcentagens das vizinhanças

Observando o resultado obtido na tabela 4, percebemos que, na média, a utilização da proporção de vizinhos K_20, ou seja, Kempe 20%, *MOVE* 40% e *SWAP* 40%, gera resultados

melhores do que as outras proporções e por isso decidimos utilizar essa proporção para a realização dos demais testes.

Além disso, é possível notar na tabela 4, que a aplicação da Cadeia de Kempe é viável, pois em média, a utilização dessa vizinhança na proporção de 20% e 30% encontrou resultados melhores do que os resultados obtidos pelo GRASP sem a utilização dessa vizinhança (Kempe 0%).

4.3. Comparação dos Resultados

Para as execuções do GRASP com Kempe (GRASPK), mantemos a geração de vizinhos a proporção de 20% para Kempe, 40% para *MOVE* e 40% para *SWAP*. As tabelas 5 e 6 apresentam a comparação do GRASPK, desenvolvido neste trabalho, com os melhores resultados obtidos pelos competidores na primeira e segunda fase do ITC-2007, respectivamente.

As colunas estão ordenadas pelas médias das respostas obtidas. Ao final de cada tabela foi adicionada uma coluna com os resultados obtidos pelo algoritmo GRASPK. A melhor resposta de cada instância está destacada em negrito. Apenas os finalistas da primeira fase (5 primeiros colocados) do ITC-2007 tiveram seus nomes divulgados e foram convidados a continuar na segunda fase.

Pode-se notar na tabela 5, que para os 7 primeiros colocados, o GRASPK está entre o quinto e o sexto lugar. As melhores respostas foram obtidas para as instâncias *comp01* e *comp11*. Na segunda fase da competição, os cinco primeiros colocados da primeira fase participaram executando seus respectivos algoritmos com as instâncias comp15 a comp21. Observamos na tabela 6, que o GRASPK nesta segunda fase está entre o quarto e o quinto colocado .

Instância	Muller	Lu	Atzuna	Clark	Geiger	6º	7º	GRASPK
comp01	**5**	**5**	**5**	10	**5**	9	23	**5**
comp02	43	**34**	55	83	108	154	86	117
comp03	72	**70**	91	106	115	120	121	129
comp04	**35**	38	38	59	67	66	63	65
comp05	**298**	**298**	325	362	408	750	851	389
comp06	**41**	47	69	113	94	126	115	123
comp07	**14**	19	45	95	56	113	92	77
comp08	**39**	43	42	73	75	87	71	81
comp09	103	**102**	109	130	153	162	177	145
comp10	**9**	16	32	67	66	97	60	61
comp11	**0**	**0**	**0**	1	**0**	**0**	5	**0**
comp12	331	**320**	344	383	430	510	828	460
comp13	66	**65**	75	105	101	89	112	118
comp14	53	**52**	61	82	88	114	96	99
Média	**79,21**	**79,21**	92,21	119,21	126,14	171,21	192,86	133,5

Table 5: Resultados da primeira fase do ITC-2007

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou o problema de tabela-horário para universidades, baseado no currículo dos cursos, de acordo com o campeonato internacional de tabela-horário (ITC-2007). Para isso, o trabalho realizado em [Rocha(2013)] que utiliza a meta-heurística GRASP com *Simulated Annealing* como busca local, foi estendido. Uma nova estrutura de vizinhança foi acrescentada, esta sendo mais complexa do que as vizinhanças *MOVE* e *SWAP*, com o objetivo de melhorar o desempenho das soluções do problema de tabela-horário geradas pela versão do GRASP antes de incluir a vizinhança da Cadeia de Kempe (GRASPK).

Instância	Muller	Lu	Atzuna	Clark	Geiger	GRASPK
comp15	84	**71**	82	128	119	127
comp16	**34**	39	40	81	84	99
comp17	**83**	91	102	124	152	139
comp18	83	69	**68**	116	110	99
comp19	**62**	65	75	107	111	99
comp20	**27**	47	61	88	144	125
comp21	**103**	106	123	174	169	149
Média	**68**	69,71	78,71	116,86	127	119,57

Table 6: Resultados da segunda fase do ITC-2007

A versão GRASPK apresentou melhoria na qualidade de soluções geradas em relação a versão sem a Cadeia de Kempe, mas o método de escolha das vizinhanças deve ser mais ajustado, variando a porcentagem de uso de cada vizinhança, ou escolhendo uma vizinhança aleatória e permanecendo na mesma vizinhança até que uma quantidade de vizinhos sem melhora seja alcançada. Os primeiros colocados do ITC-2007 apresentaram resultados superiores possivelmente devido à maior velocidade de execução de cada iteração e ao uso de vizinhanças específicas para cada restrição fraca.

Como trabalho futuro, sugere-se aumentar a velocidade da execução de uma iteração através de melhoras no cálculo da função objetivo e/ou através do uso de outras estruturas de dados. Outras estratégias busca podem ser desenvolvidas, assim como a utilização da Cadeia de Kempe como forma de diversificação ao invés de utilizá-la na busca local. Ou seja, a Cadeia de Kempe por ser mais complexa e possuir um custo computacional maior do que as vizinhanças *MOVE* e *SWAP*, pode ser utilizada após um número de iterações sem melhora, como forma de diversificação. [Tuga et al.(2007)] utilizaram dessa estratégia e obtiveram resultados muito significativos ao utilizar essa técnica no *Simulated Annealing*.

References

Burke, E., Elliman, D., and Weare, R. (1994). A genetic algorithm based university timetabling system. In *Proceedings of the 2nd East-West International Conference on Computer Technologies in Education*, volume 1, pages 35–40.

Burke, E. K. and Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280.

Carter, M. W., Laporte, G., and Lee, S. Y. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, pages 373–383.

Daskalaki, S., Birbas, T., and Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153(1):117–135.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). In *Proceedings of the 14th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, Rome, Italy*.

Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8.

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

Glover, F. (1998). A template for scatter search and path relinking. In *Artificial evolution*, pages 1–51. Springer.

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

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*.

Laguna, M. and Marti, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., and Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130.

Mcmullan, P. (2007). An extended implementation of the great deluge algorithm for course timetabling. In *Computational Science–ICCS 2007*, pages 538–545. Springer.

Mulvey, J. M. (1982). A classroom/time assignment model. *European Journal of Operational Research*, 9(1):64–70.

Nascimento, M. C., Resende, M. G., and Toledo, F. (2010). Grasp heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, 200(3):747–754.

Rocha, W. S. (2013). Algoritmo grasp para o problema de tabela-horário de universidades. Master's thesis, UNIVERSIDADE FEDERAL DO ESPIRITO SANTO.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*.

Shaker, K. and Abdullah, S. (2009). Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 149–153. IEEE.

Thompson, J. M. and Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7):637–648.

Tuga, M., Berretta, R., and Mendes, A. (2007). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pages 400–405.