RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE CURSOS UNIVERSITÁRIOS BASEADA EM CURRÍCULOS VIA UMA META-HEURÍSTICA HÍBRIDA GRASP-ILS-RELAXADO

Saulo Henrique D'Carlos Barbosa[1], **Sergio Ricardo de Souza**[1]

[1]Programa de Modelagem Matemática e Computacional,
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675, CEP 30510-000, Belo Horizonte (MG), Brasil

saulo_comp@yahoo.com.br, sergio@dppg.cefetmg.br

Resumo. *Este artigo aborda o problema de programação de cursos universitários baseada em currículos. A resolução proposta é feita através de uma meta-heurística híbrida, que utiliza GRASP para a geração de uma solução inicial e* Iterated Local Search *(ILS) com relaxamento para o refinamento da solução. O problema consiste em alocar um período/sala para todas as aulas de todas disciplinas, evitando conflitos entre disciplinas de um mesmo currículo. Neste trabalho, são considerados aspectos e instâncias da International Competition Timetabling - ITC2007, pelo fato do problema tratado ser bem definido e amplamente testado. Sendo o problema NP-difícil, a aplicação de métodos meta-heurísticos para sua resolução se justifica, por apresentar boas soluções sem um elevado custo computacional. Os resultados computacionais obtidos foram satisfatórios frente aos encontrados na literatura e também mostraram a influência positiva do procedimento de relaxamento no algoritmo híbrido proposto.*

PALAVRAS-CHAVE*: Programação de cursos universitários baseada em currículos, Meta-Heurística Híbrida, ITC 2007*

Abstract. *This paper addresses the Curriculum Based Course Timetabling Problem. The proposed solution is conducted through a hybrid meta-heuristic technique that uses GRASP to generate an initial solution and ILS with relaxation (improving technique) for the refinement of the solution. The problem consists in allocating a time/room for all classes in all disciplines, avoiding conflicts between subjects of the same curriculum. In this work, aspects and instances of ITC 2007 are considered, because the problem addressed is well defined and extensively tested in the competition. Being a NP-hard problem, the application of meta-heuristic methods for its resolution has been justified, because they present good solutions without a high computational cost. The computational results were satisfactory in view of the results of the literature and also show the positive influence of the relaxation procedure in algorithm.*

KEYWORDS*: Curriculum based Course Timetabling, Hybrid Meta-heuristic, ITC 2007*

1 INTRODUÇÃO

De acordo com Willemen (2002), o Problema de Programação de Horários (PPH), referenciado também como *Timetabling Problem*, abrange todas as atividades que envolvem o processo de construção de quadros de horários. O PPH é de grande aplicação nas mais variadas áreas, tais como indústrias, hospitais, empresas de transporte, eventos esportivos, escolas e universidades.

O Problema de Programação de Horários Educacionais (PPHE) é uma subclasse do PPH, que trata do problema de agendar eventos que ocorrem em instituições educacionais. Qu (2002) define que o problema consiste em alocar vários eventos (disciplinas, reuniões, exames etc.) para um número limitado de recursos (período de tempo, espaço físico etc.), respeitando ao máximo um conjunto pré-estabelecido de restrições.

Segundo a classificação apresentada por Schaerf (1999), a abordagem do presente artigo é classificada como um Problema de Programação de Horários de Cursos (*Course Timetabling*). Trata-se de um problema geralmente encontrado em universidades, que apresentam, como característica, a possibilidade do aluno escolher quais disciplinas serão cursadas a cada semestre, sendo as possíveis disciplinas limitadas pelo currículo ao qual o aluno pertence. Na variação estudada neste artigo, busca-se alocar um período de tempo e uma sala para cada aula de todas as disciplinas, levando-se em conta os conflitos entre disciplinas do mesmo currículo.

Em 2007, foi realizada a Segunda Competição Internacional de Programação de Horários (*International Timetabling Competition - ITC 2007*). O objetivo era criar diferentes técnicas para resolução de problemas de PPHE. Segundo Spindler (2010), a realização deste evento é considerado como um importante marco para a classe do problema, uma vez que não é comum autores realizarem testes computacionais e comparações entre diferentes técnicas. Assim sendo, esse trabalho faz uso não só das instâncias fornecidas pelo sítio de *Web* do evento, bem como respeita as regras da competição para a realização dos experimentos.

De acordo com Burke e Petrovic (2002), a classe de problema *Timetabling* tem atraído a atenção da comunidade científica nos últimos quarenta anos devido a sua importância e à complexidade de resolução do problema em tela (NP-Completo). Dentre uma grande variedade de abordagens (métodos sequenciais, métodos baseados em restrições, etc.) para o problema, têm se destacado a utilização de técnicas meta-heurísticas. Meta-heurísticas são métodos que partem de uma ou mais soluções e empregam estratégias de busca que tentam evitar um mínimo local. Alguns trabalhos relevantes que utilizam meta-heurísticas para o resolução dessa classe de problemas são: Muller (2008), que utiliza *Simulated Annealing*, aliado a outros métodos de busca local e análise estatística; Lü e Hao (2010), que utiliza a metaheurística Busca Tabu com aperfeiçoamentos; Barbosa et al. (2010), que utiliza uma implementação híbrida GRASP e Busca Tabu; Souza (2000), que realiza o teste de várias meta-heurísticas; e Bai et al. (2008), que utiliza *Simulated Annealing* com aperfeiçoamentos. Destacam-se também os trabalhos de Schaerf (1999) e Burke e Petrovic (2002), que realizam uma revisão sobre a automação dos problemas *Timetabling*.

O presente trabalho tem o objetivo de abordar o Problema de Programação de Disciplinas Universitárias baseada em Currículos, levando-se em conta os aspectos do impostos pelo *ITC 2007*. É proposta uma técnica híbrida GRASP e ILS com um aperfeiçoamento, nomeado de *relaxação*, de modo que os resultados dos testes computacionais realizados

possam ser comparados com os melhores resultados obtidos na literatura, além de verificar a eficiência do aperfeiçoamento criado. A realização do trabalho é justificada pela importância do problema e também pela colaboração teórica ao tema. Inicialmente, é apresentada uma introdução sobre o tema abordado. Posteriormente, o problema de programação de disciplinas universitárias baseada em currículos é discutido e detalhado. Em seguida, a metodologia utilizada para aplicação das técnicas pesquisadas é mostrada. Os resultados obtidos são apresentados em seguida, juntamente com as comparações com os resultados da literatura. A conclusão e propostas de trabalhos futuros finalizam o artigo.

2 PROGRAMAÇÃO DE CURSOS UNIVERSITÁRIOS BASEADA EM CURRÍCULOS

O Problema de Programação de Cursos Universitários baseada em Currículos (*Curriculum based Course Timetabling*), é uma uma variação do PPHE, descrita na terceira modalidade de problemas da competição *International Timetabling Competition 2007*, conforme `http://www.cs.qub.ac.uk/itc2007/`.

De acordo com Di Gaspero e Schaerf (2007), o problema consiste em realizar a programação semanal de disciplinas de vários cursos universitários, dentro de um determinado número de períodos e salas disponíveis em que os conflitos entre as disciplinas são determinados pelos currículos em que as mesmas estão envolvidas, além de outras restrições que determinam a viabilidade e qualidade de um quadro de horários construído. De acordo com De Cesco et al. (2010), as principais entidades envolvidas no problema são:

- Dias, Horários e Períodos: dado um número de dias letivos, cada dia é dividido em um dado número de períodos, sendo um horário correspondente ao par dia/período;
- Currículos: um currículo é composto por um grupo de disciplinas que possam ter estudantes em comum, e, desse modo, as disciplinas de um mesmo currículo não devem ser agendadas em um mesmo período;
- Disciplinas e Professores; cada disciplina apresenta um professor alocado para lecioná-la (previamente definido pela instituição); o número de aulas da mesma; o número de alunos; o mínimo de dias exigidos para oferta da disciplina; e, também, um conjunto de horários em que a mesma não está disponível para alocação;
- Salas: cada sala apresenta sua respectiva capacidade;

Uma solução para o problema é dada pela atribuição de um período de tempo e uma sala para todas as aulas de todas as disciplinas. A avaliação de uma solução é realizada considerando-se as restrições básicas, que são essenciais, pois estão ligadas a viabilidade, e também as restrições opcionais que são almejadas para melhorar a qualidade de uma solução particular e que, contudo, podem ser desrespeitadas, caso não exista outra alternativa. As restrições consideradas pelo problema serão apresentadas na subseção 3.3 deste artigo.

3 METODOLOGIA

3.1 Representação da Solução

O problema apresenta q disciplinas na forma $d_1, \ldots, d_q$; p períodos $z_1, \ldots, z_p$; e m salas $s_1, \ldots, s_m$. Cada disciplina d_i consiste em l_i aulas, agendadas em períodos distintos e atendendo a_i alunos. Cada sala s_j apresenta uma capacidade cap_j em termos de número de lugares. O problema apresenta também g grupos de disciplinas, os quais são chamados

currículos. As disciplinas de um currículo são agrupadas por apresentarem estudantes em comum e, assim sendo, não devem ser agendadas em um mesmo período.

A representação de uma solução é dada por uma matriz T de inteiros $q \times p$. Neste caso, $T_{ik} = j$ (com $1 \leq j \leq m$) significa que a disciplina d_i têm aula na sala r_j no período k e $T_{ik} = -1$ significa que a disciplina d_i não possui nenhuma aula alocada no período k. Ou seja, uma solução é dada por uma matriz que relaciona disciplina por períodos, sendo esta matriz preenchida com valores inteiros, que representam em qual sala a disciplina será ministrada. Através desta representação é possível obter a grade de horário das disciplinas, bem como o escalonamento de salas para as mesmas.

3.2 Vizinhança

Para exploração do espaço de soluções, vários tipos de movimentos foram testados, de tal que sorte que aqueles usados foram os que tiveram impacto nos resultados obtidos. Após testes empíricos, foram adotados os seguintes movimentos:

- **MovimentoPeriodo**: uma disciplina tem uma de suas aulas selecionada aleatoriamente. Esta, então, tem seu período alterado para um outro período disponível, selecionado também aleatoriamente, em que não haja conflito com outras disciplinas do mesmo currículo;
- **MovimentoSala**: uma disciplina tem uma de suas aulas selecionada aleatoriamente. Esta tem sua sala alterada para a primeira sala não conflitante (em termos de ocupação);
- **MovimentoAula**: Uma disciplina tem uma de suas aulas selecionada. Então, um novo período para a aula é selecionado também e por fim uma nova sala é escolhida. Todos estes são selecionados de forma aleatória. A aula tem seu período alterado e também uma nova sala é alocada. Caso não haja conflitos, apenas este movimento é realizado. Caso haja, a aula da disciplina conflitante tem uma troca de horários entre os horários envolvidos no movimento;
- **MovimentoEstabilidadeSala**: este movimento consiste em escolher uma nova sala aleatoriamente e atribuir a mesma para todos horários de uma determinada disciplina. É utilizado no intuito de diminuir o número de violações a respeito da restrição estabilidade de salas. Caso haja conflitos de sala após a alocação, as salas das aulas das disciplinas conflitantes são trocadas pelas salas que estavam na disciplina que recebeu o movimento;
- **MovimentoCurriculoCompacto**: uma aula isolada (ou seja, que não possui aulas do mesmo currículo alocadas de forma adjacente) é escolhida aleatoriamente e alocada em um horário adjacente a outra aula qualquer do currículo. Este movimento é utilizado no intuito de diminuir o número de violações a respeito da restrição aulas isoladas;
- **MovimentoTrocaDeAulasESalas**: este movimento consiste em selecionar duas disciplinas que utilizam a mesma sala em horários diferentes. Após a escolha das disciplinas envolvidas, as mesmas têm seus horários alternados, mantendo a alocação para a sala.

Os quatro primeiros movimentos são baseados nos movimentos adotados por Muller (2008). Este autor apresenta uma abordagem para o problema através de uma técnica híbrida que utiliza meta-heurísticas e bases estatísticas. Os movimentos propostos por

Muller (2008), juntamente com o quinto movimento, aqui proposto, foram os que apresentaram maior eficiência para exploração do espaço de busca dentre os diferentes tipos de movimentos testados.

3.3 Função de Avaliação

A função de avaliação tem, como principal tarefa, realizar a avaliação de uma solução gerada, considerando uma variedade de fatores relativos às salas, às turmas e às restrições da instituição de ensino. Após a análise da solução, a função de avaliação deve retornar um valor numérico, que representa o quanto a solução é considerada boa.

A função de avaliação conceitua um quadro de horários, analisando a qualidade e à viabilidade de uma solução encontrada. Esta análise é feita de modo a verificar as violações das restrições estabelecidas, restrições estas que são divididas em restrições básicas e restrições opcionais. Para o problema abordado, foram adotadas as restrições descritas em De Cesco et al. (2010), as quais foram consideradas no *ITC 2007*. As restrições básicas para o problema são:

- **R01 - Restrição Aulas**: determina que todas as aulas de uma disciplina devem ser agendadas e alocadas em períodos distintos. Uma violação ocorre se alguma aula da disciplina não é agendada ou se duas aulas distintas são agendadas no mesmo período;
- **R02 - Restrição Conflitos**: as aulas de disciplinas de um mesmo currículo ou aulas de duas ou mais disciplinas que possuem o mesmo professor não devem ser agendadas para um mesmo período. Um período com duas disciplinas conflitantes conta como uma violação. Um período com três disciplinas conflitantes conta como duas violações, sendo uma violação para cada par de disciplinas;
- **R03 - Restrição Ocupação das Salas**: duas ou mais aulas distintas não podem ocupar a mesma sala em um mesmo período, sendo considerado como o número de violações cada sala extra encontrada no período conflitante;
- **R04 - Restrição Disponibilidade**: caso o professor que leciona a disciplina não tenha disponibilidade em determinando período, nenhuma aula da disciplina deve ser alocada no mesmo. Cada aula agendada em um período com indisponibilidade do professor representa uma violação.

As restrições opcionais para o problema são:

- **R05 - Restrição Capacidade da Sala**: para cada aula agendada, a sala alocada deve possuir capacidade maior ou igual à exigida pela disciplina. Nesta restrição, para cada estudante a mais que a capacidade da sala, têm-se uma violação;
- **R06 - Restrição Mínimo de Dias**: as aulas de uma disciplina devem estar agendadas em um número mínimo de dias pré-estabelecido. Para cada dia a menos que o mínimo estabelecido, têm-se uma violação da restrição;
- **R07 - Restrição Aulas Isoladas**: aulas de disciplinas de um mesmo currículo devem ser agendadas sempre de maneira adjacente, ou seja, em horários consecutivos de um mesmo dia. Nesta restrição, os horários alocados para um currículo de disciplinas são analisados, e cada disciplina que não é adjacente a nenhuma outra conta como uma violação;
- **R08 - Restrição Estabilidade da Sala**: todas as aulas de uma disciplina devem ocorrer em uma mesma sala. Cada sala diferente utilizada por uma disciplina conta como uma violação.

Algoritmo 1: Descida Randômica por Disciplina

Entrada: *SolucaoCorrente*, *NumVizinhos*, *MaxIteracoes*
Saída: $SolucaoV$
início
  $Iteracao \leftarrow \emptyset$; $SAux \leftarrow \emptyset$; $SolucaoV \leftarrow SolucaoCorrente$
  enquanto ($Iteracao \leq MaxIteracoes$) **faça**
    $Iteracao \leftarrow Iteracao + 1$;
    Seleciona-se uma disciplina na lista de disciplinas disponíveis para geração da vizinhança;
    $SAux \leftarrow$ *Melhor vizinho entre os NumVizinhos gerados a partir de SolucaoVizinha*; **se** $SAux \leq SolucaoV$ **então**
      $SolucaoV = SAux$;
    fim
  fim
  Retorne $SolucaoV$;
fim

O valor dado à violação de uma restrição está diretamente ligado a relevância desta no problema. Dessa forma, as restrições básicas possuem um valor de peso alto para que sejam prioritárias. Nesta implementação, as mesmas receberam peso igual à 1000. Já as restrições opcionais recebem pesos menores, que se diferenciam de acordo com as prioridades da instituição. Nesta implementação, as restrições $R05$ e $R08$ receberam peso 1; a restrição $R06$ recebeu peso 5 e a restrição $R07$ recebeu peso igual a 2. As definições dos pesos também são feitas no trabalho de De Cesco et al. (2010). A função de avaliação é representada, então, pela expressão:

$$F(X) = w_v \times f_v(X) \tag{1}$$

em que V é o conjunto de restrições do problema, com elementos $v \in V$ identificados por $v = 1, \ldots, \bar{v}$; $f_v(X)$ é a função de avaliação da restrição R_v; e w_v é o peso da restrição de avaliação $f_v(X)$.

3.4 Busca Local

Sendo o espaço de busca para o problema muito grande, devido a quantidade de combinações e movimentos possíveis na exploração do espaço, a heurística de busca local adotada foi baseada no Método Randômico de Descida descrito em Souza (2000). Trata-se de um método que não necessariamente pesquisa toda a vizinhança de uma solução (como ocorre no método de descida) e, dessa forma, evita a pesquisa exaustiva que, neste problema, demandaria um alto custo computacional.

O método implementado consiste em analisar um conjunto de vizinhos gerados a partir de movimentos aleatórios, sendo o melhor vizinho encontrado comparado com a solução corrente. A solução vizinha somente será a nova solução corrente caso seja melhor ou igual à anterior. Para a geração da vizinhança, a cada iteração todos os movimentos descritos na seção 3.2 têm a mesma probabilidade de ocorrência. O procedimento é repetido por um número fixo de iterações, destacando-se que, devido ao vasto espaço de busca, a cada iteração são gerados vizinhos de somente uma disciplina. A disciplina é escolhida a cada iteração de forma aleatória em uma estrutura de dados do tipo lista. Após ser escolhida, a mesma é removida para que todas as disciplinas passem pelo processo de busca local com a mesma frequência. Ressalta-se que, quando a lista de disciplinas ficar vazia, é preenchida novamente com todas as disciplinas em ordem aleatória. Neste caso, são gerados dez vizinhos a cada iteração e o número de iterações é variável de acordo com o tamanho da instância, sendo igual a cinco vezes o número de disciplinas. O pseudo-código da busca local utilizada é apresentado no Algoritmo 1.

3.5 Solução Inicial GRASP

A meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) foi utilizada parcialmente para uma implementação híbrida juntamente com a meta-heurística *Iterated Local Search* (ILS). O GRASP é uma meta-heurística proposta por Feo e Resende (1995) que, conforme os autores descrevem, é um método iterativo, que consiste em duas fases: uma fase de construção da solução via um procedimento guloso-aleatório (lista restrita de candidatos e parâmetro α) e uma fase de busca local, na qual um ótimo local da vizinhança da solução construída é pesquisado. Neste trabalho, fez-se uso apenas da fase de construção de uma solução através do GRASP. Sua utilização é justificada por apresentar soluções mais eficientes que outros métodos de construção.

O procedimento implementado neste trabalho realiza as alocações de todas as aulas de cada disciplina uma a uma. A alocação é feita a partir de cada disciplina, de modo que todas as aulas de uma disciplina são alocadas, para somente assim realizar as alocações da próxima disciplina. Sendo assim, a ordem da lista de disciplinas candidatas (LDC) é de suma importância no procedimento. O ordenamento das disciplinas é determinado por uma função que considera:

- o número de currículos que a disciplina está envolvida, uma vez que quanto maior o número de currículos que uma disciplina está envolvida, maior a chance desta apresentar conflitos;
- o número de indisponibilidades de horários da disciplina;
- a quantidade de alunos na disciplina, pois disciplinas que possuam um alto número de alunos provavelmente terão poucas salas capacitadas.

As alocações serão realizadas para todas as disciplinas, respeitando-se o ordenamento da LDC. Para cada disciplina, têm-se o número de aulas a serem alocadas, sendo a alocação feita de forma a testar todas as possíveis salas em todos os possíveis horários. Funções especiais de avaliação (que consideram as mesmas restrições do problema) foram criadas para testar as possíveis alocações. Dessa forma, uma lista de possíveis alocações candidatas (LPAC) é criada de forma ordenada, de acordo com o valor de avaliação de cada possível alocação. O parâmetro α tem a função de delimitar a LPAC, criando a lista de possíveis alocações candidatas restritas (LPACR). Assim, seleciona-se aleatoriamente

Algoritmo 2: GRASP

Entrada: α, *TotalAulas*,
Saída: $Solucao$
início
  $aulasAlocadas \leftarrow 0$;
  $Solucao \leftarrow$ *inicia sem nenhuma aula alocada;*
  $LDR \leftarrow$ *Disciplinas ordenadas pela probabilidade de violação;*
  enquanto (*aulasAlocadas* $<$ *TotalAulas*) **faça**
    $disciplinaEscolhida \leftarrow$ *primeira posição da LDR*;
    para *cada aula da disciplina escolhida* **faça**
      $LPAC \leftarrow$ *todas as possíveis alocações (sala/período)*;
      $LPACR \leftarrow$ *as* α *melhores possíveis alocações de LPAC*;
      $alocacaoEscolhida \leftarrow$ *alocação escolhida aleatoriamente em LPACR*;
      $Solucao \leftarrow Solucao$ *recebe* $alocacaoEscolhida$;
      $aulasAlocadas \leftarrow aulasAlocadas + 1$;
    fim
    $LDR \leftarrow LDR -$ *primeira posição*
  fim
  Retorne $Solucao$
fim

uma alocação na LPACR e atribui-se a respectiva alocação na solução que está sendo construída. As respectivas atribuições são realizadas até que todas as disciplinas tenham todas as aulas alocadas (ou seja, até que uma solução seja construída). A LPACR é composta pelas α melhores possíveis alocações (aspecto guloso do método); entretanto, a alocação é escolhida de maneira aleatória, o que significa que, a cada execução, uma solução diferente será criada. Na implementação deste trabalho foi utilizado o valor $\alpha = 2$. O pseudo-código do método é motrado no Algoritmo 2.

3.6 Iterated Local Serch (ILS)

A meta-heurística ILS (*Iterated Local Search*), segundo Lourenço et al. (2003), é um método de busca iterativa que faz uso de pertubações da solução (alterações na solução corrente), tendo como principal objetivo a diversificação da busca, de modo a escapar e visitar diferentes ótimos locais.

Quatro são os principais componentes que definem o método ILS: geração da solução inicial, busca local, pertubação e critério de aceitação. Primeiramente, a geração da solução inicial foi realizada de forma aleatória e, posteriormente, através do GRASP, conforme será descrito na subseção 3.7. Já a busca local é realizada através do Método de Descida Randômica por Disciplina, descrito no Algoritmo 1.

A pertubação foi realizada da seguinte maneira: os movimentos de geração de vizinhança, apresentados na seção 3.2, são utilizados para aplicação em uma solução corrente, além de um novo movimento, descrito a seguir. A pertubação utilizada apresenta dois níveis diferentes, sendo que o nível 1 corresponde a 2 movimentos e o nível 2 a 3 movimentos, os quais são escolhidos aleatoriamente e aplicados à solução corrente.

O novo movimento criado para a pertubação é denominado *constroiGRASPDisciplina*. Este movimento consiste em desalocar todas as aulas de determinada disciplina e alocá-las novamente, utilizando parte da meta-heurística GRASP construída para criação da solução inicial. Este movimento apresentou grande importância, pois é capaz de perturbar a solução em vários horários ao mesmo tempo, porém, mantendo a qualidade da solução.

Os cinco primeiros movimentos apresentam probabilidade de ocorrência de $\frac{2}{11}$, enquanto o movimento *constroiGRASPDisciplina* apresenta probabilidade $\frac{1}{11}$ de ocorrer. Esta atribuição é dada pelo alto custo computacional do novo movimento proposto.

O critério de aceitação verifica se a solução encontrada após a pertubação e busca local, necessariamente nessa ordem, é melhor ou igual que a melhor solução encontrada até então ($solucao_{best}$); caso seja, a solução encontrada passa a ser a nova $solucao_{best}$. É importante ressaltar que, a cada quatro iterações sem melhora ou igualdade na $solucao_{best}$, o nível da pertubação é alterado, sendo a exploração feita de maneira circular entre os níveis de pertubação.

A estrutura bem como o funcionamento do ILS implementado poderá ser melhor entendido no pseudo-código apresentado no Algoritmo 3.

3.7 Algoritmo GRASP-ILS-Relaxado

O algoritmo implementado nesse trabalho é uma técnica híbrida que utiliza GRASP e ILS com modificações. A hibridização entre essas técnicas pode ser vista na literatura

Algoritmo 3: GRASP-ILS-Relaxado

```
   Entrada: α, TotalAulas, NumVizinhos, MaxIt , TempoDeExecucao, iterPorNivel,numDeNiveis
   Saída: Solucao
 1 início
 2     relacacao ← 0;
 3     nivel ← 1;
 4     tempo ← 0;
 5     iteracaoSemMelhora ← 0;
 6     SAux ← solução inicia sem nenhuma aula alocada;                    // Solução Auxiliar
 7     SBest ← GRASP(α, TotalAulas)                                  // Melhor solução Corrente
 8     S* ← SBest;                                                     // Melhor solução Global
 9     enquanto (tempo < TempoDeExecucao) faça
10         SAux ← pertubacao(nivel, SBest);
11         SAux ← descidaRandomicaPorDisciplina(SAux, NumVizinhos, MaxIt);
12         se SAux ≤ (SBest + relaxacao) então
13             se SAux < (SBest + relaxacao) então
14                 relaxacao ← 0;
15                 nivel ← 1;
16                 iteracaoSemMelhora ← 0
17             fim
18             SBest ← SAux;
19             se SBest < S* então
20                 S* ← SBest;
21             fim
22         senão
23             iteracaoSemMelhora ← iteracaoSemMelhora + 1
24             se iteracaoSemMelhora resto iterPorNivel == 0 então
25                 nivel ← nivel + 1;
26                 se nivel > numDeNiveis então
27                     nivel ← nivel + 1;
28                     relaxacao ← relaxacao + 1;
29                 fim
30             fim
31         fim
32     fim
33     Retorne Solucao
34 fim
```

em trabalhos como Arroyo et al. (2009) e Talbi (2009). O GRASP é utilizado para geração da solução inicial e sua utilização é de suma importância, pois influência diretamente nos resultados obtidos. Ressalta-se também que apenas uma solução é construída como solução inicial. A princípio, várias soluções eram geradas e a com melhor valor de avaliação era escolhida como solução inicial; porém, para algumas instâncias esta prática demandou um alto tempo computacional, o que inviabilizou sua utilização.

A partir da solução inicial encontrada, uma busca local é realizada, a fim de melhorar ainda mais a solução inicial. Em seguida, o ILS é aplicado, a partir da solução inicial, explorando o espaço de soluções com busca local e pertubações. Como condição de parada, foi utilizado o tempo de processamento, que é uma das entradas do algoritmo.

Uma modificação foi feita na estrutura do ILS a fim de melhorar os resultados obtidos. A modificação trata de uma condição para aceitar (relaxamento) soluções de piora depois de um certo número de soluções sem melhora. O relaxamento ocorre quando uma solução passa pelos dois níveis de pertubação do ILS sem nenhuma solução que seja menor ou igual à melhor solução corrente. Esta estratégia é um aperfeiçoamento no critério de aceitação da meta-heurística ILS.

O controle do limite de piora para uma solução em que ocorre o relaxamento é feito através da variável *relaxacao*. Esta variável recebe zero toda vez que exista uma melhora

ou igualdade; caso não ocorra por oito iterações do ILS, a variável é incrementada. A variável é utilizada juntamente com $solucao_{best}$ no critério de aceitação do ILS, mas seu valor é diferente de nulo somente quando uma solução permanece por algumas iterações sem modificação alguma (na maioria das vezes, quando está presa em uma bacia de atração).

Esta modificação é justificada pelo fato do problema estudado ser caracterizado por vários ótimos locais, os quais dificultam a exploração do espaço de soluções. Na seção 4 serão apresentados resultados que validam a utilização desta técnica. O pseudo-código do algoritmo implementado é apresentado pelo Algoritmo 3.

4 Resultados Computacionais

As implementações descritas neste artigo foram feitas sobre a plataforma Java, versão JDK 6.0, utilizando o IDE Netbeans 6.8. Os parâmetros das técnicas envolvidas foram mencionados na seção 3, sendo todos alcançados através de testes empíricos. Como critério de parada, foi utilizado um tempo máximo de 351 segundos, sendo este tempo calculado de acordo com o *hardware* em que as instâncias serão executadas. O cálculo é feito através de um *software* fornecido no sítio `http://www.cs.qub.ac.uk/itc2007/index_files/benchmarking.htm` e faz parte do regulamento do ITC 2007, sendo estas e outras informações descritas no trabalho de Di Gaspero e Schaerf (2007).

Foram testadas 21 instâncias reais da Universidade de Udine da Itália. Essas instâncias são fornecidas pelo sítio `http://tabu.diegm.uniud.it/ctt/index.php?page=instances`. Os testes foram realizados em um computador com processador Core 2 duo 2.5 Ghz, memória RAM de 2GB e sistema operacional Windows XP 32 bits. Foram 15 execuções sobre cada instância e os resultados obtidos são apresentados juntamente com os resultados dos cinco primeiros colocados no ITC 2007, para fins comparativos, na Tabela 1. Dessa forma, esta Tabela apresenta os melhores valores encontrados para cada instância, juntamente com os valores médios das execuções da técnica com e sem *relaxação*.

Os resultados obtidos foram satisfatórios, tendo em vista que na maioria das instâncias os valores alcançados foram superiores aos quarto e quinto colocados na Compteição ITC-2007. Para duas instâncias de testes, obteve-se o melhor valor encontrado até então. Em todas as soluções são alcançadas soluções factíveis, uma tarefa difícil em algumas instâncias pela alta probabilidade de conflitos.

Os valores médios alcançados através da técnica refletem a eficiência da mesma se comparada com a média de valores alcançados pelos competidores (valores que não são apresentados nesse trabalho pela limitação de espaço, mas podem ser encontrados na página da competição). Logo, a partir de comparações realizadas, é possível notar que o algoritmo criado apresenta uma eficiência superior a do quarto e quinto colocados, tendo valores médios superiores (menores no aspecto do problema) às dos competidores.

Quanto a qualidade da solução, a utilização da relaxação, se tornou uma boa alternativa, considerando que os resultados médios obtidos foram significativamente superiores aos alcançados pela execução sem a utilização da técnica. Em média, para todas as instâncias, a utilização da técnica proporcionou uma melhora de $11,69\%$ nos resultados obtidos.

Quanto à factibilidade de uma solução, destaca-se a utilização do GRASP: a técnica constrói uma solução inicial com certa qualidade, o que facilita e acelera o refinamento através do ILS. Deve-se salientar que todos os resultados obtidos foram validados e

Tabela 1. Resultados obtidos GRASP-ILS-Relaxado (resultados ~~riscados~~ são maiores ou iguais aos alcançados com a técnica) e valores médios obtidos sobre cada instância com a utilização da técnica com e sem *relaxação*.

INSTÂNCIA	Müller (USA)	Lu, Hao (França)	Atsuta et. Al (Japão)	Geiger (Alemanha)	Clark et. Al (Singapura)	GRASP ILS Relaxado	GRASP ILS (%)	GRASP ILS Relaxado (%)
comp01.ctt	~~5~~	~~5~~	~~5~~	~~5~~	~~10~~	5	5	5
comp02.ctt	51	55	50	~~111~~	~~111~~	110	173,9	160,5
comp03.ctt	84	71	82	~~128~~	~~119~~	105	250,9	141,9
comp04.ctt	37	43	35	~~72~~	~~72~~	45	65,4	56,8
comp05.ctt	330	309	312	~~410~~	~~426~~	394	1673,3	1612,6
comp06.ctt	48	53	69	100	~~130~~	101	137,6	130,9
comp07.ctt	20	28	42	~~57~~	~~110~~	78	122,0	111,9
comp08.ctt	41	49	40	~~77~~	~~83~~	53	68,3	67,5
comp09.ctt	109	105	110	~~150~~	~~139~~	119	141,7	133,0
comp10.ctt	16	21	27	~~71~~	~~85~~	68	114,5	94,2
comp11.ctt	~~0~~	~~0~~	~~0~~	~~0~~	~~3~~	0	0,0	0,0
comp12.ctt	333	343	351	~~442~~	408	409	686,7	463,2
comp13.ctt	66	73	68	~~98~~	~~113~~	87	99,3	95,4
comp14.ctt	59	57	59	~~90~~	~~84~~	70	92,1	85,2
comp15.ctt	84	71	82	~~128~~	~~119~~	112	149,1	141,5
comp16.ctt	34	39	40	~~81~~	~~84~~	81	112,1	93,6
comp17.ctt	83	91	102	~~124~~	~~152~~	120	157,3	137,9
comp18.ctt	83	69	68	~~116~~	~~110~~	86	118,5	102,9
comp19.ctt	62	65	75	~~107~~	~~111~~	102	135,1	143,1
comp20.ctt	27	47	61	88	~~144~~	99	153,2	151,1
comp21.ctt	103	106	123	~~174~~	~~174~~	140	187,4	173
RANK ITC	**1º**	**2º**	**3º**	**4º**	**5º**	**-**	-	-

cadastrados em `http://tabu.diegm.uniud.it/ctt/`, de acordo com as normas do ITC 2007.

5 Conclusões e trabalhos futuros

Este artigo apresenta a técnica GRASP-ILS-Relaxado para resolução do Problema de Programação de Cursos Universitários baseadas em Currículos. A técnica implementada foi testada e comparada com outras abordagens da literatura, sendo dessa forma avaliada a eficiência da abordagem realizada. Os resultados obtidos foram satisfatórios, tendo em vista as comparações realizadas, sendo a utilização do GRASP e também da técnica de relaxação fatores relevantes na eficiência do método ILS para o refinamento das soluções.

A definição dos parâmetros e também de algumas características das técnicas implementadas foram feitas tendo em vista o curto tempo de processamento e também a qualidade da solução final. Dessa forma, novas calibragens ainda não testadas (devido ao grande número de possibilidades) podem trazer melhorias significativas aos resultados obtidos.

A utilização de diferentes técnicas de exploração de vizinhanças, juntamente com novas técnicas meta-heurísticas, são considerados trabalhos futuros deste estudo, no intuito de aprimorar ainda mais a eficiência da técnica. Salienta-se que a utilização da técnica denominada *relaxação*, juntamente com as pertubações do ILS, foram um dos principais pontos da técnica implementada, sendo os resultados obtidos fortemente sensíveis a estes aspectos. Com a junção entre pertubações e movimentos de piora, busca-se uma forma mais eficaz de percorrer o espaço de busca, mesclando características de diversificação e intensificação.

Por fim, ressalta-se a contribuição do trabalho para a área, uma vez que os resulta-

dos obtidos servem de base para novos estudos. As principais contribuições desse trabalho consistem na descrição da aplicação da técnica híbrida GRASP-ILS para o problema e, também, na implementação da parte *relaxada* do ILS, a qual se trata de uma nova abordagem, que apresentou melhora significativa e poderá ser melhor explorada em trabalhos futuros. Os resultados estão disponíveis, incluindo os apresentados no presente artigo, no endereço `http://tabu.diegm.uniud.it/ctt/`.

Referências

Arroyo, J.E.C.; Nunes, G.V.P. e Kamke, E.H. (2009). Iterative local search heuristic for the single machine scheduling problem with sequence dependent setup times and due dates. *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*, volume 1, p. 505 –510, (2009).

Bai, R.; Burke, E. K.; Kendall, G. e Mccullum, B. (2008). A simulated annealing hyper-heuristic for university course timetabling problem extended abstract.

Barbosa, S. H. D.; Souza, S. R. e Sousa, A. M. (2010). Utilização de uma heurística híbrida para resolução do problema de alocação de horários em uma universidade. *XIII Encontro de Modelagem Computacional (EMC 2010)*. Nova Friburgo - RJ.

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

De Cesco, F.; Di Gaspero, L. e Schaerf, A. (2010). Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, p. 1–12.

Di Gaspero, L. e Schaerf, A. (2007). The second international timetabling competition (itc 2007): Curriculum-based course timetabling (track 3).

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133.

Lü, Z. e Hao, J. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*, v. 200, n. 1, p. 235 – 244.

Lourenço, H. R.; Martin, O. e Stützle, T. (2003). Iterated Local Search. Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, p. 321–353. Kluwer Academic Publishers.

Muller, T. (2008). Itc 2007 solver description: A hybrid approach. *In Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT*.

Qu, R. *Case Based Reasoning for Course Timetabling Problems*. PhD thesis, University of Nottingham, (2002).

Schaerf, A. (1999). A survey of automated timetabling. *Artif. Intell. Rev*, v. 13, n. 2, p. 87–127.

Souza, M. J. F. (2000). Programação de horários em escolas: uma aproximação por metaheurísticas. Tese de doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro.

Spindler, Morgana. (2010). Uma proposta de solução para problemas horário educacional utilizando busca dispersa e reconexão de caminhos. Master's thesis, Universidade do Vale do Rio Dos Sinos.

Talbi, E. (2009). *Metaheuristics : from design to implementation*, volume 10. John Wiley & Sons.

Willemen, R. J. (2002). School timetable construction algorithms and complexity. Master's thesis, Technische Universiteit Eindhoven.