Técnicas de Busca Local para o Problema da Programação de Horários Escolares

George H. G. Fonseca, Túlio A. M. Toffolo, Samuel S. Brito e Haroldo G. Santos

[1]Programa de Pós-graduação em Ciência da Computação
Universidade Federal de Ouro Preto (UFOP)
Caixa Postal 15.064 – 35.930-000 – Ouro Preto – MG – Brasil

george@decea.ufop.br, tulio@toffolo.com.br, {samuelsouza, haroldo}@iceb.ufop.br

Resumo. *O Problema da Programação de Horários Escolares é alvo de diversas pesquisas em Pesquisa Operacional e Inteligência Artificial devido a sua dificuldade de resolução e importância prática. Uma solução para esse problema consiste basicamente na alocação de aulas a horários e na alocação dos recursos para essas aulas. O presente trabalho considera a solução do problema proposto pela Third Interntional Timetabling Competition (ITC2011), a qual inclui um amplo conjunto de instâncias originadas de diversas instituições educacionais ao redor do mundo. Nós propomos os métodos de busca local Simulated Annealing e Iterated Local Search para o problema. Uma característica estrutural importante da nossa abordagem é o uso da plataforma KHE para gerar soluções iniciais, combinada com uma abordagem de busca multi-vizinhança. Os resultados obtidos foram animadores: dez de dezesete soluções factíveis foram encontradas e sete de vinte e uma melhores soluções conhecidas foram melhoradas ou atingidas no tempo limite estipulado pela ITC2011.*
Palavras Chave. *Busca Local. Problema da Programação de Horários Escolares. ITC2011.*
Metaheurísticas. Otimização Combinatória. PO na Educação.

Abstract. *The High School Timetabling Problem remains subject of many research in Operational Research and Artificial Intelligence fields because of its hardness to solve and practical importance. A solution for this problem basically consists in the schedule of lessons to timeslots and the assignment of resources for these lessons. This work considers the solution of the problem of the ongoing Third International Timetabling Competition (ITC2011), which includes a diverse set of instances from many educational institutions around the world. We proposed the local search methods Simulated Annealing and Iterated Local Search to the problem. One important structural feature of our approach is the use of the KHE engine to generate initial solutions combined with a multi-neighborhood search approach. The achieved results were encouraging: ten out of seventeen feasible solutions were found and seven out of twenty one all-time best solutions were improved or matched in the processing limit stipulated in the ITC2011.*
Keywords. *Local Search. High School Timetabling Problem. ITC2011.*
Metaheuristics. Combinatorial Optimization. OR in Education.

[1]Os autores agradecem à FAPEMIG (APQ-01779-10) e ao CNPq (480388/2010-5) pelo suporte ao desenvolvimento da presente pesquisa.

1 Introdução

O Problema da Programação de Horários Escolares foi proposto por Gotlieb (1963) e consiste em programar um horário escolar de modo que nenhuma turma ou professor participe de mais de uma aula no mesmo instante de tempo. Além disso, o horário deve atender a outras restrições especificadas a priori.

A programação automática de horários escolares tem sido alvo de diversas pesquisas nas áreas de Pesquisa Operacional e Inteligência Artificial. Em Santos e Souza (2007) são apresentadas algumas das razões para este interesse:

- Dificuldade de Resolução. Encontrar um quadro de horários que satisfaça todos os interesses envolvidos é uma tarefa difícil, ademais, frequentemente, a simples construção de um quadro de horários válido já é uma tarefa complicada;
- Importância Prática. A confecção de um bom quadro de horários pode melhorar a satisfação do corpo docente e permitir que a instituição de ensino seja mais eficiente na gestão de seus recursos, além do mais, a programação adequada das atividades letivas permite um melhor desempenho dos alunos;
- Importância Teórica. O problema apresentado neste trabalho é classificado como $\mathcal{NP}$-Difícil (Garey e Jonhson, 1979) e, progressos na solução de problemas desse tipo são um dos grandes objetivos das pesquisas correntes em Computação, Matemática e Pesquisa Operacional.

Métodos baseados em Programação Inteira já foram propostos para o problema (Santos *et al.*, 2012), porém são capazes de resolver apenas um subconjunto de instâncias do problema em tempo de processamento viável. Atualmente, as abordagens metaheurísticas são mais comumente aplicadas ao problema (Barbosa *et al.*, 2011) (Muller, 2011) (Lu e Hao, 2010).

Nesse sentido, o principal objetivo do presente trabalho é propor técnicas heurísticas baseadas em *Simulated Annealing* e *Iterated Local Search* para o modelo do Problema da Programação de Horários Escolares proposto pela *Third International Timetabling Competition* 2011. O referido modelo do problema é genérico e trata um conjunto amplo de restrições, tornando-se desafiador para a comunidade acadêmica.

O restante do trabalho está organizado da seguinte maneira: na Seção 2 apresentar-se-á o modelo do Problema da Programação de Horários Escolares proposto pela ITC2011. A Seção 3 apresenta a os métodos heurísticos abordados. A Seção 4 apresenta os experimentos computacionais e, por fim, a Seção 5 apresenta as considerações finais.

2 Problema da Programação de Horários Escolares

Sugerido pela ITC2011, o modelo de Programação de Horários Escolares (*High School Timetabling*) surgiu com o objetivo de fornecer um modelo genérico capaz de atender às diversas particularidades do Problema de Programação de Horários Escolares ao redor do mundo (Kingston, 2005) (Wright, 1996) (Nurmi, 2007) (Valourix, 2003) (Haan, 2007) (Santos e Souza, 2007). O modelo é dividido em três entidades principais:

2.1 Tempos e Recursos

A entidade tempo consiste de um espaço de tempo (timeslot) ou de um conjunto de espaços (grupo de espaços de tempo). Os recursos, por sua vez, são divididos em três categorias estudantes, professores e salas (Post *et al.*, 2010).

- Estudantes. Um grupo de estudantes atende a um determinado evento (aula). Restrições importantes para os estudantes são controlar o tempo ocioso e o número de aulas por dia;
- Professores. Um professor pode ser predefinido para atender um evento. Em alguns casos ele não é predefinido e deve ser alocado de acordo com sua qualificação e limites de carga de trabalho;
- Salas. A maioria dos eventos ocupa uma sala. Uma sala possui uma determinada capacidade e um conjunto de características.

2.2 Eventos

Um evento é a unidade básica de alocação, representando uma aula simples ou um conjunto de aulas (grupo de eventos). O termo classe é usado para designar um conjunto de estudantes que assistem ao mesmo conjunto de aulas. Outros tipos de eventos, como reuniões são permitidos pelo modelo (Post *et al.*, 2010).

Uma alocação de um evento a um espaço de tempo inicial é chamada encontro e uma alocação de um evento a um recurso é denominada tarefa. Um evento possui os seguintes atributos:

- A *duração*, que representa o número de espaços de tempo que têm que ser alocados ao evento;
- A *classe* relacionada ao evento;
- Os recursos que são predefinidos a atender ao evento (opcional);
- A *carga de trabalho*, que irá ser adicionada à carga de trabalho total dos recursos alocados ao evento(opcional);
- O *timeslot* em que deverá ser alocado (opcional).

2.3 Restrições

Post *et al.* (2010) agrupa as restrições em três grupos: restrições básicas do problema de agendamento, restrições para os eventos e restrições para os recursos. A função objetivo $f(.)$ é calculada em termos de violações a cada restrição penalizadas de acordo com o seu peso. Divide-se ainda as restrições entre fortes, cujo atendimento é mandatório e fracas, cujo atendimento é desejável mas não obrigatório. Cada instância pode definir se um dada restrição é forte ou fraca. Para mais detalhes, veja Post *et al.* (2010).

2.3.1 Restrições Básicas de Agendamento

1. *AssignTimeConstraint*. Alocar espaços de tempo a cada evento;
2. *AssignResourceConstraint*. Alocar os recursos a cada evento;
3. *PreferTimesConstraint*. Indica que determinado evento tem preferência por determinado(s) timeslot(s);
4. *PreferResourcesConstraint*. Indica que determinado evento tem prefereencia por determinado(s) recurso(s).

2.3.2 Restrições para os Eventos

1. *LinkEventsConstraint*. Agendar os grupos de eventos no mesmo tempo de início;

2. *SpreadEventsConstraint*. Agendar os eventos de cada grupo de eventos para cada grupo de tempo entre um número mínimo e um máximo de vezes. Essa restrição pode ser usada, por exemplo, para definir um limite diário de aulas;
3. *AvoidSplitAssignmentsConstraint*. Para cada grupo de eventos, alocar um determinado recurso a todos os eventos do grupo;
4. *DistributeSplitEventsConstraint*. Para cada evento, alocar entre um número mínimo e máximo de aulas consecutivas de uma dada duração.
5. *SplitEventsConstraint*. Limita o número de aulas não consecutivas em que um evento será agendado e sua duração.

2.3.3 Restrições para os Recursos

1. *AvoidClashesConstraint*. Agendar os recursos sem conflitos (ou seja, sem alocar o mesmo recurso a mais de um evento ao mesmo tempo);
2. *AvoidUnavailableTimesConstraint*. Evitar alocar os recursos nos horários em que estão indisponíveis;
3. *LimitWorkloadConstraint*. Agendar a carga de trabalho dos recursos entre um limite mínimo e máximo;
4. *LimitIdleTimes*. O número de horários ociosos em cada grupo de espaços de tempo deve estar entre um limite mínimo e máximo para cada recurso selecionado. Tipicamente, um grupo de espaços de tempo consiste dos *timeslots* de um dia da semana.
5. *LimitBusyTimesConstraint*. O número de horários ocupados em cada grupo de espaços de tempo deve estar entre um limite mínimo e máximo para cada recurso selecionado.
6. *ClusterBusyTimesConstraint*. O número de grupos de tempo com um *timeslot* alocado a um recurso deve figurar entre um limite mínimo e máximo. Tipicamente, o grupo de tempo são dias e, por exemplo, um professor requer no máximo três dias com aulas.

3 Técnicas Heurísticas Abordadas

3.1 Método Construtivo

O KHE (Kingston, 2012) é uma plataforma de manipulação de instâncias do problema abordado que conta ainda com um resolvedor, utilizado para a geração de soluções iniciais na presente abordagem. O resolvedor do KHE foi escolhido para gerar soluções iniciais uma vez que é capaz de encontrar soluções iniciais razoáveis de forma eficiente (veja a Tabela 3).

O resolvedor do KHE é baseado no conceito Programação de Horários Hierárquica (Kingston, 2006), onde alocações menores são unidas de modo a gerar blocos maiores de agendamento até que uma representação completa da solução seja desenvolvida.

A Programação de Horários Hierárquica é suportada pela estrutura de dados *Layer Tree* (Kingston, 2006), formada por nós que representam um número qualquer de alocações de espaços de tempo. Uma alocação pode figurar em no máximo um nó. Um *Layer* é um subconjunto de nós com a propriedade de que nenhum deles pode ser sobreposto no tempo. Comumente são agrupados em um *Layer* nós que compartilham recursos.

As restrições fortes do problema são modeladas para essa estrutura de dados e posteriormente resolve-se um problema de *Matching* para encontrar a alocação de horários/recursos. O *Matching* é feito ligando cada nó a um espaço de tempo ou recurso respeitando a propriedade da *Layer Tree*. Para mais detalhes, veja Kingston (2006) e Kingston (2012).

3.2 Estrutura de Vizinhança

A estrutura de vizinhança $N(.)$ é dividida em cinco categorias:

- EventSwap (*es*). Dois eventos e_1 e e_2 têm seus *timeslots* t_1 e t_2 trocados.
- EventMove (*em*). Um evento e_1 é movido de t_1 para um novo timeslot t_2 .
- EventBlockMove (*ebm*). Assim como es, troca o timeslot de dois eventos e_1 e e_2; porém, caso os eventos tenham duração diferente, e_1 é movido para o timeslot subsequente ao último timeslot ocupado por e_2.
- ResourceSwap (*rs*). Dois eventos e_1 e e_2 teem seus recursos r_1 e r_2 de um tipo específico trocados.
- ResourceMove (*rm*). Um evento e_1 passa a ter o recurso r_2 a ele aloccado em detrimento de r_1 anterior.

Caso a instância requeira a alocação de recursos (i.e. possua a restrição *AssignResourceConstraint*), o tipo é selecionado seguindo as seguintes probabilidades: $es = 0.2$, $em = 0.4$, $ebs = 0.1$, $rs = 0.2$ e $rm = 0.1$; caso contrário, as vizinhanças es e er são descartadas e as probabilidades são: $es = 0.5$, $em = 0.3$ e $ebs = 0.2$. Esses valores foram ajustados empiricamente.

3.3 Busca Local

3.3.1 Método Não Descendente Randômico

Este método parte de uma solução inicial qualquer e a cada passo analisa um vizinho qualquer da solução corrente e o aceita caso o mesmo seja melhor ou igual à solução. O método para após um número fixo $Iter_{max}$ de iterações sem melhora no valor da solução.

A possibilidade de realizar movimentos laterais (movimentos que não alteram o valor da solução) permite percorrer caminhos de descida que passam por regiões planas. Caso a busca chegue a uma região dessas, o método tem condições de mover-se nela e sair através de uma solução diferente daquela que a ela chegou. O Algoritmo 1 apresenta a implementação desenvolvida do Método Não Descendente Randômico, onde $f(.)$ representa a função objetivo e $N(.)$ a estrutura de vizinhança. Assumiu-se $Iter_{Max} = 1000$ para a realização dos experimentos.

3.3.2 *Iterated Local Search*

O método *Iterated Local Search* (ILS) é baseado na ideia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local. O tamanho da perturbação é um parâmetro chave do método. Caso esteja muito elevado, pode eliminar as características desejadas da solução ótimo local; caso esteja muito baixo pode não ser capaz de escapar do ótimo local corrente.

Algoritmo 1: Implementação desenvolvida do Método Não Descendente Randômico

Entrada: $f(.), N(.), s, Iter_{Max}, timeout$
Saída: Melhor solução s encontrada.
$Iter \leftarrow 0$;
enquanto $Iter < Iter_{max}$ **e** $elapsedTime < timeout$ **faça**
  $Iter \leftarrow Iter + 1$;
  Gere um vizinho aleatriório $s^{'} \in N(s)$;
  se $f(s^{'} \leq f(s))$ **então**
    $Iter \leftarrow 0$;
    $s \leftarrow s^{'}$;
retorna s;

O algoritmo ILS parte de uma busca local realizada na solução inicial s_0 e realiza perturbações de tamanho p_{size} sobre s sucedidas de busca local no intuito de escapar de ótimos locais. Uma perturbação é definida como o aceite incondicional de um vizinho qualquer de s. Assumiu-se o Método de Não Descendente Randômico ($naoDescRandomico$) como o procedimento de busca local.

A busca local origina uma solução $s^{'}$ que será aceita caso seja melhor que a melhor solução encontrada s^{*}. Caso aceita, o tamanho da perturbação p_{size} volta ao inicial p_0. Caso a iteração $Iter$ atinja um limite $Iter_{max}$ o tamanho da perturbação é incrementado. Caso ainda o tamanho da perturbação atinja um limite p_{max}, o mesmo volta ao tamanho inicial p_0. O Algoritmo 2 apresenta a implementação desenvolvida do ILS. Os parâmetros considerados nos experimentos computacionais foram $ILS_{max} = 10$, $p_0 = 2$ e $MaxIter_p = 5$.

3.3.3 *Simulated Annealing*

Proposta por Kirkpatrick *et al.* (1983), a metaheurística *Simulated Annealing* trata de um método probabilístico fundamentado em uma analogia à termodinâmica ao simular o resfriamento de um conjunto de átomos aquecidos. Esta técnica começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um loop que gera aleatoriamente, em cada iteração, um único vizinho s_0 da solução corrente s.

Seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s0) - f(s)$. O método aceita o movimento e a solução vizinha passa a ser a nova solução corrente se $\Delta < 0$. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde T é um parâmetro do método, chamado de temperatura e que regula a probabilidade de se aceitar soluções de pior custo. A temperatura T pode assumir, inicialmente, um valor elevado T_0. Após um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \times T_{k-1}$, sendo $0 < \alpha < 1$. Com esse procedimento, dá-se, no início uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de

Algoritmo 2: Implementação desenvolvida do ILS

Entrada: $f(.), N(.), ILS_{max}, p_0, MaxIter_p, s, timeout$
Saída: Melhor solução s encontrada.
$s \leftarrow naoDescRandomico(s)$; $s^* \leftarrow s$;
$p_{size} \leftarrow p_0$; $Iter \leftarrow 0$;
para $i \leftarrow 0$ *até* ILS_{max} **faça**
  se $elapsedTime \leq timeout$ **então**
    para $j \leftarrow 0$ *até* p_{size} **faça**
      $s \leftarrow s_p \in N(s)$;
    $s' \leftarrow naoDescRandomico(s)$;
    se $f(s' < f(s^*))$ **então**
      $s \leftarrow s'$; $s^* \leftarrow s'$;
      $Iter \leftarrow 0$; $p_{size} \leftarrow p_0$;
    senão
      $s \leftarrow s^*$;
      $Iter \leftarrow Iter + 1$;
    se $Iter = MaxIter$ **então**
      $p_{size} \leftarrow p_{size} + p_0$;
      se $p_{size} \geq p_{max}$ **então** $p_{size} \leftarrow p_0$;
$s \leftarrow s^*$;
retorna s;

piora (Glover *et al.*, 2003).

A implementação desenvolvida do Simulated Annealing será descrita no Algoritmo 3. Os parâmetros considerados nos experimentos computacionais foram $alpha = 0.5$ e $T_0 = 5$. O parâmetro $SAmax$ foi definido em função do número de eventos (nE) de cada instância ajustado por um multiplicador. Caso a solução inicial seja factível (i.e. não tenha violação de nenhuma restrição forte), $SAmax = nE \times 10$; caso contrário, $SAmax = nE \times 100$.

4 Experimentos Computacionais

Todos os experimentos foram realizados em um notebook Intel ® i5 2.4 Ghz com 4GB de RAM sobre o sistema operacional Ubuntu 11.10. A linguagem de programação usada no desenvolvimento do software foi C++ compilada pelo GCC 4.6.1. Os tempos de processamento foram ajustados de acordo com o *benckmark* disponibilizado pela *Third International Timetabling Competition*. O *timeout* foi definido em 1000 segundos, conforme requerido pela segunda etapa da *ITC* 2011. Todos os resultados foram validados pelo validador HSEval `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi`.

4.1 Caracterização das Instâncias

O conjunto de instâncias disponibilizado pela *ITC* 2011 `http://www.utwente.nl/ctit/hstt/archives/XHSTT-2012` foi originado de diversos países e varia de instâncias pequenas a instâncias desafiadoras para a comunidade científica. A Tabela 1 apresenta as principais características das instâncias.

Algoritmo 3: Implementação desenvolvida do *Simulated Annealing*

Entrada: $f(.), N(.), \alpha, SAmax, T_0, s, timeout$
Saída: Melhor solução s encontrada.
$s^* \leftarrow s$;
$IterT \leftarrow 0$;
$T \leftarrow T_0$;
enquanto $T > 0$ **e** $elapsedTime < timeout$ **faça**
  enquanto $IterT < SAmax$ **faça**
    $IterT \leftarrow IterT + 1$;
    Gere um vizinho aleatório $s' \in N(s)$;
    $\Delta = f(s') - f(s))$;
    se $\Delta < 0$ **então**
      $s \leftarrow s'$;
      se $f(s') < f(s^*)$ **então** $s^* \leftarrow s'$;
    senão
      Tome $x \in [0, 1]$;
      se $x < e^{-\Delta/T}$ **então** $s \leftarrow s'$;
  $T \leftarrow \alpha \times T$;
  $IterT \leftarrow 0$;
$s \leftarrow s^*$;
retorna s;

4.2 Resultados Obtidos

A Tabela 2 apresenta os resultados obtidos pelas técnicas de busca local propostas. Os resultados são expressos pelo par x/y, onde x contém a medida de infactibilidade da solução e y a medida de qualidade. As colunas Simulated Annealing e Iterated Local Search contêm os resultados obtidos por cada metaheurística. Os resultados foram coletados considerando cinco execuções de cada método. Assim, $f(s^*)$ apresenta a melhor solução obtida, $f(\overline{s})$ apresenta a média dos resultados, σ o desvio padrão e t, o tempo médio (em segundos) também seguido do desvio padrão.

A Tabela 3 apresenta um comparativo da melhor solução obtida por cada metaheurística proposta com relação à melhor solução conhecida para cada instância da ITC2011. A primeira fase da ITC2011 objetiva à obtenção das melhores soluções possíveis para as instâncias, sem limites de tempo nem de tecnologias. Dessa forma, experimentos com os métodos aplicados à melhor solução conhecida foram realizados. Os mesmos foram representados pelas colunas SA_{free} e ILS_{free}.

4.3 Discussão dos Resultados

O diferencial da abordagem proposta para outras abordagens heurísticas foi o uso do resolvedor do KHE para gerar soluções iniciais combinado com uma busca local multi-vizinhança. O resolvedor do KHE foi capaz de encontrar rapidamente soluções boas e a estrutura de vizinhança proposta foi capaz de explorar consistentemente o espaço de soluções e alcançar melhoras significantes na solução inicial.

Tabela 1. Características das instâncias da *ITC* 2011

Instância	Timeslots	Professores	Salas	Classes	Eventos	Duração
AustraliaBGHS98	40	56	45	30	387	1564
AustraliaSAHS96	60	43	36	20	296	1876
AustraliaTES99	30	37	26	13	308	806
BrazilInstance1	25	8		3	21	75
BrazilInstance4	25	23		12	127	300
BrazilInstance5	25	31		13	119	325
BrazilInstance6	25	30		14	140	350
BrazilInstance7	25	33		20	205	500
EnglandStPaul	27	68	67	67	1227	1227
FinlandArtificialSchool	20	22	12	13	169	200
FinlandCollege	40	46	34	31	387	854
FinlandHighSchool	35	18	13	10	172	297
SecondarySchool	35	25	25	14	280	306
GreeceHighSchool1	35	29		66	372	372
GreecePatras3rdHS2010	35	29		84	178	340
GreecePreveza3rdHS2008	35	29		68	164	340
ItalyInstance1	36	13		3	42	133
NetherlandsGEPRO	44	132	80	44	2675	2675
NetherlandsKottenpark2003	38	75	41	18	1156	1203
NetherlandsKottenpark2005	37	78	42	26	1235	1272
SouthAfricaLewitt2009	148	19	2	16	185	838

Pela Tabela 2 pode-se observar que ambos os métodos propostos foram capazes de resolver satisfatoriamente até as instâncias grandes no pequeno tempo limite fixado. De pequenas a médias instâncias a abordagem foi muito rápida. O desvio padrão tambem foi baixo, evidenciando a consistência dos métodos em termos de tempo de processamento.

Ao observar o custo médio de infactibilidade ($f(\overline{s})$) apresentado na Tabela 2, pode-se concluir que a metaheurística ILS foi capaz de encontrar soluções melhores que a metaheurística Simulated Annealing. Porém, o algoritmo Simulated Annealing foi, em média, aproximadamente $1,5\times$ mais rápido que o ILS.

Até mesmo encontrar soluções factíveis para essas instâncias é uma tarefa difícil. As mesmas comumente definem a maioria das restrições como fortes. Dessa forma, a ITC2011 não espera que um resolvedor seja capaz de encontrar todas as soluções factíveis e o uso do par $infactibiliadde/qualidade$ para avaliar soluções é recomendado. Nossa abordagem foi capaz de alcançar dez de dezesete [1] soluções factíveis no limite de tempo estipulado.

Pela Tabela 3, pode-se observar que a nossa abordagem foi capaz de alcançar ou melhorar a melhor solução conhecida em sete de vinte e uma instâncias. Vale ressaltar que não houve limite de tempo nem de tecnologia para a obtenção das melhores soluções conhecidas originais e a abordagem proposta, mesmo sobre o tempo limite estipulado, foi capaz de melhorar algumas soluções e se aproximar das outras.

Assumindo a melhor solução conhecida como solução inicial, as metaheurísticas propostas foram capazes de melhorar quinze de dezeseis [2] soluções. Alguns desses resul-

[1] As quatro instâncias restantes não foram consideradas pois não há garantia de que uma solução factível existe.
[2] As cinco instâncias restantes já possuem uma solução de custo zero conhecida.

Tabela 2. Resultados Obtidos

Instância	Simulated Annealing				Iterated Local Search			
	$f(s^*)$	$f(\overline{s})$	σ	$\overline{t_s}$	$f(s^*)$	$f(\overline{s})$	σ	$\overline{t_s}$
AustraliaBGHS98	11.0 / 375.0	11.0 / 381.6	±0.0 / ±5.7	147.5±3.5	9.0 / 389.0	10.6 369.4	±0.9 ±43.8	125.8±4.9
AustraliaSAHS96	11.0 / 11.0	11.0 / 11.8	±0.0/ ±1.3	167.3±1.6	10.0 / 12.0	10.8 / 13.4	±0.4 / ± 1.5	154.9±8.4
AustraliaTES99	6.0 / 148.0	6.0 / 148.0	±0.0 / ±0.0	72.1±0.4	5.0 / 148.0	5.8 / 148.0	±0,45 / ±0.0	46.5 ± 1.7
BrazilInstance1	0.0 / 36.0	0.0 / 39.6	±0.0 / ±6.5	0.7±0.0	0.0 / 24.0	0.0 / 28.2	±0.0 / ±2.7	3.6±0.4
BrazilInstance4	12.0 / 150.0	12.0 / 159.0	±0.0 / ±12.0	4.8±0.4	12.0 / 129.0	12.0 / 141.0	±0.0 / ±10.17	28.4±1.7
BrazilInstance5	6.0 / 149.0	7.0 / 167.4	±1.0 / ±20.7	4.7±1.2	5.0 / 135.0	6.6 / 156.4	± 1.5 / ±24.5	26.8±1.3
BrazilInstance6	5.0 / 258.0	5.6 / 262.8	±0.5 / ±10.3	5.3±0.0	4.0 / 246.0	4.6 / 258.0	±0.9 / ± 18.1	31.5±1.2
BrazilInstance7	13.0 / 285.0	14.6 / 281.4	±0.9 / ±11.5	8.1±0.4	13.0 / 264.0	13.4 / 280.2	±0.9 / ±15.4	48.1±1.3
EnglandStPaul	0.0 / 15208.0	0.0 / 17916.8	±0.0 / ±1991.0	592.1±3.7	0.0 / 22550.0	0.00 / 24173.2	±0.0 / ±1747.8	540.5±62.7
FinlandArtificialSchool	10.0 / 8.0	13.0 / 8.0	±1.7 / ±2.5	53.7±2.3	10.0 / 8.0	10.8 / 10.0	±0.8 / ±3.4	32.5±0.6
FinlandCollege	4.0 / 109.0	5.6 / 114.2	±1.1 / ±15.1	157.3±2.4	3.0 / 119.0	5.0 / 113.2	±1.2 / ±21.6	102.3±4.7
FinlandHighSchool	0.0 / 74.0	1.4 / 89.4	±1.1 / ±17.8	59.2±1.8	0.0 / 67.0	0.4 / 68.2	±0.5 / ±15.0	32.2±1.1
FinlandSecondarySchool	3.0 / 114.0	3.8 / 121.2	±0.8 / ±28.4	164.5±1.9	0.0 / 148.0	2.2 / 117.6	±1.5 / ±27.3	121.6±7.8
GreeceHighSchool1	0.0 / 0.0	0.0 / 0.0	±0.0 / ±0.0	73.5±1.9	0.0 / 0.0	0,0 / 0.0	±0.0 / ±0.0	135.2±1.7
GreecePatras3rdHS2010	0.0 / 80.0	0.6 / 91.2	±0.9 / ±19.1	97.9±2.4	0.0 / 94.0	0.6 / 111.2	±0.5 / ±19,1	81.1±2.9
GreecePreveza3rdHS2008	0.0 / 189.0	1.4 / 165.4	±1.1 / ±20.5	87.4±0.8	1.0 / 143.0	1.8 / 138.6	±0.4 / ±35.9	70.4±4.6
ItalyInstance1	0.0 / 34.0	0.0 / 40.4	±0.0 / ±4.5	2.7±0.0	0.0 / 23.0	0.0 / 29.2	±0.0 / ±6.9	14.8±1.5
NetherlandsGEPRO	2.0 / 6095.0	2.0 / 18511.0	±0.0 / ±3641.4	1000.0±0.0	2.0 / 49008.0	2.0 / 69118.0	±0.0 / ±2597.2	1000.0±0.0
NetherlandsKottenpark2003	0.0 / 5481.0	0.2 / 7224.0	±0.4 / ±1185.4	1000.0±0.0	0.0 / 8555.0	0.6 / 8691.2	±0.5 / ±442.2	1000.0±0.0
NetherlandsKottenpark2005	10.0 / 2783.0	11.6 / 3045.8	±0.9 / ±270.3	1000.0±0.0	11.0 / 5863.0	12.8 / 6187.4	±1.1 / ±815.0	1000.0±0.0
SouthAfricaLewitt2009	0.0 / 20.0	0.4 / 28.4	±0.5 / ±8.9	51.6±0.9	2.0 / 44.0	4.0 / 35.6	±1.4 / ±10.7	48.8±3.1
Média	4.4 / 1505.1	5.1 / 2324.2	±0.5 / ±346.3	226.2±1.2	4.1 / 4976.2	4.9 / 5247.0	±0.6 / ±160.9	332.0±1.2

Tabela 3. Comparativo entre os resultados das metaheurísticas desenvolvidas e a melhor solução conhecida

Instância	ITC2011	KHE Solver	$SA_{f(s^*)}$	SA_{free}	$ILS_{f(s^*)}$	ILS_{free}
AustraliaBGHS98	7 / 433	11 / 415	11 / 375	**4 / 367**	9 / 389	**3 / 405**
AustraliaSAHS96	23 / 44	11 / 15	**11 / 11**	**22 / 44**	**10 / 12**	**23 / 24**
AustraliaTES99	26 / 134	6 / 418	**6 / 148**	**25 / 124**	**5 / 148**	**25 / 124**
BrazilInstance1	0 / 24	0 / 81	0 / 36	**0 / 15**	**0 / 24**	**0 / 15**
BrazilInstance4	0 / 112	17 / 225	12 / 150	**0 / 103**	12 / 141	**0 / 103**
BrazilInstance5	0 / 225	12 / 258	6 / 149	**0 / 210**	5 / 135	**0 / 198**
BrazilInstance6	0 / 209	11 / 339	5 / 258	**0 / 173**	4 / 246	**0 / 156**
BrazilInstance7	0 / 330	20 / 429	13 / 285	**0 / 318**	13 / 264	**0 / 294**
EnglandStPaul	0 / 18444	0 / 49756	**0 / 15208**	**0 / 11732**	0 / 22550	**0 / 12338**
FinlandArtificialSchool	0 / 0	16 / 18	10 / 8	0 / 0	10 / 8	0 / 0
FinlandCollege	0 / 0	20 / 747	4 / 109	0 / 0	3 / 119	0 / 0
FinlandHighSchool	0 / 1	7 / 446	0 / 74	0 / 1	0 / 67	0 / 1
FinlandSecondarySchool	0 / 106	35 / 301	3 / 114	0 / 106	0 / 148	**0 / 102**
GreeceHighSchool1	0 / 0	0 / 0	**0 / 0**	0 / 0	**0 / 0**	0 / 0
GreecePatras3rdHS2010	0 / 0	8 / 399	0 / 80	0 / 0	0 / 94	0 / 0
GreecePreveza3rdHS2008	0 / 0	6 / 684	0 / 189	0 / 0	1 / 143	0 / 0
ItalyInstance1	0 / 28	0 / 323	0 / 34	0 / 28	**0 / 23**	**0 / 26**
NetherlandsGEPRO	1 / 566	2 / 69118	2 / 6095	**1 / 549**	2 / 49008	**1 / 382**
NetherlandsKottenpark2003	0 / 1410	1 / 72413	0 / 5481	**0 / 1189**	0 / 8555	**0 / 1239**
NetherlandsKottenpark2005	0 / 1078	18 / 22284	10 / 2783	**0 / 963**	11 / 5863	**0 / 1058**
SouthAfricaLewitt2009	0 / 58	361 / 0	**0 / 20**	**0 / 42**	2 / 44	**0 / 44**

tados foram os melhores alcançados na primeira fase da ITC2011.

5 Considerações Finais

O principal objetivo do presente trabalho foi alcançado, uma vez que foram apresentadas abordagens eficientes baseadas em busca local para o Problema da Programação de Horários Escolares proposto pela *Third International Timetabling Competition.*

As implementações propostas das metaheurísticas foram capazes de melhorar ou alcançar a melhor solução conhecida para sete de vinte e uma instâncias. Assumindo a melhor solução como solução inicial, nossos métodos foram capazes de melhorar quinze de dezeseis soluções. Alguns desses resultados foram os melhores da primeira fase da ITC2011 em andamento. Esses resultados são animadores devido à dificuldade do problema e sua importância prática.

Nós acreditamos que ainda há melhorias a serem feitas em nossa abordagem. Alguns possíveis trabalhos futuros são (1) desenvolver tipos de vizinhanca adicionais capazes de realizar alterações maiores em uma solução, como *Kempe Moves* e *Ejection Chains* (Lu e Hao, 2010); (2) desenvolver e realizar um estudo computacional de outras metaheurísticas, como Algoritmos Genéticos e *Variable Neighborhood Search*; e (3) desenvolver uma heurística de Programação Inteira Mista para o problema.

Referências

Barbosa, S. H. and Souza, S. R. (2011). Resolução do problema de programação de cursos universitários baseada em currículos via uma meta-heurística híbrida grasp-ils-relaxado. In *XLIII Simpósio Brasileiro de Pesquisa Operacional, Ubatuba, Proceedings of XLIII SBPO, Ubatuba : SOBRAPO*, pages 1 : 2827–2882.

de Haan, P., Landman, R., Post, G., and Ruizenaar, H. (2007). A case study for timetabling in a dutch secondary school. In *Lecture notes in computer science: VI Practice and theory of automated timetabling. Berlin : Springer*, pages 3867 : 267–279.

Garey, M. R. and Jonhson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, USA.

Glover, F. and Kochenberger, G. (2003). *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Kluwer Academic Publishers.

Gotlieb, C. C. (1963). The construction of class-teacher time-tables. In *Proc. IFIP Congress, Munich, North Holland Pub. Co.*, pages 73–77.

Kingston, J. H. (2005). A tiling algorithm for high school timetabling. In *Lecture notes in computer science: V Practice and theory of automated timetabling. Berlin: Springer*, pages 3616 : 208–225.

Kingston, J. H. (2006). Hierarchical timetable construction. In *Problems, Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling*.

Kingston, J. H. (2012). A software library for school timetabling. Disponível em http://sydney.edu.au/engineering/it/ jeff/khe/, Acessado em Abril de 2012.

Kirkpatrick, S., Gellat, D. C., and Vecchi, M. P. (1983). Otimization by simulated annealing. In *Science*, pages 202, 671–680.

Lú, Z. and Hao, J.-K. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200(1):235–244.

Muller, T. (2009). Itc2007 solver description: a hybrid approach. *Annals OR*, 172(1):429–446.

Nurmi, K. and Kyngas, J. (2007). A framework for school timetabling problem. In *Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications, Paris*, pages 386–393.

Post, G., Ahmadi, S., Daskalaki, S., Kingston, J. H., Kyngas, J., Nurmi, C., and Ranson, D. (2010). An xml format for benchmarks in high school timetabling. In *Annals of Operations Research DOI 10.1007/s10479-010-0699-9.*, pages 3867 : 267–279.

Santos, H. G. and Souza, M. J. F. (2007). Programação de horários em instituições educacionais: formulações e algoritmos. In *XXXIX Simpósio Brasileiro de Pesquisa Operacional, Fortaleza, Anais do XXXIX SBPO, Rio de Janeiro : SOBRAPO*, pages 1 : 2827–2882.

Santos, H. G., Uchoa, E., Ochi, L. S., and Maculan, N. (2012). Strong bounds with cut and column generation for class-teacher timetabling. *Annals OR*, 194(1):399–412.

Valourix, C. and Housos, E. (2003). Constraint programming approach for school timetabling. In *Computers & Operations Research*, pages 30 : 1555–1572.

Wright, M. (1996). School timetabling using heuristic search. In *Journal of Operational Research Society*, pages 47 : 347–357.