Uma abordagem de busca local utilizando GVNS para solução de problemas de programação de horários em escolas

Ulisses Rezende Teixeira

Sérgio Ricardo de Souza

ulisses.rezende@gmail.com

sergio@dppg.cefetmg.br

Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

Belo Horizonte, MG, 30510-000, Brasil

Marcone Jamilson Freitas Souza

marcone@iceb.ufop.br

Universidade Federal de Ouro Preto (UFOP)

Ouro Preto, MG, 35400-000, Brazil

Abstract. *O objetivo deste trabalho é apresentar uma alternativa eficiente de solução para problemas de programação de horários em escolas utilizando uma abordagem de busca local com o método General Variable Neighborhood Search (GVNS). Para refinamento das soluções, três diferentes estratégias de busca local foram testadas, cada qual dando origem a uma versão diferente do algoritmo. Todas as versões usam a biblioteca KHE, especializada para implementação de problemas de programação de horários. O espaço de soluções é explorado por meio dos seguintes tipos de movimento: troca de eventos, troca de recursos, troca de bloco de recursos, mudança de horário de recurso, mudança de horário de evento e cadeia Kempe. O algoritmo foi testado em problemas teste da Competição Internacional de Programação de Horários ocorrida em 2011 e seus resultados foram comparados com aqueles gerados pelo algoritmo Goal Solver, vencedor da competição.*

Keywords: *Timetabling, School Timetabling, GVNS, Metaheurísticas, VND*

1 INTRODUÇÃO

A montagem de um quadro de horários é uma tarefa comum a todas as instituições de ensino a cada início de período letivo. Esta tarefa, de forma bastante simplificada, é realizada combinando horários, turmas e professores de forma a atender ao maior número possível de restrições. As restrições podem ser de vários tipos, tais como: indisponibilidade de professores em alguns horários, redução ao máximo de janelas de horários de professores, restrição a aulas da mesma disciplina de forma sequencial, um mesmo professor só pode estar vinculado a uma única turma no mesmo horário.

Janelas de horários é um termo bastante comum, utilizado tanto por professores quanto por responsáveis por montar horários, e designa um conjunto de horários ociosos de um determinado professor entre duas aulas de um mesmo turno. Se um professor é alocado para ministrar aulas no primeiro e no terceiro horários então há uma janela no segundo horário. Há também a mesma situação relacionada às turmas, quando uma turma fica com um horário ocioso entre duas aulas. Espera-se sempre minimizar a ocorrência desses eventos; entretanto, sua ocorrência não inviabiliza uma determinada solução.

As restrições costumam ser divididas em dois grupos (Santos et al., 2005):

- Restrições fortes: são aqueles que devem ser atendidas obrigatoriamente, sob pena de a solução se tornar inviável. Como exemplo, podemos citar a alocação de um professor a uma única turma em um mesmo horário. Se um professor for alocado, por exemplo, a duas turmas no mesmo horário, a solução é inviável.
- Restrições fracas: são aquelas que devem ser atendidas sempre que possível, e o não atendimento não inviabiliza a solução. Como exemplo, podemos citar a existência de janelas no horário de um determinado professor. É esperado reduzir ao máximo a ocorrência de janelas, mas sua existência não inviabiliza a solução.

Não há pesquisas nem dados oficiais a respeito do assunto, mas acredita-se que a grande maioria das instituições de ensino, em especial no Brasil, realizem a atividade de montagem de horários de forma completamente manual. Encontrar uma única solução, conforme Bardadym (1995), pode levar horas ou até dias dependendo do tamanho da instituição e da quantidade de turmas envolvidas. Além disso, normalmente a solução utilizada é a primeira encontrada, dado o grande esforço de avaliar uma solução melhor.

Segundo Schaerf (1999), os problemas de programação de horários podem ser classificados em três classes diferentes:

- *School Timetabling*: restringe a alocação de professores a horários de uma determinada turma. Usado em especial para escolas de ensino básico (níveis fundamental e médio), em que a matrícula é realizada na forma seriada em blocos de disciplinas de um mesmo período letivo;
- *Course timetabling*: orientado à alocação de disciplinas, sendo cada disciplina parte de um curso e as turmas formadas por alunos de vários cursos. Trata-se de um modelo utilizado tipicamente por instituições de ensino superior, que têm a matrícula na forma de créditos em disciplinas;
- *Examination timetabling*: utilizado no caso de alocação de exames (avaliações) a serem aplicados a turmas de disciplinas.

Apesar de amplamente utilizada, esta classificação não abrange totalmente todos os problemas, conforme Souza (2000), já que existem variações do problema que não se enquadram de maneira exata nestas três categorias.

Desde o trabalho precursor de Gotlieb (1963), várias técnicas têm sido utilizadas para resolver os problemas de programação de horários. Os estudos no tema motivaram a criação de uma conferência internacional, conhecida como *Practice and Theory on Automated Timetabling* - PATAT, bem como a organização de competições específicas, como a *International Timetabling Competition* - ITC.

Todo o interesse se baseia em três pontos principais (Santos and Souza, 2007; Schaerf, 1999):

- Dificuldade de encontrar uma solução: devido à grande quantidade de restrições, encontrar uma solução viável se torna uma tarefa bastante árdua e que pode levar dias de trabalho manual, de acordo com a quantidade de recursos (turmas, professores, horários) envolvidos.
- Importância prática: ter um quadro de horários é uma necessidade básica de todas as instituições de ensino. Um bom quadro de horários, por outro lado, pode impactar a vida de uma grande quantidade de indivíduos, sejam eles alunos ou professores e pode, inclusive, impactar na eficiência de professores e no desempenho de alunos.
- Importância teórica: é demonstrado, de acordo com Even et al. (1975), que problemas de programação de horários pertencem à classe de problemas NP-Difíceis, sendo objeto de constantes estudos para se obter métodos que produzem boas soluções em tempos polinominais, independente de sua dimensão.

A classe de problemas conhecida como *High School Timetabling* ou Problema Classe-Professor, tem como características básicas, segundo Souza (2000), a repetição da programação de horários semanalmente, com cada turma apresentando um único conjunto de disciplinas para todos os seus alunos e suas aulas distribuídas em um mesmo turno (manhã, tarde ou noite).

O conceito de turma neste cenário é definido como sendo um conjunto de alunos que possuem uma mesma grade curricular e assistem suas aulas em uma mesma sala de aula, sendo que a alocação das salas de aula não faz parte do problema, já que é associada diretamente à turma. Em Schaerf (1999) são descritas algumas variações deste problema, como a ocorrência de aulas envolvendo mais de uma turma, disciplinas ministradas por mais de um professor e alocação de salas de aula específicas de acordo com a disciplina.

No presente trabalho, trata-se o problema de programação de horários em escolas considerado na *International Timetabling Competition* de 2011 (ITC 2011). Para resolvê-lo, propõe-se três versões de um algoritmo baseado na metaheurística GVNS (*General Variable Neighborhood Search*) usando a biblioteca KHE (*Kingston High School timetabling engine*).

O restante do artigo está organizado da seguinte forma: Na Seção 2 é apresentado o padrão XHSTT e os benefícios trazidos pela criação e padronização de problemas testes a partir da ITC 2011. Na Seção 3 são detalhadas as versões do algoritmo, bem como a biblioteca KHE. Na Seção 4 os resultados obtidos são comparados com aqueles do algoritmo *Goal Solver*, vencedor da ITC 2011. Por fim, são apresentadas as conclusões deste trabalho, bem como indicados trabalhos futuros.

2 FORMATO XHSTT E O ITC 2011

Como dito anteriormente, o interesse da comunidade científica acerca de problemas de *timetabling* levou à criação da Conferência Internacional PATAT, organizada a cada dois anos para apoiar e estimular a comunidade internacional de pesquisadores e praticantes em todos os aspectos relacionados à geração de quadros de horários por métodos computacionais.

Esta comunidade foi a responsável por lançar um projeto de criação de um formato padrão para representar os problemas do tipo Classe-Professor e também a criação de um repositório com um conjunto unificado de problemas teste para *benchmarking* de estudos. Este projeto resultou no formato denominado XHSTT, acrônimo para o nome em inglês *XML archive for High School Timetabling*, descrito por Post et al. (2014).

Como o próprio nome diz, o XHSTT é um formato baseado em XML (*eXtensive Markup Language*) e estabelece estruturas específicas para tratar recursos, horários e suas respectivas restrições. Já o repositório público criado no mesmo projeto foi composto inicialmente por problemas teste com situações reais e outros simulados coletados de diversos estudos realizados ao redor do mundo, como por exemplo Inglaterra (Wright, 1996), Finlândia (Nurmi and Kyngas, 2007), Grécia (Valouxis and Housos, 2003), Holanda (De Haan et al., 2006) e Brasil (Souza, 2000).

O modelo XHSTT é dividido em três entidades básicas, sendo:

- Tempo e recursos: a entidade tempo é composta por horários e os recursos são subdivididos em três subcategorias: turmas, professores e salas.
- Eventos: Um evento é a unidade básica de associação representando uma aula. Um evento pode ser uma tarefa, quando está vinculado a um recurso ou um encontro quando está vinculado a um horário.
- Restrições: caracterizam por orientar a distribuição dos recursos nos eventos. Podem ser definidas como sendo fortes ou fracas, de acordo com os critérios esperados para que uma determinada solução seja viável ou não. Além disso, são subdivididas em três subcategorias: restrições básicas de agendamento, restrições de eventos e restrições de recursos.

As restrições básicas de agendamento são as seguintes:

- Atribuir horário: atribui um horário a cada evento;
- Atribuir recurso: atribui um recurso a cada evento;
- Definir preferência de horários: indica que alguns eventos possuem preferência por alguns horários;
- Definir preferência de Recursos: indica que alguns eventos possuem preferência por alguns recursos.

As restrições de eventos são as seguintes:

- Vincular eventos: atribui um conjunto de eventos a um mesmo horário de início;
- Espalhar eventos: distribui eventos de forma uniforme nos horários disponíveis;
- Evitar quebra de atribuições: para cada evento, atribui um dado recurso a todos os seus horários;

- Distribuir quebra de eventos: para cada evento, distribui entre um número mínimo e um número máximo de horários de uma dada duração
- Quebrar eventos: impõe limites no número de encontros não consecutivos criados por um evento e sua duração.

As restrições de recursos são as seguintes:

- Evitar conflitos: evita que se atribua um mesmo recurso a mais de um evento por horário;
- Evitar indisponibilidade: estabelece que certos recursos são indisponíveis em certos horários;
- Limitar carga de trabalho: limita a carga total de trabalho de um recurso;
- Limitar horários vagos: o número de horários vagos em cada grupo de horários deve ficar limitado entre um número mínimo e máximo para cada recurso;
- Limitar horários ocupados: o número de horários vagos em cada grupo de horários deve ficar limitado entre um número mínimo e máximo para cada recurso;
- Reduzir horários espalhados de um recurso: força que a alocação de atividades de um dado recurso seja agrupada em certos grupos de horários.

Com este novo formato definido, os organizadores da conferência PATAT lançaram então a terceira edição da competição internacional de problemas de *timetabling* - ITC, dedicada a problemas do modelo *High School timetabling*, no ano de 2011. Esta foi a última e mais recente competição da área desde então, sendo as duas anteriores organizadas em 2002 e 2007, cada uma com seus temas específicos e distintos.

O ITC 2011 foi dividido em três etapas, sendo elas:

- Fase 1: os próprios competidores eram responsáveis pela geração das melhores soluções para os problemas teste públicos sem restrições de tempo e de recursos computacionais;
- Fase 2: os organizadores realizavam a execução de cada algoritmo participante nas mesmas condições com o tempo de 1000 segundos usando um problema teste específico não divulgado previamente;
- Fase 3: os próprios competidores realizaram a geração de soluções em um conjunto oculto de problemas teste e, tal como na fase 1, não foram definidas restrições de tempo e tecnologias. Somente os cinco melhores da fase 2 chegaram a esta fase.

3 SOLUÇÃO PROPOSTA

A proposta deste trabalho é a implementação de um algoritmo computacional baseado em uma metaheurística que permita:

- A leitura dos problemas teste utilizadas pelo ITC2011 no padrão XHSTT
- A geração de uma solução para cada problema teste utilizado no ITC2011 através da adoção de um método heurístico

De posse das soluções geradas, o objetivo é realizar uma comparação dos resultados encontrados com aqueles obtidos pelo algoritmo vencedor da competição (*Goal Solver*), avaliando a qualidade das soluções encontradas e a viabilidade ou não do uso desta abordagem para solução de problemas deste tipo.

3.1 BIBLIOTECA KHE

Em 2006, Kingston (2006) criou uma biblioteca que denominou de KHE (*Kingston High School Timetabling Engine*). Esta biblioteca foi construída exclusivamente para problemas *timetabling*, facilitando e otimizando, assim, o manejo de problemas teste e soluções de problemas deste tipo. Totalmente integrada com o padrão XHSTT, os grandes diferenciais do uso desta biblioteca são as estruturas de dados disponíveis e a possibilidade de utilizar sua função de geração de solução inicial, denominada *KheGeneralSolve*. Esta rotina gera uma solução inicial de forma bastante rápida, mesmo para problemas teste grandes e complexos, apesar de muitas vezes estas soluções não serem viáveis ou mesmo terem um custo inicial elevado. Esta biblioteca está disponível na Internet e pode ser usada livremente para estudos e pesquisas na área. Seu criador fornece também um serviço de avaliação de soluções, denominado de HsEval, disponível em http://www.it.usyd.edu.au/ jeff/hseval.cgi (Kingston, 2014).

3.2 MÉTODO DE SOLUÇÃO PROPOSTO

Para resolver o problema de programação de horários em estudo, propõe-se um algoritmo baseado na metaheurística GVNS (*General Variable Neighborhood Search*). Esta metaheurística foi proposta por Mladenović et al. (2008) e é uma extensão da metaheurística VNS (*Variable Neighborhood Search*), proposta por Mladenović and Hansen (1997).

O funcionamento do algoritmo GVNS é bastante similar ao do algoritmo VNS padrão, sendo a única diferença a etapa de refinamento da solução. O método VNS utiliza um método de busca local com uma única vizinhança. O GVNS, por sua vez, aplica o método VND (*Variable Neighborhood Descent*), em que são aplicadas várias estruturas de vizinhança retornando, ao final, um ótimo local com relação a todas essas estruturas.

O algoritmo GVNS gera uma nova solução a partir da solução atual e executa um procedimento de busca local. Ao concluí-la, é verificado se a solução encontrada é melhor que a solução atual. Se for, a solução atual é atualizada e repete-se o processo enquanto houver tempo disponível. Caso contrário, a solução encontrada é descartada e repete-se o processo de geração de uma nova solução.

As soluções são geradas aplicando-se o movimento da Cadeia de Kempe. Esse movimento foi proposto por Johnson et al. (1991) para o problema de coloração de vértices e baseia-se no conceito de que determinadas mudanças podem gerar soluções inviáveis criando conflitos e para eliminá-los é necessária uma cadeia de outros movimentos. Essas modificações sequenciais realizadas em uma determinada solução é denominada de Cadeia de Kempe. A Figura 1 ilustra um exemplo de mudanças geradas pela aplicação deste movimento.

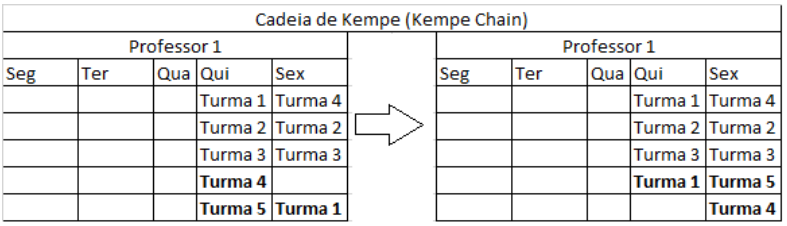


Figura 1: Movimento Cadeia de Kempe

A cada iteração em que a solução não é melhorada, o algoritmo incrementa o número de vezes que o movimento da Cadeia de Kempe é executado até o limite máximo de dez execuções (variável $Kempe_max$). Este valor foi definido de forma empírica balanceando o ganho com a maximização do espaço de busca com o esforço realizado. Quando uma solução de melhora é encontrada, o algoritmo volta a executar o movimento apenas uma vez. Este comportamento tem como objetivo buscar soluções melhores escapando de ótimos locais.

O GVNS é apresentado pelo Algoritmo 1.

Algoritmo 1: ALGORITMO GVNS

Entrada: Solução inicial s_0, Tempo máximo de execução MaxTime, Número máximo de movimentos da Cadeia Kempe $Kempe_{\max}$

Saída: Melhor solução s encontrada.

```
1  início
2      s ← Busca local (s_0);
3      k_atual ← 1;
4      enquanto tempo ≤ MaxTime faça
5          s' ← vizinho de s usando movimento Cadeia de Kempe;
6          para k = 1; k ≤ k_atual faça
7              s' ← vizinho de s' usando movimento Cadeia de Kempe;
8          fim
9          s' ← Busca local (s');
10         se f(s') < f(s) então
11             s ← s';
12             k_atual ← 1;
13         fim
14         senão se k_atual ≤ Kempe_max então
15             k_atual ← k_atual + 1;
16         fim
17     fim
18 fim
19 retorna s
```

Neste trabalho foram utilizadas três versões diferentes do algoritmo de busca local ou algoritmo de descida (linhas 2 e 9 do Algoritmo 1), descritas e apresentadas a seguir.

- VND (*Variable Neighborhood Descent*): Nesta busca local, a ordem das vizinhanças é determinística, previamente determinada conforme mostrado no Algoritmo 2.

Algoritmo 2: ALGORITMO VND

Entrada: Solução inicial s_0, Total de vizinhanças do refinamento r, Número máximo de vizinhos *MaxViz*

Saída: Melhor solução s encontrada.

```
1  início
2      k ← 1; /*Primeira vizinhança*/
3      s ← s_0;
4      enquanto k ≤ r faça
5          melhorou ← Falso; iter ← 1;
6          enquanto existir vizinho s' na k-ésima vizinhança e iter ≤ MaxViz faça
7              iter ← iter+1;
8              s' ← próximo vizinho de s na k-ésima vizinhança;
9              se f(s') ≤ f(s) então
10                 s ← s';
11                 melhorou ← Verdadeiro
12             fim
13         fim
14         se melhorou então
15             k ← 1;
16         fim
17         senão
18             k ← k + 1;
19         fim
20     fim
21 fim
22 retorna s
```

- rVND (*random Variable Neighborhood Descent*): Nesta busca local, a ordem das vizinhanças é aleatória a cada iteração. Proposta em Souza et al. (2010), ela tem como vantagem a não necessidade de determinar a ordem das vizinhanças e representa, assim, um parâmetro a menos do algoritmo. O Algoritmo 3 mostra seu funcionamento.

Algoritmo 3: ALGORITMO RVND

Entrada: Solução inicial s_0, Conjunto de N vizinhanças, Número máximo de vizinhos *MaxViz*

Saída: Melhor solução s encontrada.

início
  $s \leftarrow s_0$;
  $N \leftarrow \{N^1, N^2, ..., N^r\}$ /* Conj. de vizinhanças*/
  $\mathcal{N} \leftarrow \{\mathcal{N}^1, \mathcal{N}^2, ..., \mathcal{N}^r\}$ /*Conj. de vizinhanças randomizado */
  para *cada* $\mathcal{N}^i \in \mathcal{N}$ **faça**
    $iter \leftarrow 1$;
    enquanto *existir vizinho* s' *na vizinhança* $\mathcal{N}^i$ *e iter* $\leq$ *MaxViz* **faça**
      $iter \leftarrow$ iter+1;
      $s' \leftarrow$ próximo vizinho de s na vizinhança $\mathcal{N}^i$;
      se $f(s') <= f(s)$ **então**
        $s \leftarrow s'$;
      fim
    fim
    se *encontrou solução melhor* **então**
      $i \leftarrow 1$;
    fim
  fim
fim
retorna s

- BLVA (Busca Local em Vizinhança Aleatória): Busca local que consiste em aplicar uma sequência completamente randômica de vizinhanças, sendo escolhida uma vizinhança a cada iteração do método VND conforme mostrado no Algoritmo 4.

Algoritmo 4: ALGORITMO BLVA

Entrada: Solução inicial s_0, Conjunto de vizinhanças N, Número máximo de vizinhos *MaxViz*.

Saída: Melhor solução s encontrada.

início
  $s \leftarrow s_0$;
  $N \leftarrow \{N^1, N^2, ..., N^r\}$ /* Conj. de vizinhanças*/
  $iter \leftarrow 1$;
  enquanto *iter* $\leq$ *MaxViz* **faça**
    $iter \leftarrow$ iter+1;
    $s' \leftarrow$ gera vizinho de s na n-ésima vizinhança randômica onde $n \in N$;
    se $f(s') < f(s)$ **então**
      $s \leftarrow s'$;
    fim
  fim
fim
retorna s

Nos três métodos de busca local foi acrescentado um limite máximo de iterações para busca de vizinhos: MaxViz. Este parâmetro foi definido com o valor 1.000.000. Este limitador para o número de iterações evita que o método consuma todo o tempo disponível buscando soluções em uma única vizinhança, principalmente em problemas teste muito grandes, podendo assim reduzir sua efetividade. Ele foi obtido de forma empírica, a partir da avaliação do comportamento do algoritmo nos diversos problemas teste.

Todos os métodos de refinamento utilizam as mesmas cinco estruturas de vizinhança, $N^1, N^2, ..., N^5$, as quais são apresentadas a seguir, nesta ordem:

- Troca de eventos (*Event Swap*): este movimento seleciona duas aulas de uma turma e realiza a troca de horários entre elas.

Troca de eventos (meet swap)

Turma 1				
Seg	Ter	Qua	Qui	Sex
Mat	Mat	Fís	Mat	Bio
Port	Mat	Fís	Port	Bio
Port	Hist	Qui	Hist	Hist
Geo	Hist	Qui	Qui	Mat
Geo	Fís	Mat	Fís	Mat

Turma 1				
Seg	Ter	Qua	Qui	Sex
Port	Mat	Fís	Mat	Bio
Mat	Mat	Fís	Port	Bio
Port	Hist	Qui	Hist	Hist
Geo	Hist	Qui	Qui	Mat
Geo	Fís	Mat	Fís	Mat

Figura 2: Movimento de troca de eventos

- Mudança de eventos (*Event move*): este movimento seleciona uma aula e move ela para um outro horário que esteja disponível.

Mudança de evento (event move)

Professor 1				
Seg	Ter	Qua	Qui	Sex
			Turma 1	Turma 4
			Turma 2	Turma 2
			Turma 3	Turma 3
			Turma 4	
			Turma 5	Turma 1

Professor 1				
Seg	Ter	Qua	Qui	Sex
			Turma 1	Turma 4
			Turma 2	Turma 2
			Turma 3	Turma 3
			Turma 4	**Turma 5**
				Turma 1

Figura 3: Movimento de mudança de eventos

- Troca de eventos em bloco (*Event Block Swap*): este movimento é similar ao movimento de troca de eventos, onde são selecionadas duas aulas de uma turma e seus horários são trocados. A diferença neste movimento ocorre se para uma das aulas selecionados existir outra aula em um horário adjacente à ela. Se isso correr, a troca envolve não somente a aula escolhida, mas sim suas adjacências.

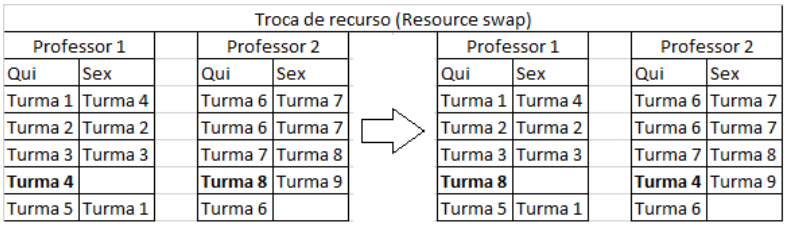
Troca de eventos em bloco (meet block swap)

Turma 1				
Seg	Ter	Qua	Qui	Sex
Mat	Mat	Fís	Mat	Bio
Port	Mat	Fís	Port	Bio
Port	Hist	Qui	Hist	Hist
Geo	Hist	Qui	Qui	Mat
Geo	Fís	Mat	Fís	Mat

Turma 1				
Seg	Ter	Qua	Qui	Sex
Port	Mat	Fís	Mat	Bio
Port	Mat	Fís	Port	Bio
Mat	Hist	Qui	Hist	Hist
Geo	Hist	Qui	Qui	Mat
Geo	Fís	Mat	Fís	Mat

Figura 4: Movimento de troca de eventos em bloco

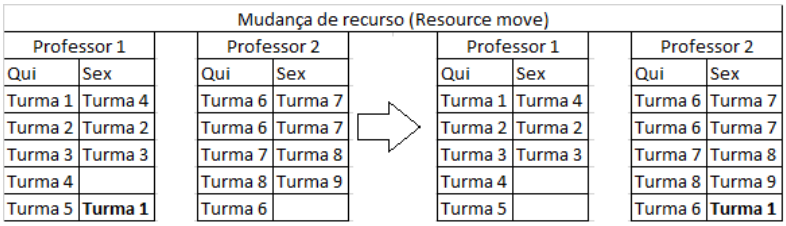
- Troca de recursos (*Move Time*): este movimento seleciona duas aulas e realiza a troca de recursos entre elas.



Troca de recurso (Resource swap)							
Professor 1		Professor 2		Professor 1		Professor 2	
Qui	Sex	Qui	Sex	Qui	Sex	Qui	Sex
Turma 1	Turma 4	Turma 6	Turma 7	Turma 1	Turma 4	Turma 6	Turma 7
Turma 2	Turma 2	Turma 6	Turma 7	Turma 2	Turma 2	Turma 6	Turma 7
Turma 3	Turma 3	Turma 7	Turma 8	Turma 3	Turma 3	Turma 7	Turma 8
Turma 4		**Turma 8**	Turma 9	**Turma 8**		**Turma 4**	Turma 9
Turma 5	Turma 1	Turma 6		Turma 5	Turma 1	Turma 6	

Figura 5: Movimento de troca de recursos.

- Mudança de recursos (*Change Time*): este movimento seleciona uma aula e faz a troca de recurso dela para um outro recurso que esteja disponível no horário.



Mudança de recurso (Resource move)							
Professor 1		Professor 2		Professor 1		Professor 2	
Qui	Sex	Qui	Sex	Qui	Sex	Qui	Sex
Turma 1	Turma 4	Turma 6	Turma 7	Turma 1	Turma 4	Turma 6	Turma 7
Turma 2	Turma 2	Turma 6	Turma 7	Turma 2	Turma 2	Turma 6	Turma 7
Turma 3	Turma 3	Turma 7	Turma 8	Turma 3	Turma 3	Turma 7	Turma 8
Turma 4		Turma 8	Turma 9	Turma 4		Turma 8	Turma 9
Turma 5	**Turma 1**	Turma 6		Turma 5		Turma 6	**Turma 1**

Figura 6: Movimento de mudança de recursos.

4 RESULTADOS

O algoritmo GVNS e suas três versões de busca local foram implementados em C++ utilizando-se a IDE Code::Blocks.

Todos os testes foram realizados em um notebook Intel Core i5, com 4 GB de RAM rodando o sistema operacional Windows 10.

Assim como na competição ITC 2011, foram utilizadas como base para a validação do algoritmo GVNS os mesmos vinte e um problemas teste públicos. Esses problemas teste retratam situações de programação de horários em escolas de vários países e possuem características bastante diversificadas, sendo alguns simples e outros com alto número de turmas e professores, tornando-os altamente complexos.

As características dos problemas teste são apresentadas na Tabela 1. Nesta tabela, a primeira coluna indica o nome do problema teste, a segunda coluna mostra o números de horários, a terceira coluna mostra o número de professores envolvidos, a quarta coluna o de salas, a quinta coluna o de turmas e, finalmente, na última coluna é registrada a quantidade de aulas do referido problema teste.

Tabela 1: Problemas-teste da ITC 2011

Problema-teste	#Horários	#Profs	#Salas	#Turmas	#Aulas
AustraliaBGHS98	40	56	45	30	1564
AustraliaSAHS96	60	43	36	20	1876
AustraliaTES99	30	37	26	13	806
BrazilInstance1	25	8	-	3	75
BrazilInstance4	25	23	-	12	300
BrazilInstance5	25	31	-	13	325
BrazilInstance6	25	30	-	14	350
BrazilInstance7	25	33	-	20	500
EnglandStPaul	27	68	67	67	1227
FinlandArtificialSchool	20	22	12	13	200
FinlandCollege	40	46	34	31	854
FinlandHighSchool	35	18	13	10	297
SecondarySchool	35	25	25	14	306
GreeceHighSchool1	35	29	-	66	372
GreecePatras3rdHS2010	35	29	-	84	340
GreecePreveza3rdHS2008	35	29	-	68	340
ItalyInstance1	36	29	-	3	133
NetherlandsGEPRO	44	132	80	44	2675
NetherlandsKottenpark2003	38	75	41	18	1203
NetherlandsKottenpark2005	37	78	42	26	1272
SouthAfricaLewitt2009	148	19	2	16	838

Assim como na primeira fase da ITC 2011, os testes foram realizados sem restrições computacionais nem de ambiente, mas para fins de comparação, foi utilizado o mesmo tempo de 1000s para cada problema teste, mesmo valor utilizado pelo algoritmo *Goal Solver*, vencedor da competição. Além do tempo de execução, foram utilizadas também, as mesmas soluções iniciais de cada problema teste. Essas soluções foram divulgadas pela organização do evento e disponibilizadas juntamente com cada problema teste em seus respectivos arquivos XHSTT. Os três problemas teste oriundos da Austrália (AustraliaBGHS98, AustraliaSAHS96 e AustraliaTES99) apresentam uma característica em comum: a solução inicial divulgada pelo ITC é de qualidade inferior à solução inicial obtida pela execução do método disponível na biblioteca KHE. Para esses três problemas teste em particular, a solução inicial utilizada no método

GVNS (e também pelo algoritmo *Goal Solver*) foi a melhor solução, desconsiderando a solução disponibilizada pela organização da competição.

Dos vinte e um problemas teste apresentados no evento, cinco deles já apresentavam solução inicial ótima, portanto não fazia sentido avaliá-las, são eles:

- FinlandArtificialSchool
- FinlandCollege
- GreeceHighSchool1
- GreecePatras3rdHS2010
- GreecePreveza3dHS2008

Dos dezesseis problemas teste restantes, todos foram avaliados considerando as condições descritas anteriormente e os resultados são apresentados na Tabela 2. Nesta tabela são mostrados:

- O valor da solução inicial divulgado pela organização do evento (os valores entre parênteses mostram o valor obtido pela biblioteca KHE, quando este for melhor).
- Os resultados retornados pelo algoritmo *Goal Solver*, executados na mesma máquina utilizada pelo GVNS
- Os resultados do algoritmo GVNS utilizando-se o algoritmo VND
- Os resultados do algoritmo GVNS utilizando-se o algoritmo rVND
- Os resultados do algoritmo GVNS utilizando-se o algoritmo BLVA

Os valores apresentados entre parênteses mostram o *gap* percentual entre os algoritmos GVNS e *Goal Solver* e é calculado pela Equação (1):

$$gap_i = 100 \times (f_i^\star(\text{GVNS}) - f_i^\star(\text{Goal}))/f_i^\star(\text{Goal}) \tag{1}$$

sendo $f_i^\star$(GVNS) o resultado alcançado pelo algoritmo GVNS, e respectivo método de busca local, no problema teste i e $f_i^\star$(Goal) o alcançado pelo *Goal Solver*.

Quando os valores de avaliação das restrições fortes são diferentes, o *gap* é calculado apenas com base nesses valores. Nos demais casos são usados somente os valores das restrições fracas. Resultados negativos de *gap* indicam que o algoritmo GVNS foi melhor que o algoritmo *Goal Solver*.

Os valores destacados em negrito indicam qual o método de descida, dentre os três implementados como métodos de busca local para o algoritmo GVNS, obteve o melhor resultado.

Tabela 2: Comparação de resultados entre os algoritmos GVNS e GOAL Solver

Problema-teste	ITC 2011	*Goal Solver* (SA + ILS)	GVNS VND	GVNS rVND	GVNS BLVA
AustraliaBGHS98	7/433 (6/450)	6/450	4/370 (-33,33)	**4/364 (-33,33)**	6/448 (-0,44)
AustraliaSAHS96	23/44 (17/550)	14/50	12/51 (-14,29)	**12/47 (-14,29)**	14/49 (-2,00)
AustraliaTES99	26/134 (7/163)	7/161	**7/151 (-6,21)**	**7/151 (-6,21)**	7/161 (0,00)
BrazilInstance1	0/24	0/12	**0/11 (-8,33)**	0/12 (0,00)	**0/11 (-8,33)**
BrazilInstance4	0/112	0/91	**0/94 (3,30)**	**0/94 (3,30)**	**0/94 (3,30)**
BrazilInstance5	0/225	0/164	**0/155 (-5,49)**	0/161 (-1,83)	0/158 (-3,66)
BrazilInstance6	0/209	0/149	0/148 (-0,67)	0/148 (-0,67)	**0/137 (-8,05)**
BrazilInstance7	0/330	0/264	0/249 (-5,68)	0/252 (-4,55)	**0/240 (-5,30)**
EnglandStPaul	0/18.444	0/18.092	**0/12.542 (-30,68)**	0/14.564 (-19,60)	0/18.418 (1,80)
FinlandHighSchool	0/1	0/1	**0/1 (0,00)**	**0/1 (0,00)**	**0/1 (0,00)**
FinlandSecondarySchool	0/106	0/86 (1,16)	0/87 (1,16)	0/87	**0/86 (0,00)**
ItalyInstance1	0/28	0/19	**0/18 (-5,26)**	0/19 (0,00)	**0/18 (-5,26)**
NetherlandsGEPRO	1/566	1/566	**1/434 (-23,32)**	1/464 (-18,02)	1/441 (-22,08)
NetherlandsKottenpark2003	0/1.410	0/1.409	**0/1.216 (-13,70)**	0/1.248 (-11,43)	0/1.343 (-4,68)
NetherlandsKottenpark2005	0/1.078	0/1.078	**0/881 (-18,27)**	0/972 (-9,83)	0/1.016 (-5,75)
SouthAfricaLewitt2009	0/58	0/22 (9,09)	0/24	0/24 (9,09)	**0/20 (-9,09)**

Dos dezesseis problemas teste avaliados, em somente dois (Finland High school e Finland Secondary school) nenhum dos três algoritmos conseguiu resultados melhores que o algoritmo *Goal Solver*.

Dos quatorze problemas teste restantes, o algoritmo VND alcançou resultados melhores em dez. O algoritmo rVND obteve bons resultados em dois problemas teste e igualou seus resultados a outro método em outros três problemas teste e o algoritmo BLVA alcançou o melhor resultado em dois problemas teste e teve o melhor resultado equivalente à outro método em outros dois.

A Figura 7 mostra, para cada versão do algoritmo, o número de problemas teste em que foi obtido o melhor resultado. Em caso de empate, foram contabilizados todos os métodos que obtiveram o melhor valor. Nota-se pelos resultados obtidos que há um equilíbrio entre o método VND e o método BLVA, com uma pequena vantagem para o método VND. Nota-se, também, que o procedimento de randomizar a relação de vizinhanças implementado no método rVND não trouxe muitos benefícios para a busca local visto que a utilização deste método não conseguiu produzir efetivamente melhores soluções em nenhum problema teste.

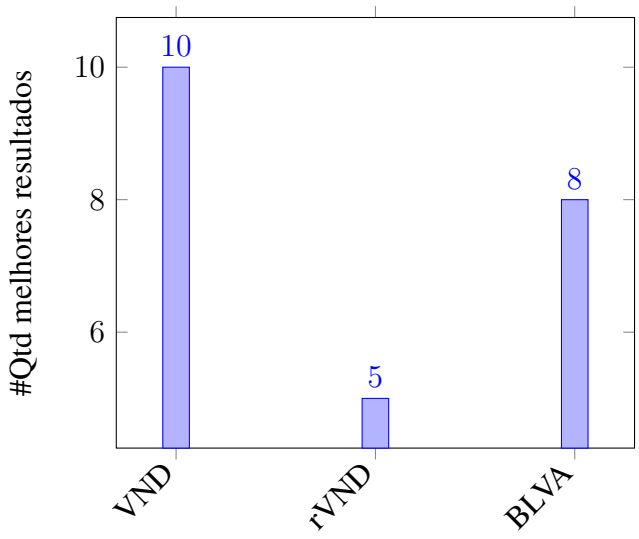


Figura 7: Comparação entre as diferentes versões de busca local

5 CONCLUSÕES

O presente artigo propôs uma abordagem de solução de problemas de programação de horários em escolas utilizando como base o algoritmo GVNS. Foram implementados três tipos diferentes de busca local: a primeira baseada no métdo VND, a segunda no método rVND e a terceira no método BLVA.

Dentre os três métodos de busca local testados no algoritmo GVNS, observou-se a superioridade do método VND, o qual, ao ser usado pelo GVNS, obteve resultados melhores que o vencedor da competição internacional ITC 2011, *Goal Solver*, em 12 de 16 problemas teste que não apresentavam solução inicial ótima. O método BLVA aplicado ao GVNS alcançou resultados melhores que o *Goal Solver* em onze problemas teste, enquanto que com o método rVND, o GVNS alcançou resultados melhores que o *Goal Solver* em dez problemas teste.

Esta nova abordagem mostrou-se bastante aderente a este tipo de problema, visto que o algoritmo GVNS, com qualquer das buscas locais implementadas, alcançou resultados melhores que o algoritmo vencedor da competição.

O algoritmo GVNS implementado apresenta como vantagem a simplicidade e o fato de ter poucos parâmetros, independentemente do método de descida utilizado. Essa é uma vantagem sobre o algoritmo *Goal Solver*, que tem oito parâmetros, sendo quatro de cada um dos dois métodos usados na sua hibridização.

Dentre os métodos de refinamento, o método VND foi o que teve melhor desempenho comparado com os demais, pois foi o método que conseguiu o maior número de melhores resultados.

Como possíveis trabalhos futuros, sugere-se avaliar o comportamento de cada vizinhança em cada problema teste específico e implementar uma priorização no uso das vizinhanças que geram mais incrementos na solução. Além disso, sugere-se estender os testes para outros problemas teste, em especial, os utilizados em trabalhos mais recentes, como por exemplo, o de Saviniec and Constantino (2017).

Agradecimentos

Os autores agradecem ao CNPq, FAPEMIG, CEFET-MG e UFOP pelo apoio ao desenvolvimento deste trabalho. Agradecem, também, aos autores do algoritmo *Goal Solver* pela disponibilização do código do algoritmo.

Referências Bibliográficas

Bardadym, V. A. (1995). Computer-aided school and university timetabling: The new wave. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 22–45. Springer.

De Haan, P., Landman, R., Post, G., and Ruizenaar, H. (2006). A case study for timetabling in a dutch secondary school. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 267–279. Springer.

Even, S., Itai, A., and Shamir, A. (1975). On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE.

Gotlieb, C. (1963). The construction of class-teacher timetables. *In Proceedings of IFIP Congress*, pages 73–77.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1991). Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations research*, 39(3):378–406.

Kingston, J. (2014). http://www.it.usyd.edu.au/ jeff/hseval.cgi.

Kingston, J. H. (2006). Hierarchical timetable construction. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 294–307. Springer.

Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., and Čangalović, M. (2008). General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191(3):753–770.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

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., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., et al. (2014). Xhstt: an xml archive for high school timetabling problems in different countries. *Annals of Operations Research*, 218(1):295–301.

Santos, H. G., Ochi, L. S., and Souza, M. J. (2005). A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *Journal of Experimental Algorithmics (JEA)*, 10:2–9.

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

Saviniec, L. and Constantino, A. A. (2017). Effective local search algorithms for high school timetabling problems. *Applied Soft Computing*.

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

Souza, M. J. (2000). *ProgramaÃ§Ã£o de horÃ¡rios em escolas: uma aproximaÃ§Ã£o por metaheurÃsticas*. Tese de doutorado, Universidade Federal do Rio de Janeiro.

Souza, M. J., Coelho, I. M., Ribas, S., Santos, H. G., and Merschmann, L. H. d. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051.

Valouxis, C. and Housos, E. (2003). Constraint programming approach for school timetabling. *Computers & Operations Research*, 30(10):1555–1572.

Wright, M. (1996). School timetabling using heuristic search. *Journal of the Operational Research Society*, 47(3):347–357.