UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LANDIR SAVINIEC

Operadores de vizinhança eficientes para algoritmos de busca local aplicados ao problema de horários em escolas

Maringá
2013

LANDIR SAVINIEC

Operadores de vizinhança eficientes para algoritmos de busca local aplicados ao problema de horários em escolas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ademir Aparecido Constantino

Maringá
2013

Dados Internacionais de Catalogação na Publicação (CIP)
(Biblioteca Central - UEM, Maringá, PR, Brasil)

Saviniec, Landir
S267o Operadores de vizinhança eficientes para algoritmos de busca local aplicados ao problema de horários em escolas / Landir Saviniec. -- Maringá, 2013.
99 f. : il. color., figs., tabs.

Orientador: Prof. Dr. Ademir Aparecido Constantino.
Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2013.

1. Operadores de vizinhança. 2. Estruturas de vizinhança. 3. Neighborhood operators. 4.ILS - Horário em escolas - Problemas. 5. VNS - Horário em escolas - Problemas. 6. High school timetabling problem. 7. Algoritmos de busca local. I. Constantino, Ademir Aparecido, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22.ed. 006.31
AMMA-00653

FOLHA DE APROVAÇÃO

LANDIR SAVINIEC

Operadores de vizinhança eficientes para algoritmos de busca local aplicados ao problema de horários em escolas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Ademir Aparecido Constantino
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Wesley Romão
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Haroldo Gambini Santos
Universidade Federal de Ouro Preto – DECOM/UFOP

Aprovada em: 21 de fevereiro de 2013.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá

AGRADECIMENTOS

Agradeço a Deus por ter me abençoado nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

A minha família pelo carinho, apoio e incentivo.

A todos os meus professores de ensino primário, fundamental, médio, graduação e mestrado.

Aos colegas de turma pelo companheirismo.

Ao meu orientador professor Dr. Ademir Aparecido Constantino pelo apoio, comentários e sugestões no desenvolvimento deste projeto.

Aos demais membros da banca examinadora, professores Dr. Wesley Romão e Dr. Haroldo Gambini Santos, pela avaliação e contribuições dadas ao trabalho.

A amigo professor Ms. Amauri Jersi Ceolim pelos comentários sobre o texto.

E a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro concedido a este trabalho.

Operadores de vizinhança eficientes para algoritmos de busca local aplicados ao problema de horários em escolas

RESUMO

Esta dissertação aborda o problema de horários em escolas. Este é um problema combinatório clássico que possui muitas variantes. Ele é NP-Completo e geralmente é resolvido por métodos heurísticos. O trabalho apresentado propõe algoritmos de busca local para resolver uma variante do problema originado de treze escolas públicas de ensino médio brasileiras. No trabalho é realizado um estudo comparativo entre dois operadores de vizinhança propostos para o problema e um operador da literatura. Os algoritmos propostos implementam estes operadores e são baseados nas metaherísticas *ILS* e *VNS*. Experimentos computacionais foram realizados aplicando estes algoritmos para resolver instâncias de uma base de dados extraída de casos reais destas escolas. Os resultados obtidos mostraram que algoritmos de busca local baseada nos operadores propostos são mais eficientes que algoritmos de busca local baseada no operador da literatura. Além disso, foi observado que os algoritmos *ILS* e *VNS* usando combinações de heurísticas de busca local baseadas nos operadores propostos produziram os melhores resultados e que estes resultados são satisfatórios.

Palavras-chave: Operadores de vizinhança. Algoritmos de busca local. Problema de horários em escolas.

Efficient neighborhood operators for local search algorithms applied to the high school timetabling problem

ABSTRACT

This paper addresses the high school timetabling problem. This is a classical problem and has many combinatorial variations. It is NP-Complete and is usually tackled using heuristic methods. In this work we propose local search algorithms to solve a variant of the problem faced on thirteen brazilian public high schools. The work performs a comparative study among two proposed neighborhood operators and an operator from literature. The proposed algorithms are based on metaheristics ILS and VNS and incorporates these operators. We have performed computational experiments by applying these algorithms to solve real instances of a database we have taken from these schools. The results have shown that local search algorithms using our operators are more efficient than algorithms using the literature operator. Furthermore, we have observed that ILS and VNS algorithms using combinations of local search heuristics based in our operators have produced the best results and these results are satisfactory to the problem.

Keywords: Neighborhood operators. Local search algorithms. High school timetabling problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CP: *Constraint Programming*
CPU: *Central Processing Unit*
DM: Operador *Double Move*
ILS: *Iterated Local Search*
IMLS: *Iterated Multi Local Search*
MT: Operador *Matching*
PAS: Problema de Alocação de Salas
PD: Problema de Designação
PEE: Problema de Escalonamento de Enfermeiros
PEHs: Problemas de Escalonamento de Horários
PEH-IE: Problema de Escalonamento de Horários em Instituições de Ensino
PHC: Problema de Horários de Cursos
PHE: Problema de Horários em Escolas
PHEX: Problema de Horários de Exames
POC: Problema de Otimização Combinatória
RNA: *Randomized Nonascendent Method*
TQ: Operador Torque
VNS: *Variable Neighborhood Search*

SUMÁRIO

1

Introdução

Problemas de Escalonamento de Horários (*PEHs*) são problemas de otimização combinatória de difícil solução do ponto de vista da teoria da complexidade computacional. Em outras palavras, para a maioria dos casos reais envolvendo muitas restrições, não se conhece qualquer algoritmo de complexidade polinomial capaz de encontrar a solução ótima do problema. Como algoritmos capazes de fornecer a solução ótima para estes casos, possuem complexidade exponencial, sua aplicabilidade para resolver instâncias com dimensões consideráveis torna-se impraticável devido ao alto custo computacional. Por esta razão, justifica-se o uso de métodos heurísticos para tratar *PEHs* em muitas aplicações reais. Tais métodos não garantem a solução ótima do problema, embora são capazes de fornecer boas soluções em tempo computacional aceitável.

Uma classe de problemas de escalonamento de horários muito conhecida na literatura se encontra em instituições de ensino. Denominaremos esta classe de *PEH-IE*. Neste contexto o problema de horários é classificado em três categorias: Problema de Horários em Escolas, Problema de Horários de Cursos e Problema de Horários de Exames. Eles diferem entre si pelo tipo de instituição envolvida (Schaerf, 1999b).

Problema de Horários em Escolas - PHE: consiste na escala semanal de um conjunto de aulas de pares ordenados (professor, turma) sobre um quadro de horários com um número fixo de períodos de mesma duração. Em algumas instituições, devido à escassez de salas de aula, este recurso também deve ser levado em consideração no processo de escalonamento, aumentando a complexidade do problema.

Problema de Horários de Cursos - PHC: consiste na escala semanal de aulas de cursos universitários que compartilham estudantes em comum. Em alguns casos, este problema também leva em consideração a escala de salas de aula.

Problema de Horários de Exames - PHEX: consiste na escala dos exames de um conjunto de cursos universitários a serem realizados durante um período de tempo, geralmente uma semana. Como nos problemas anteriores, em alguns casos a alocação de salas de aula como parte integrante do processo de escalonamento também se faz necessária.

Em geral, o problema de escalonamento de horários em instituições de ensino é resolvido de forma manual na maioria das instituições, e na prática as soluções obtidas não são de boa qualidade. Além disso, um quadro de horários feito manualmente, pode levar vários dias para ser realizado e devido as frequentes modificações nos recursos envolvidos, por exemplo, professores que deixam a escola por algum motivo, o quadro de horários tem que ser refeito, gerando sérios inconvenientes nas atividades diárias da instituição. Por estas razões a construção manual de horários torna-se uma tarefa onerosa.

Devido ao exposto acima, maior atenção tem sido dedicada pela comunidade acadêmica a respeito da automação computacional do problema de horários voltado a instituições de ensino nos últimos quarenta anos. Em especial, nas últimas duas décadas uma grande quantidade de trabalhos experimentais atacando o problema via algoritmos heurísticos tem sido introduzido na literatura.

Por outro lado, as pesquisas voltadas ao problema de horários em instituições de ensino têm se desenvolvido em maior volume sobre o *PHC* e o *PHEX* que sobre o *PHE*. A pesquisa no campo do *PHE*, durante as últimas décadas, tem se desenvolvida de forma fragmentada (Pillay, 2010a; Post et al., 2008, 2010). Como descrito por estes autores, a principal barreira nesta área se deve a estudos de casos isolados, não abordando problemas em comum. Como consequência, até 2008 nenhuma base de dados com um número significativo de instâncias de teste era compartilhada entre os pesquisadores da área para comparar diferentes métodos. Esta questão é melhor detalhada na seção 2.4.

Dentre as técnicas heurísticas comumente aplicadas na resolução do *PHE*, destacam-se as metaheurísticas:

- **Algoritmos Genéticos** (Abramson et al., 1991; Beligiannis et al., 2008; Caldeira e Rosa, 1997; Cerdeira-Pena et al., 2008; Filho e Lorena, 2001; Pillay, 2010b; Raghavjee e Pillay, 2008, 2009, 2010a,b, 2011; Souza et al., 2002a; Srndic et al., 2009);

- ***Simulated Annealing*** (Avella et al., 2007; Fonseca et al., 2012b; Nguyen et al., 2010; Poulsen e Bandeira, 2012; Zhang et al., 2010);
- **Busca Tabu** (Bello et al., 2008; De Haan et al., 2006; da Fonseca et al., 2011; Jacobsen et al., 2006; Rahoual e Saad, 2006; Santos et al., 2005; Schaerf, 1996, 1999a; Sousa et al., 2008);
- ***Greedy Randomized Adaptive Search Procedures - GRASP*** (Moura e Scaraficci, 2010; Moura et al., 2004; Souza et al., 2003);
- ***Variable Neighborhood Search - VNS*** (Brito et al., 2012; Yuri et al., 2008);
- ***Iterated Local Search - ILS*** (Coelho e de Souza, 2006; Fonseca et al., 2012a,b).

Uma característica comum entre estes algoritmos é o uso da técnica de busca local baseada no conceito de vizinhança, que permite que estes algoritmos explorem de forma eficiente o espaço de soluções do problema. Vizinhança é um subconjunto de soluções do espaço de busca, gerado a partir de uma solução Z, por uma função denominada operador de vizinhança ou estrutura de vizinhança.

Existem dois tipos de operadores de vizinhança, o de uso geral e o de uso específico:

Uso geral: são operadores que pouco dependem da estrutura do problema para ser implementado. Geralmente este tipo de operador pode ser reusado na resolução de problemas com natureza diferentes.

Uso específico ou dedicado: são operadores que dependem fortemente da estrutura do problema em estudo. Geralmente este tipo de operador serve somente ao problema para o qual foi projetado. Uma desvantagem desse tipo de operador é que sua definição para um dado problema em estudo, nem sempre é trivial.

Osogami e Imai (2000) descrevem, em seu artigo, que o desempenho de algoritmos de busca local varia muito dependendo dos operadores de vizinhança adotados. Por isso, devem ser especificados cuidadosamente. Especificamente, dois pontos são importantes: a conectividade do espaço de soluções e o tamanho das vizinhanças gerados pelos operadores implementados.

Melicio et al. (2004) citam que uma boa calibragem de parâmetros em algoritmos de busca, nunca compensará falhas de implementação como definição de operadores de vizinhança e função objetivo ineficientes. Por isso é importante definir operadores de vizinhança e função objetivo eficientes.

Di Gaspero e Schaerf (2006) mencionam que um dos pontos mais críticos no desenvolvimento de heurísticas de busca local é a definição de operadores de vizinhança.

Aladag et al. (2009) afirmam que o fator que mais afeta a eficiência de algoritmos baseados em busca local são os operadores de vizinhança definidos.

No campo do *PHE*, as pesquisas em grande escala tem tido como foco o desenvolvimento de algoritmos eficientes baseados apenas no estudo de novas estratégias de busca, como por exemplo, diversificação e intensificação. E com relação aos trabalhos que abordam o problema por meio de algoritmos de busca local, nota-se que a maioria faz uso intensivo de um conjunto de operadores de uso geral, sem se preocupar se tais operadores podem afetar o desempenho dos algoritmos implementados. Com base em revisão de literatura, vários trabalhos mencionam a definição de operadores de vizinhança como um "ponto crítico" no desenvolvimento de algoritmos de busca local. Em contrapartida, poucos trabalhos tem se dedicado a investigar a eficiência de operadores de vizinhança para o problema.

Nesta dissertação apresentamos três propostas. Na primeira, propomos a criação de uma nova base de dados com instâncias de teste para o *PHE*.

Na segunda, investigamos sobre o *PHE*, o quanto diferentes operadores de vizinhança podem afetar a eficiência de algoritmos baseados em busca local. Esta etapa consiste em:

1. Projetar dois novos operadores de vizinhança de uso dedicado para o problema;

2. Implementar os dois operadores propostos e um outro operador clássico de uso geral, disponível na literatura;

3. Analisar o quanto o desempenho de um algoritmo de busca é afetado usando cada operador de forma independente. Para fazer esta análise, definimos uma estratégia que consiste em analisar o desempenho da metaheurística *ILS* executando sobre três casos distintos. Para cada caso, a heurística de busca local incorporada por *ILS* é composta por um, e apenas um, dos três operadores mencionados. Usando esta estratégia, foi possível avaliar o quanto o desempenho de *ILS* foi afetado por cada operador.

Na terceira proposta, desenvolvemos e avaliamos a eficiência de alguns algoritmos baseados em *ILS* e *VNS*. Estes algoritmos se caracterizam por incorporarem dois ou mais procedimentos de busca local definidos com os operadores estudados na primeira fase. Nesta proposta foram testados três algoritmos baseados em *VNS* e três em *ILS*.

1.1 Motivação e justificativa

Nossa principal motivação ao investigar operadores de vizinhança eficientes para algoritmos de busca local, se deve ao fato de que, tais algoritmos tem sido aplicados com sucesso na resolução de muitos problemas práticos de grande complexidade. A justificativa para se tomar o *PHE* como estudo de caso, se deve a sua alta complexidade de resolução e importância prática.

1.2 Objetivos e contribuições

O principal objetivo é desenvolver algoritmos eficientes para o *PHE*. Os objetivos específicos são:

- Construir uma nova base de dados com instâncias de teste para o *PHE* que poderá ser usada por outros pesquisadores futuramente;
- Desenvolver novos operadores de vizinhança para o problema;
- Avaliar a eficiência de diferentes operadores;
- Desenvolver e avaliar a eficiência de vários algoritmos baseados em *ILS* e *VNS*, de forma a identificar as versões que possuem melhor desempenho para resolver o problema;

As contribuições desse trabalho são:

- A introdução na literatura, de uma nova base de dados com instâncias de teste para o *PHE*[1], visto que isto é uma deficiência neste campo de pesquisa;
- O desenvolvimento de operadores de vizinhança eficientes para uso em aplicações do *PHE*;
- Desenvolvimento de algoritmos eficientes para resolver o problema.

[1]Esta base de dados será disponibilizada gratuitamente, para fins acadêmicos, na página "http://www.din.uem.br/gpea/benchmark/" do Grupo de Pesquisa em Engenharia de Algoritmo do Departamento de Informática da Universidade Estadual de Maringá.

1.3 Organização do texto

O texto está organizado como segue: O capítulo 2 define o problema em estudo. O capítulo 3 apresenta uma revisão de literatura sobre operadores de vizinhança e os métodos *ILS* e *VNS*. O capítulo 4 descreve os operadores, os algoritmos e a base de dados proposta. O capítulo 5 apresenta os experimentos realizados e discute os resultados obtidos. O capítulo 6 finaliza com as conclusões e propostas para trabalhos futuros.

2

O problema de horários em escolas

Este capítulo define o problema de horários tratado neste trabalho. Antes de apresentar o problema real propriamente dito, são apresentadas duas versões clássicas do *PHE* encontradas na literatura, ver (Schaerf, 1999b). Estas duas formulações matemáticas servem de base para formulações de casos reais mais complexos.

O capítulo está dividido como segue: A seção 2.1 apresenta uma versão simplificada, cuja formulação consiste em um problema de localização que pode ser resolvido em tempo polinomial. A seção 2.2 descreve uma segunda versão do *PHE* baseada na formulação da seção 2.1, em que professores e turmas estão sujeitos a restrições de disponibilidade. A seção 2.3 discute algumas variantes do *PHE*. A seção 2.4 faz algumas considerações sobre a importância de *benchmarks* para compartilhamento de resultado de estudos entre os pesquisadores. A última seção do capítulo, seção 2.5, define o problema real abordado.

2.1 O problema polinomial simplificado

Considere os conjuntos $P = \{p \mid 1 \leq p \leq np,\ p \in \mathbb{N}\}$ contendo np professores, $T = \{t \mid 1 \leq t \leq nt,\ t \in \mathbb{N}\}$ contendo nt turmas, $D = \{d \mid 1 \leq d \leq nd,\ d \in \mathbb{N}\}$ contendo nd dias e $H = \{h \mid 1 \leq h \leq nh,\ h \in \mathbb{N}\}$ contendo nh horários. Seja a matriz $R_{|P|\times|T|}$, $r_{pt} \in \mathbb{N}$ denominada matriz de requerimentos, tal que r_{pt} é o número de aulas lecionadas pelo professor p a turma t.

O problema consiste em distribuir as aulas da matriz R em um quadro de horários de duração semanal com $nd \times nh$ períodos, de forma que professores e turmas não estejam envolvidos em mais que uma aula por período. Uma formulação matemática deste problema é como segue:

$$\textbf{Encontrar} \quad x_{ptdh} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh)$$

$$\textbf{Sujeito a} \quad \sum_{d=1}^{nd} \sum_{h=1}^{nh} x_{ptdh} = r_{pt} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt) \tag{2.1}$$

$$\sum_{p=1}^{np} x_{ptdh} \leq 1 \quad (1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.2}$$

$$\sum_{t=1}^{nt} x_{ptdh} \leq 1 \quad (1 \leq p \leq np,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.3}$$

$$x_{ptdh} \in \{0, 1\} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.4}$$

Em que $x_{ptdh} = 1$ se o professor p lecionar para a turma t no horário h do dia d, e $x_{ptdh} = 0$ caso contrário. A restrição (2.1) garante que todas as aulas requeridas em R sejam alocadas, a restrição (2.2) (resp. 2.3) assegura que cada turma (resp. professor) esteja envolvida no máximo em uma aula por período.

2.2 O problema de localização básico

A versão apresentada nesta seção é uma formulação mais realística do *PHE* que inclui a possibilidade de haver professores e/ou turmas indisponíveis em certos períodos.

Uma formulação matemática similar a apresentada em Schaerf (1999b) é mostrada a seguir. Para tratar as restrições de disponibilidade o modelo introduz duas matrizes binárias $M_{|T|\times|D|\times|H|}$ e $Q_{|P|\times|D|\times|H|}$, em que $m_{tdh} = 1$ (resp. $q_{pdh} = 1$) indica que a uma turma t (resp. professor p) está disponível no horário h do dia d, e $m_{tdh} = 0$ (resp. $q_{tdh} = 0$) caso contrário.

$$\textbf{Encontrar} \quad x_{ptdh} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh)$$

$$\textbf{Sujeito a} \quad \sum_{d=1}^{nd} \sum_{h=1}^{nh} x_{ptdh} = r_{pt} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt) \tag{2.5}$$

$$\sum_{p=1}^{np} x_{ptdh} \leq m_{tdh} \quad (1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.6}$$

$$\sum_{t=1}^{nt} x_{ptdh} \leq q_{pdh} \quad (1 \leq p \leq np,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.7}$$

$$x_{ptdh} \in \{0, 1\} \quad (1 \leq p \leq np,\ 1 \leq t \leq nt,\ 1 \leq d \leq nd,\ 1 \leq h \leq nh) \tag{2.8}$$

Nesta formulação as restrições (2.2) e (2.3) da formulação da seção 2.1 foram substituidas pelas restrições (2.6) e (2.7) respectivamente, as quais garantem que turmas (resp. professores) não sejam alocadas para horários que estão indisponíveis. Esta segunda formulação do *PHE* foi demonstrada ser um problema da classe NP-Completo por Even et al. (1975) por meio da redução do problema *3-SAT* ao *PHE*.

2.3 Variantes do problema

Esta seção apresenta alguns aspectos que fazem o *PHE* assumir diversas configurações (variantes) em diferentes instituições. Estas características estão ligadas a disponibilidade e manejo dos três grupos de recursos geralmente envolvidos no *PHE*: professores, turmas e salas de aulas (Post et al., 2012).

Dimensão do problema: Talvez este seja o aspecto que separa o *PHE* em duas variantes bem definidas. A primeira variante é um problema tri-dimensional envolvendo as três variáveis turmas, professores e salas. Ou seja, além de alocar as aulas de professores e turmas, as salas para ocorrência dessas aulas devem ser alocadas de forma apropriada. Haja visto que em muitos casos, espaço é um recurso escasso. A segunda variante é um caso particular da primeira, configurando um problema bidimensional envolvendo apenas as variáveis turmas e professores. Neste último caso, salas de aula são um recurso abundante, não havendo necessidade de serem alocadas, e apenas as aulas de turmas e professores são consideradas no processo de escalonamento.

Distribuição de disciplinas aos professores: Este aspecto é o mais comum entre os *PHE's* encontrados na literatura. Ele diz respeito a forma como os professores são distribuídos para lecionar as disciplinas das turmas. Geralmente esta associação é de antemão prefixada pela direção das escolas e determinada por acordo feito entre o corpo docente.

Carga horária de turmas e professores: Em muitas escolas a carga horária de aulas de algumas turmas é menor que o total de períodos de aulas semanais, configurando a necessidade de uma compactação na agenda semanal de tais turmas. O mesmo acontece para professores, devido a diferentes políticas educacionais de um país para outro, ou até dentro de um mesmo país, muitos professores tem sua carga horária de trabalho semanal distribuída entre várias escolas, e por isso sua agenda de aulas semanais deve ser compacta.

Disponibilidade de recursos: Diretamente relacionado com o aspecto anterior, muitas turmas e professores podem estar indisponíveis para participar de aulas em certos horários ou até mesmo em certos dias. Uma ocorrência muito simples deste fato são professores que ficam indisponíveis para lecionar na escola, devido estarem alocados para lecionar em outra escola em certos dias da semana. Além disso, se a alocação de sala for levada em consideração, pode haver a não disponibilidade de certas salas em alguns horários.

Aulas pré-alocadas: Outro aspecto importante que diferencia muitos *PHE's* é o fato de que em muitas escolas algumas disciplinas precisam ser pré-alocadas em horários específicos.

Aulas simultâneas com divisão de turma: Em algumas escolas, turmas são divididas em dois grupos de alunos durante alguns horários, e cada grupo é trabalhado em paralelo por professores diferentes.

Em Birbas et al. (2009) este processo ocorre quando uma turma t tem aula de laboratório e este não possui capacidade para abrigar todos os alunos, então a turma é dividida em dois grupos t_1 e t_2, o grupo t_1 assiste a disciplina que exige a sala do laboratório ministrada por um professor p_1, enquanto o grupo t_2 assiste uma disciplina com outro professor p_2.

Beligiannis et al. (2008) citam dois casos:

i) turmas são divididas durante as aulas de língua estrangeira para formar dois novos grupos temporários, *iniciantes* e *avançados*, e cada um é lecionado por professores diferentes no mesmo horário.

ii) turmas são divididas, em certos horários, para assistir aulas de disciplinas diferentes e com professores diferentes.

No Brasil, muitos colégios oferecem cursos técnicos com disciplinas eletivas. Neste caso, em dados horários pode haver duas disciplinas, cada uma com professor diferente, sendo lecionadas em paralelo para a mesma turma. Por exemplo, em colégios agrícolas, turmas tem disciplinas eletivas como *produção animal* e *produção vegetal* que são lecionadas por professores distintos. Neste caso a turma é dividida em dois grupos e as disciplinas são lecionadas em paralelo, uma para cada grupo.

Aulas simultâneas com divisão de turmas seguida de união das partes: Este aspecto ocorre quando duas turmas t_i e t_j são divididas em subgrupos de alunos t_{i1}, t_{i2}, t_{j1} e t_{j2}. Em seguida os subgrupos são temporariamente unidos para formar duas novas classes $t_h = t_{i1} \cup t_{j1}$ e $t_k = t_{i2} \cup t_{j2}$, para assistirem aulas de disciplinas distintas.

Birbas et al. (2009) citam o caso em que disciplinas de língua estrangeira são fornecidas em dois níveis, *iniciante* e *avançado*. Neste caso, devido a existência de um único professor para uma disciplina, em um dado horário um grupo t_h é formado para assistir a aula do nível *iniciante*, enquanto o restante dos alunos, grupo t_k, assiste aula de *educação física*, por exemplo.

Aula com professor auxiliar: Birbas et al. (2009) também citam o caso de algumas disciplinas que são ministradas em laboratórios, cujas as aulas necessitam de dois professores trabalhando de forma colaborativa, um para conduzir a aula e outro para auxiliar.

Schaerf (1999a) formula em seu trabalho, um modelo que leva em consideração os três últimos aspectos apresentados.

Uma ampla revisão de literatura sobre o problema de horários em escolas pode ser encontrada em Pillay (2010a); Post et al. (2012, 2008); Schaerf (1999b).

Schaerf (1999b) apresenta uma revisão de literatura sobre os três tipos de problemas de horários em instituições de ensino. Para cada um destes problemas são apresentados: algumas variantes do problema encontradas na literatura, as formulações matemáticas básicas e os métodos empregados até então para solucionar cada tipo.

Pillay (2010a) apresenta uma visão geral da pesquisa na área do *PHE*. O trabalho faz um levantamento dos estudos abordando o problema, das várias variantes existentes, restrições, métodos de resolução do problema e aponta futuras direções das pesquisas na área.

Post et al. (2012, 2008) descrevem as características do *PHE* encontradas em diversos países e identificam vários tipos de restrições ligadas a cada variante. Além disso, os autores observam que este problema tem sido menos estudado na literatura que os problemas de horários de cursos e horários de exames.

2.4 A importância dos benchmarks

Post et al. (2008) mencionam que a principal razão pela qual a pesquisa na área do *PHE* tem se desenvolvida de forma lenta e em pequena quantidade, se deve ao fato de que a maioria dos estudos tem sido realizada de forma fragmentada e abordando estudo de casos particulares. Não existindo base de dados significativas para permitir a comparação de diferentes estudos. Como solução, os autores criaram um formato de arquivo XML (*Extensible Markup Language*) para descrever instâncias do *PHE* e iniciaram a construção de uma base de dados internacional. O objetivo era juntar algumas instâncias do problema, já existentes na literatura e incentivar pesquisadores a contribuir com novas instâncias.

Atualmente a base de dados contém 50 instâncias provindas dos países: Austrália, Brasil, Dinamarca, Espanha, Finlândia, Grécia, Itália, Kosovo, Holanda, Reino Unido e África do Sul. A base de dados é pública e pode ser acessada em "http://www.utwente.nl/ctit/hstt/". Recentemente foi realizada uma competição internacional (*International Timetabling Competition* 2011) para incentivar a pesquisa na área. A competição também permitiu que as instâncias da base de dados construída fossem resolvidas por algoritmos de diversos competidores. As soluções encontradas nesta competição são importantes e servirão como parâmetro de comparação em estudos futuros.

2.5 O problema de otimização real abordado

O problema abordado neste trabalho é a variante do *PHE* que considera apenas as variáveis turmas e professores. Além das restrições básicas apresentadas na formulação do seção 2.1, várias outras restrições que visam atender requisitos pedagógicos e do corpo docente são levadas em consideração. O problema é baseado em casos reais de treze

escolas públicas de ensino médio brasileiras. De forma a obter uma definição genérica do problema nestas escolas, somente as restrições comuns entre elas foram consideradas.

No *PHE* encontrado nestas escolas, todas as atividades escolares são distribuídas sobre cinco dias durante a semana, de segunda a sexta-feira. Cada dia de atividade é dividido em três turnos, com cinco períodos de aula cada. Sendo assim, um quadro de horários se refere à escala de todas as aulas em um dado turno. Turmas são grupos disjuntos de estudantes matriculados num mesmo conjunto de disciplinas. Cada disciplina é lecionada por um único professor cuja carga horária é de antemão definida pela escola. Todas as turmas estão disponíveis em todos os períodos de aula durante a semana. Salas de aula são prefixadas para cada turma e não consideradas no processo de escalonamento das aulas, ou seja, espaço não é um recurso escasso para estas instituições. Por isto não há necessidade de se otimizar o uso deste tipo de recurso. A maioria dos professores não possui dedicação exclusiva e leciona em várias escolas. Por este motivo as aulas semanais dos professores devem se concentrar em um número mínimo de dias e respeitar os horários para os quais eles não estão disponíveis para lecionar.

A seguir, a definição 2.1 apresenta de forma precisa os dados que formam uma instância do *PHE*.

Definição 2.1 (Instância do PHE) *Uma instância do PHE se refere aos dados de entrada para a programação dos horários das aulas de um determinado turno, sendo representada por dois conjuntos:*

- *Um conjunto* $L = \{\langle t, p, \theta, \lambda, \mu\rangle | t \in T,\ p \in P,\ \theta \in \mathbb{N},\ \lambda \in \mathbb{N}\ e\ \mu \in \mathbb{N}\}$ *de quíntuplas, denominado requerimento de aulas, tal que uma quíntupla* $\langle t, p, \theta, \lambda, \mu\rangle$ *indica que a turma* t *requer* θ *aulas com o professor* p*. E* λ *e* μ *expressam respectivamente, o número máximo de aulas diárias e o número mínimo de aulas duplas consecutivas (geminadas) requeridas pelo professor* p *com a turma* t*.*

- *Um conjunto* $I = \{\langle p, d, h\rangle | p \in P,\ d \in D,\ h \in H\}$ *de triplas, denominado conjunto de horários de indisponibilidade dos professores, tal que uma tripla* $\langle p, d, h\rangle$ *indica que o professor* p *está indisponível no horário* h *do dia* d*.*

Sendo assim, o problema consiste em produzir um quadro de horários Z de duração semanal, cinco dias com cinco períodos cada, para as aulas requeridas em L, levando em consideração dois tipos de restrições, fortes e fracas.

Definição 2.2 (Restrições fortes) *São aquelas que devem ser satisfeitas para que uma solução seja viável, ou seja, passível de uso pela escola. Seja* $A = \{a_i | 1 \leq i \leq 4\}$ *o conjunto de restrições fortes. Então* A *é definido pelas restrições:*

a_1 : *O número θ de aulas requeridas por uma turma t com um professor p deve ser atendido;*
a_2 : *Uma turma t deve assistir aula com no máximo um professor p em cada horário h;*
a_3 : *Um professor p deve lecionar no máximo para uma turma t em cada horário h;*
a_4 : *Professores não devem ser alocados para lecionar em horários que estão indisponíveis.*

Definição 2.3 (Restrições fracas) *São aquelas cujo número de violações deve ser minimizado para que o quadro de horários apresente boa qualidade. O conjunto $B = \{b_j | 1 \leq j \leq 6\}$ das restrições fracas é definido pelas seguintes restrições:*

b_1 : *Uma turma t deve ser alocada para assistir no máximo λ aulas diárias com um mesmo professor p;*
b_2 : *Aulas não consecutivas de uma turma t com um professor p em um mesmo dia devem ser evitadas;*
b_3 : *O número mínimo μ de aulas duplas consecutivas (geminadas) requeridas por um professor p com uma turma t deve ser atendido sempre que possível;*
b_4 : *Professores não devem ter janelas*[1] *em sua agenda diária. Períodos para os quais o professor está indisponível não são considerados como janelas;*
b_5 : *A carga horária semanal do professor deve ser concentrada em um número mínimo de dias (compactação);*
b_6 : *A compactação na agenda semanal dos professores deve ser balanceada, de forma a não beneficiar uns e prejudicar outros.*

[1] Períodos ociosos entre dois períodos de atividade em um dado dia.

3

Revisão de literatura

Este capítulo revisa alguns conceitos sobre algoritmos baseados em técnicas de busca local e discute os trabalhos que nortearam o estudo desenvolvido nesta dissertação.

A revisão apresentada discute alguns trabalhos que empregam algoritmos de busca local para resolução de problemas do *PEH-IE*. O objetivo é fornecer uma visão geral dos tipos de operadores de vizinhança existentes nesta área. São discutidos também, alguns trabalhos de outras áreas de escalonamento, o quais contribuíram com conceitos importantes para o desenvolvimento do presente trabalho.

O capítulo está dividido como segue. A seção 3.1 apresenta alguns conceitos sobre algoritmos de busca local. A seção 3.2 discute os dois tipos de representação computacional comumente utilizados na literatura para tratar o problema de horários em escolas. Em geral, a maioria dos operadores de vizinhança disponíveis para o *PHE* são baseados nestas estruturas de dados. A seção 3.3 revisa as estratégias de busca envolvidas nos *frameworks* das metaheurísticas *ILS* e *VNS*. A seção 3.4 discute alguns trabalhos no campo do *PEH-IE* e a seção 3.5, alguns trabalhos relacionados a outras áreas de escalonamento.

3.1 Conceitos básicos

Em um problema de otimização combinatória (POC) como é o caso do *PHE*. Todas as possíveis soluções, viáveis ou não, para uma dada instância do problema, definem um conjunto discreto e finito S, denominado espaço de busca ou espaço de soluções do problema. Cada solução neste espaço de busca pode ser considerada como uma solução candidata. A resolução de um POC requer a sua formulação como um problema de

maximização ou minimização, em que se deseja maximizar ou minimizar o valor de uma função objetivo $f : S \to \mathbb{R}$.

No caso do *PHE*, o problema é formulado como um problema de minimização e f é uma medida definida pela ponderação do número de violações as restrições do problema. Sendo assim, se deseja satisfazer as restrições fortes e minimizar o número de violações nas restrições fracas do problema. Sua resolução consiste em encontrar uma solução $Z^* \in S$, tal que $f(Z^*) \leq f(Z), \forall Z \in S$, Z^* é denominado mínimo global de S e o conjunto $S^* \subseteq S$ de todas as soluções Z^* é dito conjunto de mínimos globais.

Como para a maioria dos *POCs* os melhores algoritmos conhecidos requerem tempo exponencial, pois não se conhece qualquer algoritmo que os resolvam em tempo polinomial, então a resolução deste tipo de problema geralmente é feita por métodos aproximados. Um método muito utilizado são os algoritmos heurísticos baseados em técnicas de busca local.

Uma heurística de busca local é um procedimento que começa com uma solução inicial Z_0, construída de forma aleatória ou por um método construtivo guloso qualquer e, iterativamente, substitui a solução Z atual por uma solução vizinha Z' que seja melhor ou igual a Z em sua vizinhança N(Z), até que não seja mais possível obter melhoras. Onde N é um operador de vizinhança definido apropriadamente (Blum e Roli, 2003).

Formalmente um operador de vizinhança ou estrutura de vizinhança é definido como:

Definição 3.1 (Operador de vizinhança) *Um operador de vizinhança é uma função $N : S \to P(S)$ que atribui para cada solução $Z \in S$ um subconjunto de soluções $N(Z) \subseteq S$. $P(S)$ é o conjunto das partes de S e $N(Z)$ é dito conjunto de vizinhos ou, simplesmente, vizinhança de Z.*

A definição de operadores de vizinhança permite a definição do conceito de mínimo local:

Definição 3.2 (Mínimo local) *Um mínimo local em relação a um operador de vizinhança N é uma solução $Z^{*\prime}$, tal que $\forall Z \in N(Z^{*\prime}) \Rightarrow f(Z^{*\prime}) \leq f(Z)$.*

Nas últimas décadas surgiu uma espécie de algoritmos de uso mais geral, conhecidos hoje em dia como metaheurísticas. Em resumo uma metaheurística é uma estratégia de alto nível para explorar o espaço de busca de um problema, usando diferentes métodos (Blum e Roli, 2003). A maioria das metaheurísticas baseadas em busca local guia uma ou mais heurísticas de busca local durante sua execução, para explorar o espaço de busca do problema. As mais conhecidas e amplamente abordadas na literatura são: *Iterated Local Search*, *Variable Neighborhood Search*, *Simulated Annealing* e Busca Tabu.

3.2 Representação computacional

Na literatura existem duas formas de representação computacional muito utilizadas para o *PHE*. Neste trabalho estas duas formas serão denotadas por: *visão "professores × períodos"* e *visão "turmas × períodos"*.

3.2.1 Visão "professores × períodos"

Um quadro de horários, ou simplesmente solução, é representado por uma matriz bidimensional $Z_{|P|\times nk}$ de valores inteiros, com $nk = |D| \times |H|$. Cada linha de Z representa a escala semanal de um professor p, e uma célula $z_{pk} \in \{-1, 0, 1, 2, \ldots, nt\}$ indica a atividade do professor p no período de aula k, com $k = (d-1) \times nd + h$ e $1 \leq k \leq nk$. Valores positivos maiores que zero indicam a turma t associada ao professor p no período k. Valores negativos indicam períodos para os quais o professor está indisponível e valores nulos indicam períodos de inatividade na agenda do professor. Por exemplo, supondo $nd = 5$ e $nh = 5$, se ao professor $p = 2$ for agendado uma aula com a turma $t = 4$ no horário $h = 1$ do dia $d = 2$ (terça-feira), então $k = (2-1) \times 5 + 1 = 6$ e $z_{2,6} = 4$. Esta forma de representação é utilizada em Bello et al. (2008); Cerdeira-Pena et al. (2008); Kaneko et al. (1999); Santos et al. (2005); Schaerf (1996, 1999a); Souza et al. (2002a, 2003); Yuri et al. (2008). Neste tipo de representação as restrições a_1 e a_3 da definição 2.2 são automaticamente satisfeitas. A Figura 3.1 exibe um exemplo desta representação.

Profs	Períodos				
	h_1	h_2	h_3	...	h_{nk}
p_1	4	4	1	...	10
p_2	1	0	5	...	7
p_3	9	9	-1	...	0
...	...	...	...	...	...
p_{np}	2	-1	9	...	2

Figura 3.1: Representação *visão "professores × períodos"*

3.2.2 Visão "turmas × períodos"

Um quadro de horários é representado por uma matriz bidimensional $Z_{|T|\times nk}$ de valores inteiros, com $nk = |D| \times |H|$. Cada linha de Z representa a escala semanal de uma turma t, tal que uma célula $z_{tk} \in \{-1, 0, 1, 2, \ldots, np\}$ indica a atividade da turma t no período k, com $k = (d-1) \times nd + h$ e $1 \leq k \leq nk$. Valores positivos maiores que zero indicam o professor p alocado para lecionar a turma t no período de aula k. Valores nulos indicam

períodos de inatividade na agenda da turma, caso turmas tenham horários livres durante a semana, e valores negativos podem ser utilizados para representar indisponibilidade de turmas em certos períodos se esta restrição for necessária. Esta forma de representação é utilizada em Beligiannis et al. (2008); Kaneko et al. (1999); Yuri et al. (2008). Neste tipo de representação as restrições a_1 e a_2 da definição 2.2 são automaticamente satisfeitas. A Figura 3.2 exibe um exemplo desta representação.

Turmas	Períodos				
	h_1	h_2	h_3	...	h_{nk}
t_1	10	5	15	...	15
t_2	1	4	5	...	7
t_3	2	9	1	...	6
...	...	...	...	...	...
t_{nt}	9	6	9	...	2

Figura 3.2: Representação *visão "turmas × períodos"*

Uma vantagem da representação "*turmas* × *períodos*" sobre a representação "*professores* × *períodos*" é que na prática o número de turmas é menor que o de professores, e a representação "*turmas* × *períodos*" resulta numa representação mais compacta que a forma "*professores* × *períodos*".

3.3 Metaheurísticas

Esta seção discute o *framework* geral das metaheurísticas *Iterated Local Search* e *Variable Neighborhood Search*.

3.3.1 Iterated Local Search

A metaheurística *Iterated Local Search - ILS* (Lourenço et al., 2003) é uma abordagem geral para resolução de problemas de otimização combinatória. A técnica utilizada por *ILS* consiste em explorar as soluções fazendo um passeio sobre o espaço de busca do problema. O passeio se inicia em um mínimo local, salta deste para um segundo, do segundo para um terceiro, e assim por diante.

Sendo assim, partindo de uma solução inicial Z_0, o algoritmo *ILS* executa uma heurística de busca local subordinada até ficar preso em um mínimo local Z^*. Então, iterativamente, aplica um procedimento de perturbação para escapar do mínimo local, saltando para outra região do espaço de busca, e novamente executando a heurística de busca local para atingir um outro mínimo local $Z^{*\prime}$. Se $Z^{*\prime}$ passar pelo critério de

aceitação, então $Z^{*\prime}$ será perturbado na próxima iteração. Caso contrário, retorna-se para a melhor solução Z^* encontrada até então. O algoritmo 3.1 esboça um pseudocódigo do método *ILS*.

Lourenço et al. (2003) descrevem uma revisão detalhada sobre esta metaheurística. O trabalho discute os principais módulos que compõem o método e reporta várias aplicações de *ILS*. Segundo os autores, a ideia básica de *ILS* tem um longo histórico e tem sido utilizada por pesquisadores sobre diferentes nomes durante anos.

A eficiência de *ILS* depende basicamente de quatro módulos (Lourenço et al., 2003):

- procedimento gerador de soluções iniciais;
- heurística de busca local;
- procedimento de perturbação;
- critério de aceitação de soluções.

Algoritmo 3.1 Metaheurística *ILS*

```
ITERATED-LOCAL-SEARCH()
1  Z_0 = GENERATE-INITIAL-SOLUTION()
2  Z* = LOCAL-SEARCH(Z_0)
3  repeat
4        Z' = PERTURBATION(Z*)
5        Z*' = LOCAL-SEARCH(Z')
6        Z* = ACCEPTANCE-CRITERION(Z*, Z*')
7  until (stop criterion met)
8  return  Z*
```

3.3.2 Variable Neighborhood Search

A metaheurística *Variable Neighborhood Search - VNS* (Hansen et al., 2010) é um método de uso geral para resolução de problemas de otimização combinatória. Uma versão básica do método *VNS* é apresentada nos pseudocódigos dos algoritmos 3.2 e 3.3, pois existem várias versões na literatura. A técnica empregada em *VNS* consiste basicamente numa troca sistemática de uma sequência N_k $(1 \leq k \leq k_{max})$ de vizinhanças para escapar de mínimos locais. Para evitar ciclos, antes de aplicar uma busca local, soluções são selecionados aleatoriamente pelo procedimento SHAKING na linha 6 do algoritmo 3.3.

Uma revisão da extensiva literatura sobre o método pode ser encontrada em Hansen et al. (2010).

Algoritmo 3.2 Procedimento avaliador de soluções e de troca de operadores

NEIGHBORHOOD-CHANGE($Z^*, Z^{*\prime}, k$)

```
1  if   f(Z*') < f(Z*)   then
2       Z* = Z*'
3       k = 1
4  else
5       k = k + 1
```

Algoritmo 3.3 Metaheurística *VNS*

VARIABLE-NEIGHBORHOOD-SEARCH(k_{max}, t_{max})

```
 1  Z_0 = GENERATE-INITIAL-SOLUTION()
 2  Z* = Z_0
 3  repeat
 4       k = 1
 5       repeat
 6            Z' = SHAKING(Z*, k)
 7            Z*' = LOCAL-SEARCH(Z', k)
 8            NEIGHBORHOOD-CHANGE(Z*, Z*', k)
 9       until (k > k_max)
10       t = CPUTIME()
11  until (t > t_max)
12  return   Z*
```

3.4 Operadores de vizinhança aplicados em problemas do PEH-IE

No texto desta e da próxima seção, os trabalhos apresentados são relacionados somente a abordagens que empregam algoritmos de busca local e estão organizados por ordem cronológica do ano de publicação na literatura.

Schaerf (1996, 1999a) implementa em seu algoritmo baseado em Busca Tabu e *Randomized Nonascendent Method - RNA* para resolver o *PHE*, dois operadores denominados *atomic move* e *double moves*. O operador *atomic move* é identificado por uma tripla

$\langle p, h_i, h_j \rangle$ e consiste em permutar as aulas do professor p entre os horários h_i e h_j. O operador *double moves* é um operador mais elaborado e consiste em um par de *atomic move*, de forma que o segundo tenta reparar violações geradas pelo primeiro. Uma desvantagem do operador *double moves* é que nem sempre é possível reparar as violações geradas pelo primeiro movimento. Visto que ao tentar reparar uma violação, uma nova violação pode ser gerada, como mostrado na Figura 3.3(b) e na Figura 3.3(c).

Prof.	d_1				
	h_1	h_2	h_3	h_4	h_5
p_1	3	5	3	7	3
p_2	1	4	5	6	7
p_3	2	1	1	9	6
p_4	6	3	2	2	1
p_5	9 ↔	6	9	8	2

(a) Solução Z

Prof.	d_1				
	h_1	h_2	h_3	h_4	h_5
p_1	3	5	3	7	3
p_2	1	4	5	6	7
p_3	2	1	1	9	6
p_4	6 ↔	3	2	2	1
p_5	6	9	9	8	2

(b) Solução Z' após operador *atomic move*

Prof.	d_1				
	h_1	h_2	h_3	h_4	h_5
p_1	3	5	3	7	3
p_2	1	4	5	6	7
p_3	2	1	1	9	6
p_4	3	6	2	2	1
p_5	6	9	9	8	2

(c) Solução Z' após operador *double moves*

Figura 3.3: Operadores para o *PHE*

Kaneko et al. (1999) desenvolveram um operador de vizinhança denominado *billiard movement* para melhorar o desempenho de uma combinação de dois algoritmos denominados *Really-Full-Lookahead Greedy* e *Min-Conflicts Hill-Climbing* para resolução do *PHE*. A estratégia implementada no operador permite que os algoritmos escapem de certos mínimos locais, por meio da realocação de aulas que simulam o movimento de bolas de bilhar. A Figura 3.4 exibe um exemplo dos movimentos executados por este operador. Na figura as aulas X e Y estão em colisão no horário $Tue.4$ e não podem ser movidas para o horário $Mon.4$ (sem alocação), devido a restrições de disponibilidade ou por causarem nova colisão. Então o operador move uma das aulas X ou Y, em colisão, para o horário $Tue.1$, enquanto a aula C é movida para o horário $Mon.4$.

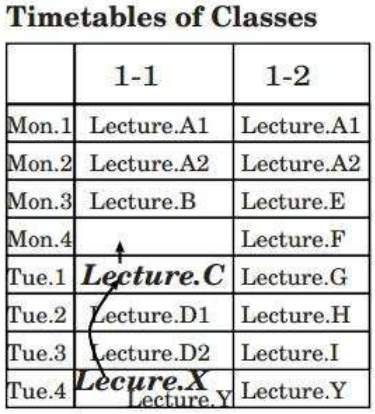


Figura 3.4: Operador *billiard movement* (Kaneko et al., 1999)

Souza et al. (2002b) abordam o problema de alocação de salas de aula (PAS)[1] por meio de um algoritmo *VNS*. Dois operadores de vizinhança são utilizados:

- *Movimento 1-optimal*: realoca as aulas de uma turma de uma sala para outra;
- *Movimento 2-optimal*: troca as aulas de duas turmas entre salas.

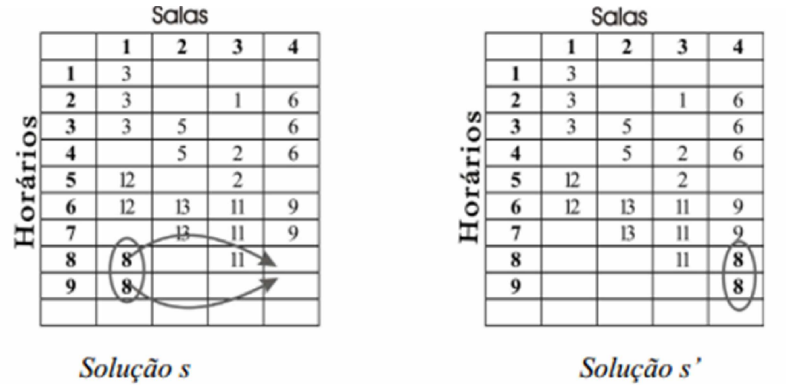
Embora estes operadores (ver Figura 3.5) realoquem mais que uma aula ao mesmo tempo, eles são semelhantes aos operadores de Schaerf (1996, 1999a). Porém são aplicados a problemas diferentes. Operadores deste tipo podem ser considerados como de uso geral, pois como descrito anteriormente neste trabalho, eles não dependem de um conhecimento específico da estrutura do problema para serem implementados e podem se adaptar a vários problemas diferentes. Isto já não ocorre, por exemplo, com os operadores implementados no procedimento "*Intraclasses-Interclasses*" de Souza et al. (2003) a seguir, que são fortemente dependentes da estrutura do problema e da forma como ele foi representado.

Souza et al. (2003) propõem para o *PHE*, um algoritmo iterativo híbrido baseado em *GRASP* (*Greedy Randomized Adaptive Search Procedure*) e Busca Tabu denominado *GTS-II*. O algoritmo consiste basicamente em repetir, por um número máximo de iterações *GTSmax*, os seguintes procedimentos:

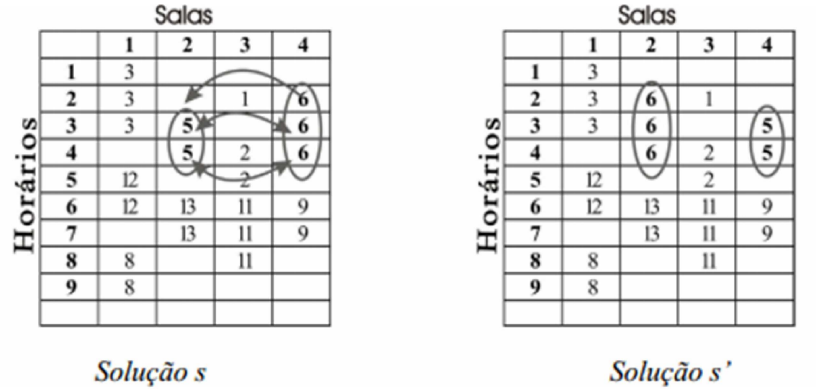
i) construir uma solução inicial usando um procedimento *GRASP* parcialmente guloso;

ii) aplicar um algoritmo Busca Tabu para refinar a solução obtida com *GRASP*.

[1] O *PAS* é um problema derivado do *PHC*.



(a) Movimento 1-optimal



(b) Movimento 2-optimal

Figura 3.5: Operadores para o *PAS* (Souza et al., 2002b)

Após *GTSmax* iterações a melhor solução encontrada é retornada. O algoritmo Busca Tabu explora o espaço de busca usando um operador definido por $\langle p, i, j \rangle$ que consiste em permutar as aulas do professor p alocadas nos horários i e j. Além desses operadores, é usado um procedimento de busca local denominado "*Intraclasses-Interclasses*". Os operadores implementados neste procedimento são definidos pela realocação de um conjunto de aulas no grafo de uma turma t, cuja sequência de realocação gera um ciclo de custo negativo. No grafo de aulas de uma turma t, cada vértice representa um horário e o professor designado para atender a turma t naquele horário. Arestas $\langle i, j \rangle$ representam a variação no valor da função objetivo, caso a aula do professor p no horário i, seja transferida para o horário j.

Valouxis e Housos (2003) modelam e resolvem o *PHE* usando uma abordagem baseada em *Constraint Programming - CP*. A abordagem consiste em duas fases:

i) resolver um modelo *CP* para todo o problema durante um intervalo de tempo;

ii) aplicar um procedimento de busca local para melhorar a solução encontrada após resolução do modelo *CP*.

A busca local de Valouxis e Housos (2003) consiste em um procedimento iterativo cujo operador é constituído por um modelo *CP* definido sobre partes menores do problema. Especificamente, o procedimento "congela" as variáveis do modelo para certos dias e o modelo é resolvido com as variáveis estando livre somente para um ou dois dias. Este procedimento é repetido até que não seja possível obter melhoras.

Melicio et al. (2004) apresentam uma análise comparativa entre dois operadores de vizinhança denominados *single move* e *double move*. Os operadores são idênticos aos operadores de Schaerf (1996, 1999a) e consistem em:

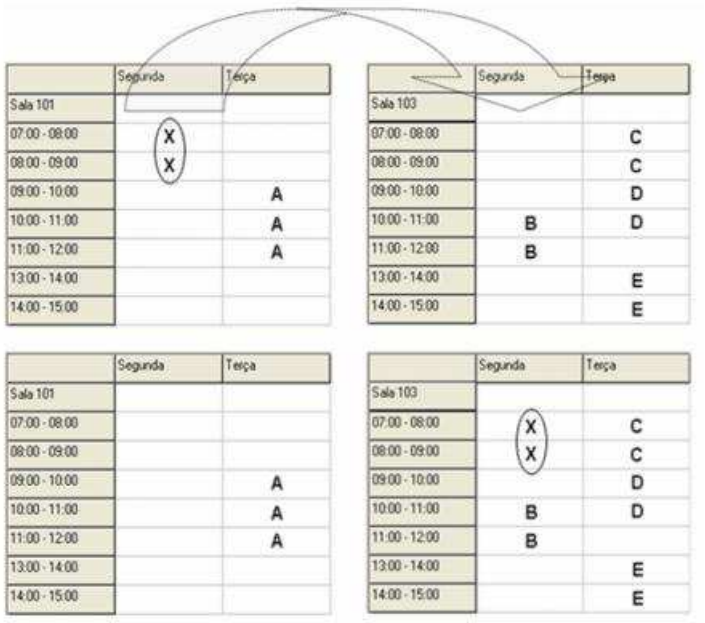
- *single move*: realocar aleatoriamente, uma aula de um horário para outro;
- *double move*: permutar aleatoriamente, duas aulas de horário;

Na análise, Melicio et al. (2004) comparam o tamanho das vizinhanças geradas por cada operador. Para uma análise de eficiência, cada operador foi implementado em um algoritmo *Hill-Climbimg* e aplicado ao *PHE* de três escolas em Portugal, com diferentes tamanhos. No texto os autores enfatizam ainda, a importância de operadores de vizinhança que gerem vizinhanças pequenas, de forma a limitar o tamanho do espaço de busca do problema.

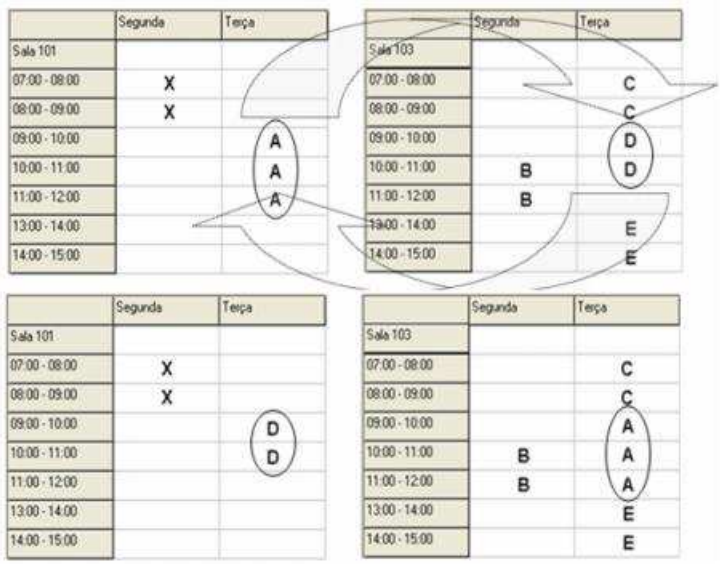
Santos et al. (2005) propõem para o *PHE*, um algoritmo de Busca Tabu com uma estratégia de diversificação baseada em dois tipos de memórias.

- *Memória baseada em transição*: consiste em memorizar a frequência que as aulas de cada par $\langle p, t \rangle$ foram realocadas pelo algoritmo.
- *Memória baseada em residência*: consiste em memorizar o número de iterações que a *m-ésima* aula do professor p com a turma t ocupou o horário k.

Baseado nestas memórias o algoritmo penaliza movimentos executados pelo operador em uso, para introduzir diversificação e escapar de mínimos locais. O operador de vizinhança implementado é similar ao usado em Schaerf (1996, 1999a); Souza et al. (2003), sendo definido por uma tripla $\langle p, i, j \rangle$. Um movimento consiste em permutar as aulas de um professor p, alocadas nos horários i e j.



(a) Realocação de aulas de uma sala para outra



(b) Troca de aulas entre salas

Figura 3.6: Operadores para o *PAS* (Subramanian et al., 2006)

Subramanian et al. (2006) abordam o *PAS* por meio de um algoritmo Busca Tabu. Os dois operadores de vizinhança empregados (ver Figura 3.6) são os mesmos utilizados na abordagem de Souza et al. (2002b).

Di Gaspero e Schaerf (2006) mencionam que o uso e a troca sistemática entre diversos operadores, aumentam a capacidade do algoritmo de explorar o espaço de busca do problema. A proposta de Di Gaspero e Schaerf (2006) consiste na composição de operadores. A partir de um conjunto $N = \{N_i | 1 \leq i \leq k\}$ de operadores básicos, são construídos operadores compostos e mais complexos, usando três princípios:

i) *União de operadores*: o algoritmo de busca seleciona a cada iteração, um operador dentro do conjunto $\bigcup_{1 \leq i \leq k} N_i$ de operadores, ver Figura 3.7(a);

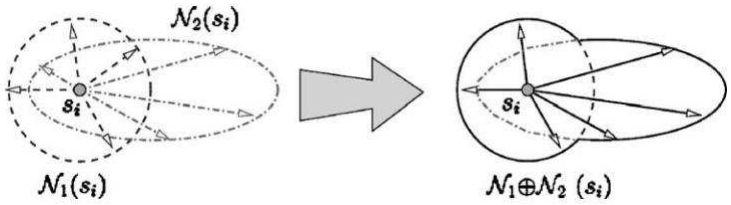
ii) *Sequência de operadores*: um operador é composto por uma cadeia de operadores básicos, ver Figura 3.7(b). Por exemplo, um vizinho Z' é obtido a partir de uma solução Z, por uma função composta $Z' = N_1 \circ \ldots \circ N_k(Z)$ obedecendo a sequência $i = \langle 1, 2, \ldots, k \rangle$;

iii) *Composição de operadores*: é um generalização do caso anterior, em que um vizinho Z' é obtido a partir de uma solução Z, por uma função composta de h operadores básicos, ou seja, $Z' = N_1 \circ \ldots \circ N_j \circ \ldots \circ N_h(Z)$, $j \in \{1, 2, \ldots, k\}$, ver Figura 3.7(c).

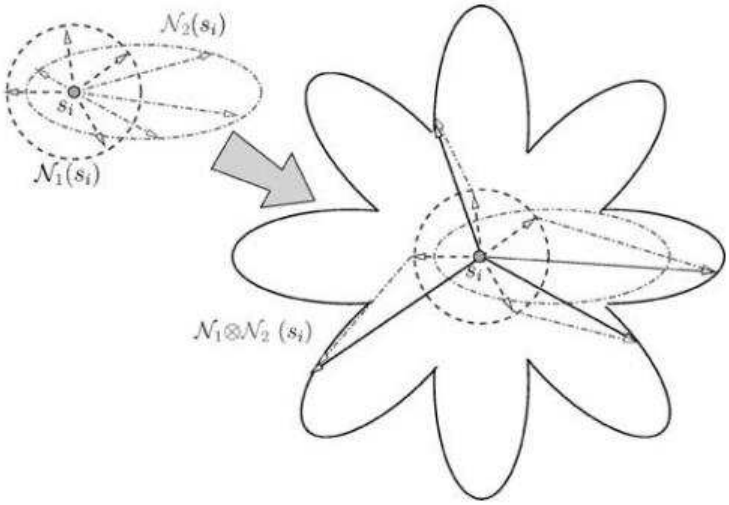
Di Gaspero e Schaerf (2006) implementaram seus operadores usando Busca Tabu e *Hill-Climbing* aplicados na resolução do *PHC* e *PHEX*. Contudo, o artigo não descreve quais foram os operadores básicos utilizados na abordagem.

Abdullah et al. (2007) adaptam para o *PHEX*, os operadores propostos por Ahuja et al. (2001) para o problema da árvore geradora mínima capacitada. Estes operadores consistem em identificar movimentos de melhora, encontrando (ciclos/caminhos) de movimentos em um grafo de melhoramento. No caso do *PHEX*, os operadores consistem em encontrar ciclos de trocas dos exames alocados em certos horários. A Figura 3.8(a) e a Figura 3.8(b) mostram um clico de trocas, onde os exames são realocados de um horário para outro, de forma cíclica. Nas figuras os horários são representados pelos conjuntos S_k $(1 \leq k \leq 3)$ e os exames por E_i $(1 \leq i \leq 7)$. A Figura 3.8(c) mostra o grafo de melhoramento, onde uma aresta $\langle i, j \rangle$ representa o custo de inserção de um exame i em um horário S_k, ao mesmo tempo que um exame j é ejetado.

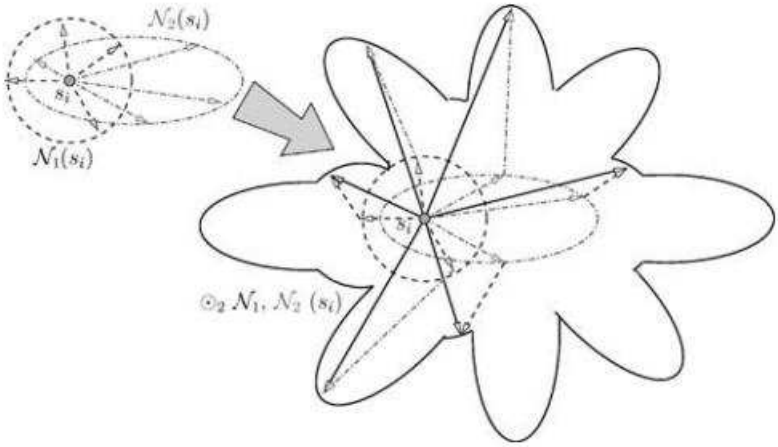
Avella et al. (2007) resolvem o *PHE* com um algoritmo *Simulated Annealing*. A busca local implementada no algoritmo é baseada no conceito de operadores que exploram vizinhanças em larga escala. No trabalho de Avella et al. (2007) os vizinhos de uma solução



(a) União de operadores

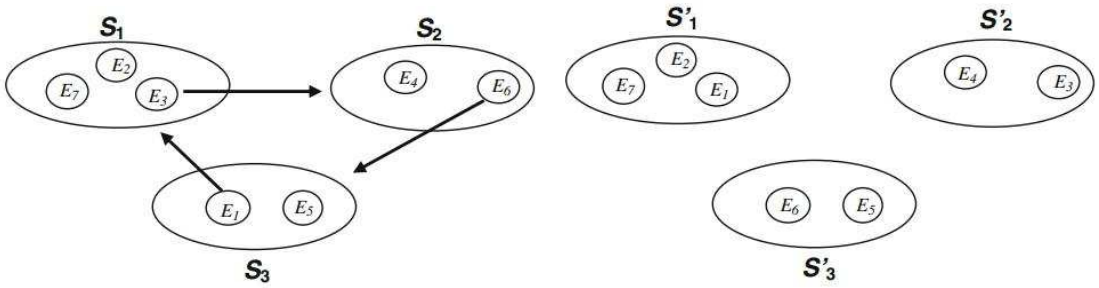


(b) Sequência de operadores



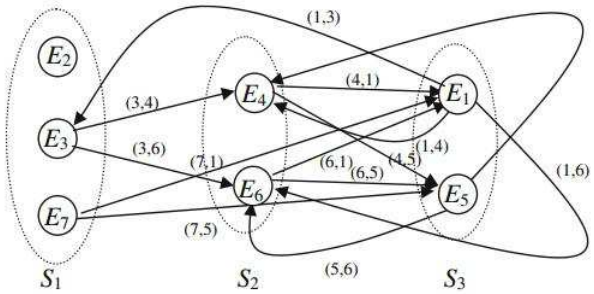
(c) Composição de operadores

Figura 3.7: Operadores para o *PHC* e *PHEX* (Di Gaspero e Schaerf, 2006)



(a) Antes da aplicação do operador

(b) Após a aplicação do operador



(c) Grafo de melhoramento

Figura 3.8: Operadores para o *PHEX* (Abdullah et al., 2007)

são explorados resolvendo-se um modelo de programação linear inteira para o problema, de forma que as aulas de todos os professores, exceto dois, estão pré-alocadas.

Beligiannis et al. (2008) implementam em seu Algoritmo Genético para o *PHE*, um operador de mutação que é representado por uma tripla $\langle t, h_i, h_j \rangle$ e consiste em permutar as aulas de uma turma t alocadas nos horários h_i e h_j. O operador genético é semelhante aos operadores apresentados em Santos et al. (2005); Schaerf (1996, 1999a); Souza et al. (2003), mas com algumas diferenças na forma de ser aplicado. Ele é aplicado de duas formas:

i) de acordo com certa probabilidade as aulas de dois horários totalmente aleatórios são selecionadas e permutadas;

ii) como no caso anterior, mas as aulas selecionadas devem pertencer aos professores com as piores agendas semanais em termos de qualidade.

Cerdeira-Pena et al. (2008) implementam, para resolução do *PHE*, uma abordagem híbrida combinando Algoritmos Genéticos e um método de busca local *RNA*. Os operadores de vizinhança embutidos no método *RNA* e o operador de mutação implementado nos algoritmos genéticos propostos, são os mesmos operadores propostos por Schaerf (1996, 1999a) (ver Figura 3.3), sendo denominados por Cerdeira-Pena et al. (2008) como *simple-moves* e *double-moves*.

Sousa et al. (2008) apresentam em seu artigo que trata da resolução do *PHE*, uma abordagem heurística combinando busca local aleatória e Busca Tabu. Vizinhos de uma solução são explorados utilizando operadores similares aos apresentados inicialmente por Schaerf (1996, 1999a). Os operadores são aplicados de forma a reparar soluções, caso sejam obtidas inviabilidades com a troca das aulas de um professor p entre dois horários h_i e h_j.

Clark et al. (2008) descrevem detalhes da implementação de um software genérico para resolver problemas do *PEH-IE*. A abordagem heurística utilizada se baseia na técnica de reparo de restrições, em que uma restrição violada é selecionada e um movimento é gerado, por um operador, para repará-la. Dois tipos de operadores são implementados:

- *Teacher-Period Swap Moves*: as aulas alocadas a dois pares $\langle professor, horário \rangle$ são permutadas;
- *Room-Period Swap Moves*: as aulas alocadas a dois pares $\langle sala, horário \rangle$ são permutadas.

Yuri et al. (2008) usam dois tipos distintos de operadores para o *PHE*. Considerando que $\langle t, h_i, h_j \rangle$ é o operador que troca as aulas de uma turma t entre os horários h_i e h_j. Os operadores de Yuri et al. (2008) são:

- *Swaps neighborhoods*: dada uma turma t, executar k diferentes trocas do tipo $\langle t, h_i, h_j \rangle$;
- *Kernighan-Lin neighborhoods*: dada uma solução Z. Executar os seguintes passos:
  1) a partir de Z, selecionar a tripla $\langle t, h_i, h_j \rangle$ que gere o melhor vizinho Z' na vizinhança de Z, mesmo que Z' seja pior que Z;
  2) definir $Z = Z'$;
  3) repetir k vezes os passos 1 e 2.

Os operadores de Yuri et al. (2008) são implementados e usados para resolver o *PHE* com Busca Tabu e *VNS*.

Raghavjee e Pillay (2009) aplicam em sua abordagem por Algoritmo Genético para resolver o *PHE*, um procedimento semelhante a uma busca local *Hill-Climbing* para reparar indivíduos (soluções) que violam as restrições fortes em novas gerações. O procedimento de busca local consiste em aplicar iterativamente um operador de mutação. O operador de mutação permuta de horário duas aulas de uma dada turma, em que pelo menos uma das aulas permutadas estão gerando conflitos. Esta estratégia permite que as novas gerações contenham apenas soluções viáveis.

Zhang et al. (2010) também usam operadores baseados em permutar duas aulas de horário, em sua abordagem via *Simulated Annealing*, para resolver 15 instâncias do *PHE*.

Lü et al. (2011) estudam a eficiência de quatro operadores de vizinhança para o *PHC*. Supondo que o *PHC* seja representado por uma matriz $X_{m \times n}$, onde m é o número de salas de aula, n é o número de períodos e $x_{r,h}$ representa o curso alocado na sala r no período h. Os quatro operadores analisados em Lü et al. (2011) são:

- *SimpleMove*: realocar a aula de um curso c_1 alocada numa posição x_{r_k,h_i} para uma posição livre x_{r_l,h_j};
- *SimpleSwap*: trocar de posições duas aulas de cursos diferentes. Ou seja, mover a aula de um curso c_1 alocado numa posição x_{r_k,h_i} para uma posição x_{r_l,h_j} e mover a aula do curso c_2 alocado na posição x_{r_l,h_j} para a posição x_{r_k,h_i};

- *KempeMove*: é definido por uma cadeia de movimentos de troca conhecida como *Kempe chain*. Supondo que as aulas do *PHC* na matriz $X_{m \times n}$ sejam transformadas em um grafo G, onde vértices sejam aulas de cursos e arestas conectem aulas com estudantes ou professores em comum.

  Uma *Kempe chain* é definida como um conjunto de aulas K ($|K| \geq 3$) que formam uma componente conexa em um subconjunto de aulas que pertencem a dois períodos h_i e h_j distintos. Supondo que K seja uma componente conexa em relação aos períodos h_i e h_j, K_i seja o conjunto de aulas de K que estão alocadas no período h_i e K_j o conjunto de aulas de K alocadas no período h_j. Então o operador *KempeMove* consiste em realocar as aulas de K_i para h_j e as de K_j para h_i. Após as aulas serem realocadas de período, um problema de designação é resolvido para alocar as aulas em salas apropriadas.

  A Figura 3.9 ilustra um exemplo deste operador. Neste exemplo existem quatro *Kempe chain* possíveis, porém somente K_4 gera um movimento válido para o operador *KempeMove*. Para K_1 e K_2 o número de aulas envolvidas é menor que 3. Para K_3 não é possível gerar uma solução viável, pois trocando $\{C_2, C_3\}$ e $\{C_7, C_8, C_{10}\}$ de horário, torna o número de aulas no período h_i maior que o número de salas disponíveis.

- *KempeSwap*: é uma generalização do operador *KempeMove* e consiste em trocar as aulas de duas *Kempe chain* distintas. Por exemplo, $K_1 \cup K_3$ e $K_2 \cup K_4$ são movimentos possíveis, ver Figura 3.9.

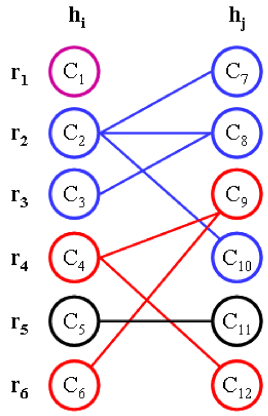
Os operadores de Lü et al. (2011) foram implementados e testados usando uma heurística de busca local *Steepest Descent* e algoritmos baseados nas metaheurísticas Busca Tabu e *ILS*.

3.5 Operadores de vizinhança aplicados em outros tipos de problemas de escalonamento

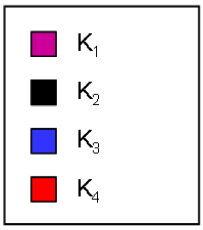
Nos trabalhos relacionados abaixo é possível observar um importante conceito aplicado no desenvolvimento de operadores de vizinhança.

Conceito 3.1 *Operadores devem ser definidos, se possível, para gerar apenas soluções que não violem as restrições fortes do problema em questão, ou pelo menos algumas delas.*

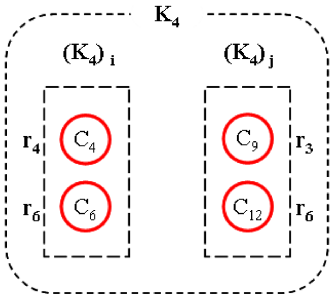
De maneira mais sucinta podemos definir este conceito como:



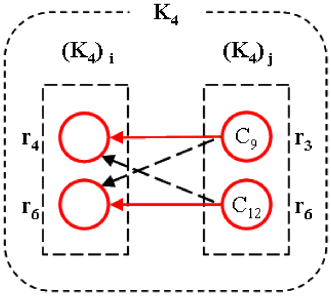
(a) Grafo das aulas alocadas em h_i e h_j antes de aplicar o operador *Kempe-Move*



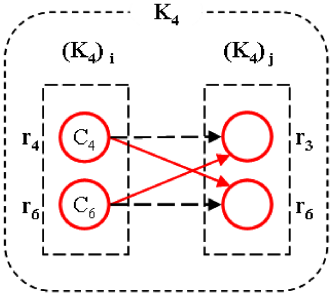
(b) *Kempe chain* possíveis



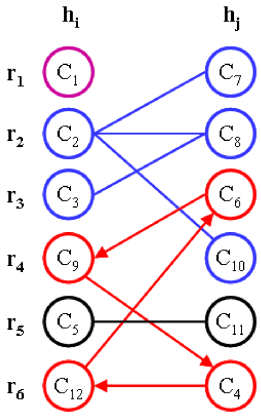
(c) *Kempe chain* K_4 e seus subconjuntos $(K_4)_i$ e $(K_4)_j$



(d) Realocando as aulas de $(K_4)_j$ para o período h_i e resolvendo um problema de designação para alocar salas apropriadas



(e) Realocando as aulas de $(K_4)_i$ para o período h_j e resolvendo um problema de designação para alocar salas apropriadas



(f) Grafo das aulas alocadas em h_i e h_j após aplicar o operador *KempeMove*

Figura 3.9: Exemplo do operador *KempeMove* para o *PHC*

Conceito 3.2 *Operadores que preservam restrições.*

Van Laarhoven et al. (1992) apresentam um *Simulated Annealing* para resolver o problema de *Job Shop Scheduling*. Os operadores de vizinhança utilizados permitem reduzir o espaço de busca eliminando soluções candidatas inviáveis. Provas matemáticas são apresentadas para demostrar que realmente o operador de vizinhança utilizado reduz o tamanho das vizinhanças geradas. Também é apresentado um teorema que prova a convergência assintótica do algoritmo *Simulated Annealing* para um ótimo global, quando utilizando tal operador. A abordagem proposta foi comparada com os métodos heurísticos dedicados de outros dois autores e embora não tenha conseguido melhores resultados com relação ao tempo de execução, ainda assim, conseguiu resultados quando não melhores, muito próximos ou iguais aos métodos dedicados. Porém, sobre tempos mais longos de execução. Por este fato os autores concluíram que heurísticas dedicadas são mais eficientes que heurísticas de uso geral apenas em tempo de computação, mas não em capacidade de encontrar boas soluções.

Dell'Amico e Trubian (1993) aplicam Busca Tabu na resolução do problema de *Job Shop Scheduling*. O trabalho apresenta dois tipos de operadores de vizinhança e provas de conectividade do espaço de busca do problema (a partir de uma solução viável qualquer, sempre é possível aplicar uma sequência do operador definido, que leva a um mínimo global) e viabilidade (toda solução gerada pelo operador definido é viável).

Osogami e Imai (2000) apresentam um estudo teórico que classifica vários operadores de vizinhança, para o problema de escalonamento de enfermeiros, em relação a conectividade do espaço de soluções e tamanho das vizinhanças geradas. Os operadores de Osogami e Imai (2000) são baseados nos conceitos usados anteriormente por Van Laarhoven et al. (1992) e Dell'Amico e Trubian (1993). A principal característica dos operadores de Osogami e Imai (2000) é que as vizinhanças geradas pelos operadores, contêm somente soluções que satisfazem algumas restrições importantes do problema. Estas restrições são denominadas por eles, de *restrições elementares do problema*. Os autores enfatizam que este tipo de operador é preferível, porque eles restringem o espaço de busca, evitando que soluções inviáveis sejam visitadas. Isto contribui para que algoritmos de busca local consigam explorar de forma eficiente o espaço de soluções do problema.

Burke et al. (2004) publicam uma suíte de operadores para o problema de escalonamento de enfermeiros. Seus operadores possuem uma variedade de características. Dentre eles, alguns não violam as restrições fortes do problema. Burke et al. (2004) enfatizam ainda, que é melhor usar simples heurísticas com uma variedade de operadores, que usar heurísticas muito sofisticadas e que são cegas para muitas partes do espaço de busca.

4

Proposta

Este capítulo apresenta a abordagem heurística proposta para tratar o *PHE* da seção 2.5. Os algoritmos propostos são baseados nos *frameworks* das metaheurísticas *ILS* e *VNS* da seção 3.3.

O texto do capítulo está organizado como segue: A seção 4.1 define a forma de representação e a função objetivo utilizada para avaliar as soluções do *PHE*. A seção 4.2 define o procedimento utilizado para gerar soluções iniciais. A seção 4.3 discute a técnica empregada nas heurísticas de busca local utilizadas pelos algoritmos. A seção 4.4 define os operadores de vizinhança propostos. A seção 4.5 apresenta os algoritmos propostos. E por último, na seção 4.6 é apresentada a base de dados proposta para o *PHE*.

4.1 Representação e função objetivo

Uma solução do *PHE*, definição 4.1, é representada por uma variação da *visão "turmas × períodos"* discutida na seção 3.2.2.

Definição 4.1 (Solução do PHE) *Um quadro de horários é armazenada em uma matriz tridimensional $Z_{|T|\times|D|\times|H|}$ de valores inteiros, onde uma célula $z_{tdh} \in \{1, 2, \ldots, np\}$ indica a atividade da turma t no horário h do dia d, ou seja, indica o professor p designado para lecioná-la.*

Observe que turmas estão sempre disponíveis e em atividades em todos os dias e horários. Além disso, como na representação da seção 3.2.2, as restrições fortes a_1 e a_2

são automaticamente satisfeitas. Esta forma de representação foi adotada pelo fato de ser compacta em termos de espaço (memória) e gerar vizinhanças menores que a representação "*professores* $\times$ *períodos*" para alguns tipos de operadores.

A função objetivo $f : S \to \mathbb{R}$, definição 4.2, que associa a cada solução Z um número real, é composta pela soma ponderada do número de violações a cada restrição do *PHE* apresentadas nas definições 2.2 e 2.3. A função f permite mensurar o nível de viabilidade (atendimento as restrições fortes) e qualidade (atendimento as restrições fracas) de uma solução. Esta função é utilizada pela metaheurística, para guiar a busca pelo espaço de soluções do problema, cujo objetivo é encontrar soluções que minimizem o valor de f.

Definição 4.2 (Função objetivo) *Uma solução Z do PHE é avaliada de acordo com a seguinte expressão:*

$$f(Z) = f_A(Z) + f_B(Z) \tag{4.1}$$

O primeiro termo mede o nível de viabilidade e o segundo mede a qualidade da solução. Os termos parciais da equação 4.1 são calculados por:

$$f_A(Z) = f_{a_3}(Z) + f_{a_4}(Z) \tag{4.2}$$

$$f_{a_3}(Z) = \alpha_{a_3} \times \beta_{a_3} \tag{4.3}$$

$$f_{a_4}(Z) = \alpha_{a_4} \times \beta_{a_4} \tag{4.4}$$

$$f_B(Z) = \sum_{b_j \in B} \alpha_{b_j} \times \beta_{b_j} \tag{4.5}$$

Onde:

O peso α_{a_i} (resp. α_{b_j}) reflete a importância relativa de se minimizar a quantidade de violações β_{a_i} (resp. β_{b_j}) da restrição $a_i \in A$ (resp. $b_j \in B$), com $i = \{3, 4\}$ e $j = \{1, \ldots, 6\}$.

Da definição 4.2, para que uma solução Z seja viável, ela deve possuir $f_A(Z) = 0$. O número de violações em cada restrição é quantificado como segue:

$\beta_{a_3} = \sum_{p \in P} \sum_{d \in D} \sum_{h \in H} (\pi_{pdh} - 1), \forall(\pi_{pdh} > 1)$. Onde π_{pdh} é o número de aulas do professor p no período h do dia d;

$\beta_{a_4} = \sum_{p \in P} \sum_{d \in D} \sum_{h \in H} \rho_{pdh}$. Onde $\rho_{pdh} = 1$ se o professor p está indisponível e alocado para lecionar no horário h do dia d, e $\rho_{pdh} = 0$ caso contrário;

$\beta_{b_1} = \sum_{t \in T} \sum_{p \in P} \sum_{d \in D} (\sigma_{tpd} - \lambda_{tp})$, $\forall(\sigma_{tpd} > \lambda_{tp})$. Onde σ_{tpd} é o número de aulas alocadas para a turma t como o professor p no dia d e λ_{tp} é o número máximo de aulas diárias (definição 2.1);

$\beta_{b_2} = \sum_{t \in T} \sum_{p \in P} \sum_{d \in D} \tau_{tpd}$. Onde τ_{tpd} é o número de aulas alocadas para a turma t como o professor p no dia d se as aulas são não consecutivas, e $\tau_{tpd} = 0$ caso contrário;

$\beta_{b_3} = \sum_{t \in T} \sum_{p \in P} (\mu_{tp} - \phi_{tp})$, $\forall(\mu_{tp} > \phi_{tp})$. Onde μ_{tp} é o número mínimo de aulas geminadas requeridas pelo professor p com a turma t (definição 2.1) e ϕ_{tp} é o efetivo número de aulas geminadas alocadas;

$\beta_{b_4} = \sum_{p \in P} \sum_{d \in D} \eta_{pd}$. Onde η_{pd} é o número de janelas na agenda do professor p no dia d. A unidade considerada ao quantificar as janelas na agenda do professor é o período com ociosidade. Por exemplo, se a agenda de um professor em um dado dia for composta por: atividade no primeiro período, indisponível no segundo, ocioso no terceiro e quarto, e atividade no último período, então este professor terá duas janelas;

$\beta_{b_5} = \sum_{p \in P} \chi_p$. Onde $\chi_p = (da_p - md_p)$, da_p é o número de dias alocados para o professor p na escala e md_p é o número mínimo de dias possíveis;

$\beta_{b_6} = max\{\chi_p\}$, $p \in P$. Onde $max\{\chi_p\}$ é o valor máximo das diferenças em β_{b_5}.

4.2 Procedimento gerador de soluções iniciais

Os métodos mais utilizados na literatura para geração de soluções iniciais para o *PHE* são: procedimentos gulosos e procedimentos aleatórios.

Procedimentos gulosos: basicamente consistem em alocar primeiro as aulas com maior prioridade, de forma a satisfazer as restrições mais importantes.

Procedimentos aleatórios: consistem em alocar as aulas de forma aleatória sem se preocupar com quaisquer restrições.

Segundo alguns autores, soluções iniciais construídas com métodos gulosos, permitem que uma metaheurística de resolução consiga convergir para boas soluções em menor tempo de computação que soluções geradas por métodos aleatorizados. Isto se deve ao fato das soluções construídas por métodos gulosos possuírem menos violações que as construídas por métodos aleatorizados.

Visto que um dos objetivos deste trabalho é testar a capacidade de operadores de vizinhança. Optou-se por gerar soluções iniciais com um procedimento aleatorizado. Pois

soluções iniciais com muitas violações, exigem maior esforço da metaheurística no processo de busca e consequentemente, uso intenso dos operadores de vizinhança.

O pseudocódigo do procedimento gerador de soluções iniciais está descrito no algoritmo 4.1. O procedimento recebe como parâmetro o conjunto L (definição 2.1) de requerimento de aulas de uma dada instância do *PHE*, em seguida aloca, de forma aleatória, as aulas requeridas em cada quíntupla $e \in L$ e retorna a solução Z gerada.

Algoritmo 4.1 Procedimento gerador de soluções iniciais

GENERATE-RANDOM-SOLUTION(L)

```
1  Inicializar Z
2  for each e ∈ L do
3        t = e.t
4        p = e.p
5        NumberOfLesson = e.θ
6        while NumberOfLesson > 0 do
7              Alocar o professor p em uma célula z_tdh ∈ Z aleatória
8              NumberOfLesson = NumberOfLesson − 1
9  return Z
```

4.3 Busca local

De forma simplificada, para problemas de minimização, dada uma solução inicial Z_0 de entrada, uma heurística de busca local deve se mover de Z_0 para um mínimo local Z'. Hansen et al. (2010) descrevem duas técnicas geralmente utilizadas para construir heurísticas de busca local: *best improvement* e *first improvement*.

Algoritmo 4.2 Heurística baseada na técnica *best improvement*

BEST-IMPROVEMENT-HEURISTIC(Z_0, N)

```
1  Z = Z_0
2  repeat
3        Z' = Z
4        Z = min{Z'' ∈ N(Z)}
5  until (f(Z) ≥ f(Z'))
6  return Z'
```

***Best improvement*:** a heurística de busca local parte de uma solução $Z' = Z_0$ de entrada, e a cada iteração, substitui Z' pela solução $Z = min\{Z'' \in N(Z')\}$ enquanto $f(Z) < f(Z')$. Caso contrário termina. Esta técnica consiste em explorar toda a vizinhança e se mover para o vizinho com menor valor de função objetivo enquanto ele for menor que a solução corrente.

***First improvement*:** dependendo do tamanho da vizinhança, a técnica anterior pode consumir muito tempo de CPU para explorá-la completamente. Uma alternativa, é a cada iteração, mover-se para o primeiro vizinho que tenha valor de função objetivo menor que a solução corrente. Ou seja, a partir de uma solução $Z' = Z_0$ de entrada, a cada iteração, Z' é substituída pela primeira solução Z_i encontrada em $N(Z')$ que satisfaça $f(Z_i) < f(Z')$. O processo é repetido enquanto existe $Z_i \in N(Z')$ tal que $f(Z_i) < f(Z')$, caso contrário termina.

Os algoritmos 4.2 e 4.3 apresentam os pseudocódigos destas técnicas.

Algoritmo 4.3 Heurística baseada na técnica *first improvement*

FIRST-IMPROVEMENT-HEURISTIC(Z_0, N)

```
1  Z = Z_0
2  repeat
3       Z' = Z
4       i = 0
5       repeat
6            i = i + 1
7            Z = min{Z, Z_i}, Z_i ∈ N(Z')
8       until (f(Z) < f(Z') or i = |N(Z')|)
9  until (f(Z) ≥ f(Z'))
10 return Z'
```

Neste trabalho adotamos a técnica *first improvement* por ser mais rápida e apresentar melhores resultados em testes preliminares realizados.

4.4 Operadores de vizinhança

Esta seção explana os operadores de vizinhança investigados. Na literatura, se observa que a maioria dos estudos sobre algoritmos de busca local se dedicam ao desenvolvimento de novas estratégias de busca, visando por exemplo, diversificação e intensificação durante a exploração do espaço de busca do problema. Por outro lado, como discutido na seção

3.5, alguns pesquisadores como Dell'Amico e Trubian (1993); Osogami e Imai (2000); Van Laarhoven et al. (1992) observaram que o desempenho destas estratégias é afetado pelos operadores de vizinhança implementados. Sendo assim, eles propuseram operadores específicos para o domínio do problema tratado. Tais operadores refinam o espaço de busca e eliminam do processo de busca, um grande número de soluções infactíveis, permitindo que os algoritmos executem de forma mais eficiente.

No texto que segue apresentamos uma discussão sobre este conceito e nas seções seguintes ele é utilizado na definição dos dois operadores de vizinhança propostos para o *PHE*.

Inicialmente Dell'Amico e Trubian (1993); Van Laarhoven et al. (1992) desenvolveram operadores de vizinhança para o problema de sequenciamento de máquinas com a propriedade de que *"toda solução $Z' \in N(Z)$ é viável para uma solução Z viável"*. Osogami e Imai (2000) baseados nesta propriedade, introduziram para o problema de escalonamento de enfermeiros - *PEE*, o seguinte conceito sobre operadores de vizinhança.

Conceito 4.1 *No PEE algumas restrições tem alta prioridade. Então é possível definir o espaço de solução como sendo o conjunto das soluções que satisfazem algumas destas restrições, de forma que o espaço de busca seja reduzido e um algoritmo de busca local possa explorá-lo de forma eficiente.*

As restrições de alta prioridade geralmente são as restrições fortes do problema ou um subconjunto delas.

A principal contribuição de Osogami e Imai (2000) foi a introdução da noção de operadores com capacidade de refinar o espaço de busca do problema, de forma que um algoritmo de busca local consiga evitar durante o processo de busca, a visita a um subconjunto de soluções indesejáveis. Um exemplo de subconjunto de soluções indesejáveis seria o conjunto das soluções inviáveis do problema. De fato, qualquer solução dentro deste conjunto não é desejável de ser verificada. Portanto, isto induz ao fato de que busca dentro deste conjunto é apenas desperdício de tempo de *CPU*.

Tendo em vista os conceitos acima, Osogami e Imai (2000) desenvolveram operadores de vizinhança para o *PEE*, que permitem que uma heurística de busca local explore somente sobre um subconjunto restrito do espaço de soluções. Este subconjunto é composto por todas as soluções que satisfazem algumas das principais restrições do problema.

A seguir são apresentados os operadores *DM*, *MT* e *TQ*. Dentre os três operadores, *MT* e *TQ* são operadores propostos e projetados com base nos conceitos apresentados anteriormente. O operador *DM* é um operador de uso geral encontrado na literatura e

muito utilizado em algoritmos de resolução do *PHE*. Ele também é largamente empregado em algoritmos de busca local para resolver vários outros tipos de problemas de otimização combinatória.

4.4.1 Operador Double Move

O termo *Double Move - DM* é utilizado por Melicio et al. (2004) e significa que duas aulas são trocadas de horários, de forma semelhante ao operador *atomic move* de Schaerf (1996, 1999a). No *PHE* deste trabalho o operador *DM* é definido sobre a estrutura de dados descrita na definição 4.1 e consiste em permutar de horários, duas aulas de uma dada turma. A definição 4.3 formaliza este operador e a Figura 4.1 mostra um exemplo.

Definição 4.3 (Operador *Double Move*) *O operador DM é uma função* $DM : S \rightarrow P(S)$ *que atribui para cada solução* $Z \in S$ *uma vizinhança* $DM(Z) \subseteq S$*, dada por:*

$$DM(Z) = \{Z' \in S | z'_{td_i h_i} = z_{td_j h_j} \ e \ z'_{td_j h_j} = z_{td_i h_i}, \ z_{td_i h_i} \neq z_{td_j h_j}\}$$

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	9	6	9	8	2

(a) Solução Z antes de aplicar o operador *DM*

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	6	9	9	8	2

(b) Solução vizinha Z' após aplicar o operador *DM*

Figura 4.1: Exemplo do operador *DM*

O operador *DM* é um operador de uso geral empregado em larga escala com algoritmos de busca local para resolver diversos problemas, inclusive nos problemas de horários de cursos e exames universitários. No *PHE* este operador é amplamente utilizado, como discutido na seção 3.4.

Uma desvantagem deste operador é a geração de uma vizinhança com muitas soluções indesejáveis, ou seja, dada uma solução viável Z para o *PHE*, uma grande quantidade de soluções em $DM(Z)$ viola a restrição a_3, principal restrição do *PHE*. Como mostrado na Figura 4.1(b), após aplicar o operador *DM* sobre uma solução viável Z, a solução vizinha Z' passa a violar duas vezes a restrição a_3. O professor 6 passa a lecionar duas turmas t_4

e t_5 no horário h_1 e consequentemente, o professor 9 passa a lecionar as turmas t_3 e t_5 no horário h_2 do dia d_1.

4.4.2 Operador Matching

O operador *Matching - MT* é baseado na resolução do problema de designação - *PD*. Este operador é uma adaptação da técnica usada por Constantino et al. (2009) para resolver o *PEE*. Em Constantino et al. (2009) a resolução do *PEE* é feita por heurísticas que consistem em resolver sucessivos *PDs* para melhorar uma escala inicial. O algoritmo usado por Constantino et al. (2009) para resolver *PDs* é o algoritmo de tempo polinomial proposto por Carpaneto e Toth (1987), este mesmo algoritmo também será utilizado neste trabalho. A definição 4.4 formaliza o operador *MT*.

Definição 4.4 (Operador *Matching*) *Sejam:*

- Z *uma solução do PHE;*
- $\Delta P_t \subseteq P$ *o conjunto de professores que lecionam para uma turma* t*;*
- $\Delta Z_t = \{z_{tdh} \in \Delta P_t\}$ *um multiconjunto de* ΔP_t*, definido pelas aulas de* t *em* Z*;*
- $U_t = \{Y \in P(\Delta Z_t) | Y = \Delta P_t\}$*;*
- $\widehat{Y} = \{(z_{tdh})_i | 1 \leq i \leq |\Delta P_t|,\ i \in \mathbb{N}\}$ *um conjunto indexado,* $\widehat{Y} \in U_t$*.*

O operador MT é uma função $MT : S \rightarrow P(S)$ *que atribui para cada solução* $Z \in S$*, uma vizinhança* $MT(Z) \subseteq S$ *formada por todas as soluções* $Z' \in S$ *obtidas a partir de* Z*, resolvendo-se um PD formulado sobre um conjunto* $\widehat{Y} \in U_t$*, dado por:*

$$\textbf{\textit{Minimizar}} \quad \sum_{i=1}^{|\widehat{Y}|} \sum_{j=1}^{|\widehat{Y}|} c_{ij} x_{ij}$$

$$\textbf{\textit{Sujeito a}} \quad \sum_{i=1}^{|\widehat{Y}|} x_{ij} = 1 \quad (1 \leq j \leq |\widehat{Y}|) \tag{4.6}$$

$$\sum_{j=1}^{|\widehat{Y}|} x_{ij} = 1 \quad (1 \leq i \leq |\widehat{Y}|) \tag{4.7}$$

$$x_{ij} \in \{0, 1\} \quad (1 \leq i \leq |\widehat{Y}|,\ 1 \leq j \leq |\widehat{Y}|) \tag{4.8}$$

Onde a matriz de custos $C_{|\widehat{Y}| \times |\widehat{Y}|}$ *é computada como segue:*

i) $Z = Z - \widehat{Y}$*, remove-se de* Z *os professores do conjunto* $\widehat{Y}$*;*

ii) c_{ij} *é o valor de* $f(Z)$ *se o professor que está na posição* i *em* $\widehat{Y}$*, for realocado para o horário onde estava alocado o professor da posição* j *de* $\widehat{Y}$*;*

Após a resolução do PD, Z' *é obtida realocando-se em* Z*, os professores de* $\widehat{Y}$ *com base na variável resposta* x_{ij} *do PD.*

A Figura 4.2 mostra um exemplo deste operador. Para simplificar, vamos considerar a resolução do *PHE* tratando apenas a restrição a_3 com peso $\alpha_{a_3} = 1$. O operador é aplicado sobre as aulas da turma t_3 na solução Z da Figura 4.2(a), onde:

- $\Delta P_t = \{1, 2, 6, 9\}$;
- $\Delta Z_t = \{2, 9, 1, 9, 6\}$;
- $\widehat{Y} = \{\overbrace{2}^{t_3 d_1 h_1}, \overbrace{9}^{t_3 d_1 h_2}, \overbrace{1}^{t_3 d_1 h_3}, \overbrace{6}^{t_3 d_1 h_5}\}$.

As figuras (b), (c), (d) e (e) da Figura 4.2 ilustram o processo de construção e resolução do *PD* para o conjunto $\widehat{Y} = \{2, 9, 1, 6\}$ e a Figura 4.2(f) mostra a solução Z' obtida.

Uma observação importante é que $\widehat{Y}$ deve respeitar a definição de conjuntos, ou seja, não deve possuir professores repetidos. A razão para isto, é que professores repetidos impedem a construção da matriz de custo $C_{|\widehat{Y}| \times |\widehat{Y}|}$.

Como mencionado anteriormente, este operador foi desenvolvido com base nos conceitos apresentados no início da seção 4.4. Uma propriedade interessante deste operador é apresentada a seguir.

Propriedade 4.1 *Dada uma solução* Z *do PHE. Então* $\forall Z' \in MT(Z)$*,* $f(Z') \leq f(Z)$*.*

A propriedade 4.1 diz que na pior das hipóteses, quando o operador MT for executado sobre uma solução Z do PHE, a solução vizinha Z' gerada, terá valor de função objetivo $f(Z')$ igual ao da solução atual Z. Sendo assim, MT possui um "filtro natural" que permite eliminar do processo de busca, as soluções que são piores que a solução atual, pois a vizinhança gerada por MT contém somente soluções que não deterioram a função objetivo atual.

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	7	15	17	15
t_2	1	6	5	4	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	6	9	9	8	10

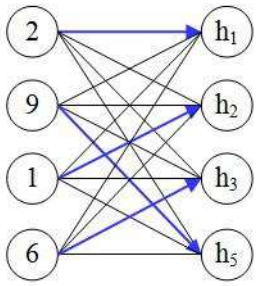
(a) Selecionando um conjunto $\widehat{Y}$ em uma solução Z com $f(Z) = 2$

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	7	15	17	15
t_2	1	6	5	4	7
t_3				9	
t_4	6	10	12	2	11
t_5	6	9	9	8	10

(b) Removendo o conjunto $\widehat{Y}$ da solução Z. A resultante é a solução $Z - \widehat{Y}$ com $f(Z - \widehat{Y}) = 1$

	h_1	h_2	h_3	h_5
p=2	1	1	1	1
p=9	1	2	2	1
p=1	2	1	1	1
p=6	2	2	1	1

(c) Matriz de custos $C_{|\widehat{Y}|\times|\widehat{Y}|}$



(d) Resolução do PD

	h_1	h_2	h_3	h_5
p=2	1	1	1	1
p=9	1	2	2	1
p=1	2	1	1	1
p=6	2	2	1	1

(e) Designação x_{ij}

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	7	15	17	15
t_2	1	6	5	4	7
t_3	2	1	6	9	9
t_4	6	10	12	2	11
t_5	6	9	9	8	10

(f) Solução vizinha Z' com $f(Z') = 1$, após realocar os professores de $\widehat{Y}$ com base na variável resposta x_{ij}

Figura 4.2: Exemplo do operador MT

4.4.3 Operador Torque

O operador Torque[1] - TQ é uma generalização do operador DM e está fundamentado na necessidade de reparar as violações causadas na restrição a_3, após aplicar o operador DM.

[1]Este operador é uma analogia com um sistema de duas forças paralelas com sentidos opostos, que atuam sobre um corpo tendendo a causar rotação.

Em outras palavras, uma vez que o operador DM seja aplicado e este gere novas violações na restrição a_3, aplica-se uma cadeia finita de movimentos DM até que as violações geradas sejam reparadas. Além disto, este operador herda algumas características do conceito de cadeia de movimentos conhecida como *Kempe chain*, ver exemplo apresentado na Figura 3.9 da seção 3.4. A Figura 4.3 ilustra a estratégia fundamental do operador TQ.

O restante do texto desta seção discute um modelo baseado em grafos para este operador. A definição 4.5 formaliza este modelo.

Definição 4.5 (Operador *Torque*) *Sejam: Z uma solução do PHE; $t \in T$; $d_i, d_j \in D$, $h_i, h_j \in H$ com $d_i \neq d_j$ ou $h_i \neq h_j$; e um grafo $G = (V, A)$ onde:*

- *$V = \{v_t | t \in T\}$ é o conjunto de vértices de G, formado por pares ordenados de aulas $\langle z_{tdihi}, z_{tdjhj} \rangle$, tal que:*

$$\langle z_{tdihi}, z_{tdjhj} \rangle \in V \iff z_{tdihi} \neq z_{tdjhj} \quad (1 \leq t \leq |T|) \tag{4.9}$$

- *$A = \{\langle u, v \rangle | u \neq v \text{ e } u, v \in V\}$ é o conjunto de arestas de G e existe uma aresta em A conectando dois vértices u e v cujos atributos são $\langle i_u, j_u \rangle$ e $\langle i_v, j_v \rangle$ respectivamente, se as seguintes condições são satisfeitas:*

$$i_u = j_v \text{ ou } j_u = i_v \tag{4.10}$$

$$\forall \langle u, v \rangle, \langle u, k \rangle \in A \text{ se } i_u = j_v \text{ e } i_u = j_k \text{ então } v = k \tag{4.11}$$

$$\forall \langle u, v \rangle, \langle u, k \rangle \in A \text{ se } j_u = i_v \text{ e } j_u = i_k \text{ então } v = k \tag{4.12}$$

O operador TQ é uma função $TQ : S \to P(S)$ que atribui para cada solução $Z \in S$, uma vizinhança $TQ(Z) \subseteq S$, onde cada solução $Z' \in TQ(Z)$ é obtida a partir de Z selecionando-se uma componente conexa de G e permutando-se de horário, entre si, os pares de aulas em cada vértice da componente conexa.

A condição 4.9 impede que vértices com atributos i e j iguais pertençam ao grafo, pois a troca de horários entre duas aulas iguais é uma operação sem efeito. A condição 4.10 permite que dois vértices sejam ligados com arestas, somente se seus atributos opostos são iguais. As condições 4.11 e 4.12 serão explicadas mais adiante.

O operador TQ apresenta uma propriedade interessante com relação aos conceitos apresentados no início da seção 4.4.

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	9	6	9	8	2

(a) Operador DM

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	6	9	9	8	2

(b) Violações geradas após aplicar DM

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	9	6	9	8	2

(c) Operador TQ

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	5	10	15	17	15
t_2	1	4	5	6	7
t_3	9	2	1	9	6
t_4	10	6	12	2	11
t_5	6	9	9	8	2

(d) Solução Z' após aplicar TQ

Figura 4.3: Exemplo do operador TQ

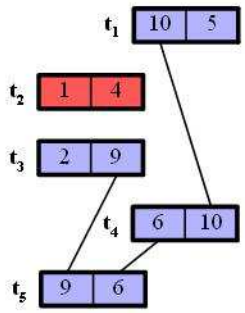
Propriedade 4.2 *Dada uma solução Z do PHE. Então $\forall Z' \in TQ(Z)$, $f_{a_3}(Z') \leq f_{a_3}(Z)$.*

Esta propriedade garante que, dada uma solução Z, TQ nunca gera uma solução vizinha Z' cujo número de violações na restrição a_3 seja maior que o de Z. Sendo assim, se Z for uma solução que satisfaz a restrição a_3, então Z' também satisfará. Note que a propriedade 4.2 garante que se um algoritmo de busca local, usando TQ, for executado a partir de uma solução que satisfaz a restrição a_3, então o algoritmo será capaz de evitar visita a soluções que violam a_3. Isto o habilitará para explorar somente sobre um conjunto restrito de soluções do espaço de busca.

O exemplo da Figura 4.4 ilustra a propriedade 4.2. Dado uma solução Z sem conflitos em a_3 (Figura 4.4(a)), após aplicar o operador, uma nova solução Z' sem conflitos é gerada (Figura 4.4(c)) permutando-se de horário, as aulas (professores) dos vértices que formam a componente conexa (subgrafo) colorido em azul no grafo da Figura 4.4(b). Note na Figura 4.4(b), que cada vértice representa uma turma t e um vértice é composto por dois atributos, o primeiro atributo representa a aula de t que está alocada no horário $h_i = h_1$ e o segundo representa a aula alocada no horário $h_j = h_2$. Note também, que dois vértices são ligados por arestas somente se eles tem atributos opostos iguais.

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	10	5	15	17	15
t_2	1	4	5	6	7
t_3	2	9	1	9	6
t_4	6	10	12	2	11
t_5	9	6	9	8	2

(a) Solução Z sem conflitos em a_3



(b) Grafo G para $d_i = d_j = d_1$, $h_i = h_1$ e $h_j = h_2$

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	5	10	15	17	15
t_2	1	4	5	6	7
t_3	9	2	1	9	6
t_4	10	6	12	2	11
t_5	6	9	9	8	2

(c) Solução vizinha Z' sem conflitos em a_3, obtida após TQ, a partir da componente marcada com cor azul

Figura 4.4: Exemplo do operador TQ - explorando sobre o conjunto de soluções que satisfazem a_3

Uma vantagem do operador TQ é que ele pode ser empregado a partir de uma solução que contenha conflitos na restrição a_3, para facilmente repará-los e obter um solução sem conflitos. A Figura 4.5(a) mostra uma solução com dois conflitos.

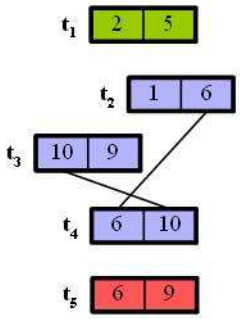
- **Conflito 1**: o professor 6 está lecionando as turmas t_4 e t_5 no horário h_1;
- **Conflito 2**: o professor 9 está lecionando as turmas t_3 e t_5 no horário h_2.

A regra para reparar conflitos consiste em:

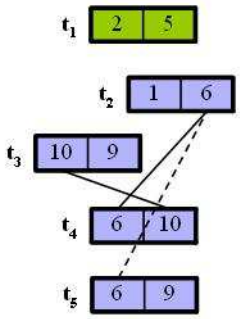
- **Passo 1**: encontrar um professor p com uma ou mais aulas em conflito em um dado horário;
- **Passo 2**: selecionar uma das aulas e realocá-la para um horário ao qual o professor p está ocioso;

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	2	5	7	17	15
t_2	1	6	6	4	1
t_3	10	9	5	9	2
t_4	6	10	10	2	11
t_5	6	9	9	8	10

(a) Solução Z com violações em a_3



(b) Grafo G para $d_i = d_j = d_1$, $h_i = h_1$ e $h_j = h_2$ com três componentes conexas



(c) Grafo G para $d_i = d_j = d_1$, $h_i = h_1$ e $h_j = h_2$ considerando somente a condição 4.10

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	2	5	7	17	15
t_2	6	1	6	4	1
t_3	9	10	5	9	2
t_4	10	6	10	2	11
t_5	6	9	9	8	10

(d) Solução vizinha Z' após aplicar TQ para reparar o conflito 2

Turmas	d_1				
	h_1	h_2	h_3	h_4	h_5
t_1	2	5	7	17	15
t_2	6	1	6	4	1
t_3	9	10	5	9	2
t_4	10	6	10	2	11
t_5	9	6	9	8	10

(e) Solução vizinha Z' após aplicar TQ no grafo que considera somente a condição 4.10

Figura 4.5: Exemplo do operador TQ - reparando conflitos na restrição a_3

- **Passo 3**: repetir os passos 1 e 2 até que não existam conflitos.

A Figura 4.5(b) e a Figura 4.5(d) mostram um exemplo em que esta regra foi aplicada para reparar o **conflito 2**. A solução da Figura 4.5(d) mostra a solução obtida após o operador TQ reparar o **conflito 2**. Note que ela foi obtida após permutar as aulas do

vértice t_3, Figura 4.5(b), cuja segunda aula é uma das aulas envolvidas no **conflito 2**, e dos demais vértices pertencentes a mesma componente conexa de t_3.

A Figura 4.5(c) e a Figura 4.5(e) ilustram um exemplo no qual as condições 4.11 e 4.12 da definição 4.5 são usadas. Estas condições são necessárias para impedir que uma aresta adicional (aresta pontilhada no grafo da Figura 4.5(c)) ligando o vértice t_2 ao vértice t_5 seja adicionada ao grafo. Se esta aresta fosse adicionada, a regra para reparar conflitos falharia da forma que foi aplicada anteriormente e o resultado seria a solução da Figura 4.5(e) com os mesmos conflitos em horários trocados.

4.5 Algoritmos propostos

As seções seguintes descrevem os algoritmos propostos para a resolução do *PHE*. Ao todo são propostos nove algoritmos. Na seção 4.5.1 são descritos três algoritmos **"Mono-Operadores"**[2] baseados em *ILS*, cuja estratégia é avaliar a eficiência dos operadores de vizinhança discutidos na seção 4.4. Na seção 4.5.2 são descritos outros seis algoritmos **"Poli-Operadores"**[3], três baseados em *ILS* e três em *VNS*.

Na abordagem empregada neste trabalho o *PHE* é resolvido em duas fases:

Primeira fase: Consiste em encontrar uma solução viável para o problema. Nesta fase apenas as restrições a_3 e a_4 da definição 2.2 são levadas em consideração. O algoritmo parte de uma solução inicial gerada pelo procedimento descrito no algoritmo 4.1 e é executado até que as quantidades β_{a_3} e β_{a_4} (número de violações descritas na seção 4.1) sejam zeradas.

Segunda fase: É a fase de melhoramento. O algoritmo parte da solução viável encontrada na primeira fase, considerando agora as demais restrições do problema, e é executado durante um intervalo de tempo fixo.

4.5.1 Algoritmos Mono-Operadores

Esta seção define uma estratégia para analisar a eficiência de cada um dos operadores discutidos na seção 4.4. São apresentadas três versões de *ILS*, cada uma implementa uma busca local baseada em apenas um dos operadores da seção 4.4. O objetivo com a implementação destas versões é analisar o quanto cada operador afeta o resultado final do método *ILS*. Este método foi escolhido pelo fato de funcionar com uma única heurística de

[2]Este termo foi usado pelo fato destes algoritmos utilizarem um único operador.
[3]Este termo foi usado pelo fato destes algoritmos empregarem mais que um operador.

busca local baseada em apenas um operador, o que o torna ideal para nossos propósitos. Isto já não seria apropriado usando *VNS*, visto que este método funciona com vários operadores.

Algoritmo 4.4 Perturbação aleatória

N-Random-Perturbation(Z, N, n)

```
1  while  (n > 0)  do
2        Z = Random Z' ∈ N(Z)
3        n = n − 1
4  return  Z
```

Algoritmo 4.5 Busca local determinista usando *first improvement*

Deterministic-FIH(Z, N)

```
1  repeat
2        Z' = Z
3        i = 0
4        repeat
5              i = i + 1
6              Z = min{Z, Z_i}, Z_i ∈ N(Z')
7        until (f(Z) < f(Z') or i = |N(Z')|)
8  until (f(Z) ≥ f(Z'))
9  return  Z'
```

Os três algoritmos propostos são:

1. ***ILS-DM* (algoritmo 4.6):** Neste algoritmo o procedimento de busca local (algoritmo 4.5) é baseado na técnica *first improvement* e definido com o operador *DM*. O procedimento de perturbação (algoritmo 4.4), consiste em aplicar dois movimentos aleatórios na solução Z, usando o operador *DM*.

2. ***ILS-TQ* (algoritmo 4.7):** Neste, o procedimento de busca local tem a mesma estrutura da busca local do anterior, mas usando o operador *TQ*. O procedimento de perturbação (algoritmo 4.4) é composto por um movimento aleatório usando o operador *TQ*.

3. ***ILS-MT* (algoritmo 4.9):** Neste, a busca local é composta pelo algoritmo 4.8 usando a estratégia *first improvement* e executando o operador *MT*. Os vizinhos

gerados pelo operador MT são explorados de forma aleatória, visto que o tamanho da vizinhança deste operador cresce desordenadamente em função da quantidade de turmas. As perturbações são geradas com dois movimentos aleatórios do operador DM, por meio do algoritmo 4.4.

Algoritmo 4.6 Algoritmo *ILS* com busca local usando o operador DM

```
ILS-DM(Z_0, t_max)
 1  Z = DETERMINISTIC-FIH(Z_0, DM)          // busca local
 2  Z* = Z                                  // melhor solução encontrada
 3  NotImproved = 0
 4  repeat
 5      Z = N-RANDOM-PERTURBATION(Z, DM, 2)
 6      Z = DETERMINISTIC-FIH(Z, DM)        // busca local
 7      if f(Z) < f(Z*) then
 8          NotImproved = 0
 9      else
10          NotImproved = NotImproved + 1
11      if f(Z) ≤ f(Z*) then                // critério de aceitação
12          Z* = Z
13      if NotImproved ≥ 3 then             // se não houve melhoras após três iterações
14          Z = Z*                          // retornar para Z*
15          NotImproved = 0
16      t = CPUTIME()
17  until (t > t_max ou f(Z) = 0)
18  return Z*
```

Com o intuito de introduzir diversificação na busca, as versões de *ILS* propostas neste trabalho, permitem que o algoritmo aceite até três soluções (mínimos locais) piores e consecutivas. Caso uma solução melhor não seja encontrada após três iterações, o algoritmo reinicia a busca a partir da melhor solução encontrada até o momento. Isto é controlado pela variável $NotImproved$, que controla o número de iterações sem melhora.

4.5.2 Algoritmos Poli-Operadores

Os algoritmos propostos nesta seção foram separados em dois grupos: o primeiro grupo, baseado em *ILS*, é formado por três algoritmos. O segundo grupo, também formado por três algoritmos, se baseia em *VNS*.

***Iterated Multi Local Search*:** Neste grupo de algoritmos a heurística de busca local embutida dentro de *ILS* é formada por "composição de heurísticas usando os

Algoritmo 4.7 Algoritmo *ILS* com busca local usando o operador *TQ*

ILS-TQ(Z_0, t_{max})

```
1  Z = DETERMINISTIC-FIH(Z_0, TQ)          // busca local
2  Z* = Z                                  // melhor solução encontrada
3  NotImproved = 0
4  repeat
5      Z = N-RANDOM-PERTURBATION(Z, TQ, 1)
6      Z = DETERMINISTIC-FIH(Z, TQ)        // busca local
7      if  f(Z) < f(Z*)  then
8          NotImproved = 0
9      else
10         NotImproved = NotImproved + 1
11     if  f(Z) ≤ f(Z*)  then              // critério de aceitação
12         Z* = Z
13     if  NotImproved ≥ 3  then           // se não houve melhoras após três iterações
14         Z = Z*                          // retornar para Z*
15         NotImproved = 0
16     t = CPUTIME()
17 until (t > t_max ou f(Z) = 0)
18 return  Z*
```

Algoritmo 4.8 Busca local aleatorizada usando *first improvement*

RANDOMIZED-FIH(Z, n)

```
1  repeat
2      Z' = Z
3      i = n × |T|
4      while  (i > 0)  do
5          t = Random t' ∈ T               // selecionar uma turma t aleatória
6          Ŷ = Random Ŷ' ∈ U_t             // selecionar um conjunto aleatório Ŷ em U_t
7          Z = MT(Z, Ŷ)                    // resolver o PD para Ŷ e retornar o vizinho de Z
8          i = i − 1
9  until (f(Z) ≥ f(Z'))
10 return  Z'
```

Algoritmo 4.9 Algoritmo *ILS* com busca local usando o operador *MT*

ILS-MT(Z_0, t_{max})

```
1  Z = Randomized-FIH(Z_0, 6)                    // busca local
2  Z* = Z                                        // melhor solução encontrada
3  NotImproved = 0
4  repeat
5       Z = N-Random-Perturbation(Z, DM, 2)
6       Z = Randomized-FIH(Z, 6)                 // busca local
7       if  f(Z) < f(Z*)  then
8            NotImproved = 0
9       else
10           NotImproved = NotImproved + 1
11      if  f(Z) ≤ f(Z*)  then      // critério de aceitação
12           Z* = Z
13      if  NotImproved ≥ 3  then // se não houve melhoras após três iterações
14           Z = Z*                  // retornar para Z*
15           NotImproved = 0
16      t = CpuTime()
17 until (t > t_max ou f(Z) = 0)
18 return  Z*
```

Algoritmo 4.10 Busca local usada pelo algoritmo *IMLS-DM-TQ-MT*

FIH-DM-TQ-MT(Z)

```
1  repeat
2       Z' = Z
3       Z = Deterministic-FIH(Z, DM)          // busca local 1
4       Z = Deterministic-FIH(Z, TQ)          // busca local 2
5       Z = Randomized-FIH(Z, 6)              // busca local 3
6  until (f(Z) ≥ f(Z'))
7  return  Z'
```

Algoritmo 4.11 Algoritmo *IMLS-DM-TQ-MT*

```
IMLS-DM-TQ-MT(Z_0, t_max)
 1  Z = FIH-DM-TQ-MT(Z_0)                    // busca local composta
 2  Z* = Z                                   // melhor solução encontrada
 3  NotImproved = 0
 4  repeat
 5      Z = N-Random-Perturbation(Z, TQ, 1)
 6      Z = FIH-DM-TQ-MT(Z)                  // busca local composta
 7      if f(Z) < f(Z*) then
 8          NotImproved = 0
 9      else
10          NotImproved = NotImproved + 1
11      if f(Z) ≤ f(Z*) then                 // critério de aceitação
12          Z* = Z
13      if NotImproved ≥ 3 then              // se não houve melhoras após três iterações
14          Z = Z*                           // retornar para Z*
15          NotImproved = 0
16      t = CpuTime()
17  until (t > t_max ou f(Z) = 0)
18  return Z*
```

algoritmos 4.5 e 4.8. Esta estratégia é semelhante a noção de "composição de funções matemáticas". Por exemplo, seja $FIH_{comp}(Z)$ uma heurística composta pelas heurísticas $FIH_1(Z)$ e $FIH_2(Z)$, então um mínimo local $Z^{*\prime}$, obtido por $Z^{*\prime} = FIH_{comp}(Z)$ é como segue: $Z' = FIH_1(Z)$, $Z^{*\prime} = FIH_2(Z')$.

Os três algoritmos propostos neste grupo são descritos a seguir.

1. ***IMLS-DM-TQ-MT*:** a heurística de busca local é composta pelas heurísticas Deterministic-FIH usando o operador *DM*, seguida por Deterministic-FIH com o operador *TQ* e por último, Randomized-FIH que emprega o operador *MT*.
2. ***IMLS-MT-TQ*:** a heurística de busca local é composta pelas heurísticas Randomized-FIH com *MT*, seguida de Deterministic-FIH com o operador *TQ*.
3. ***IMLS-TQ-MT*:** a heurística de busca local é composta pelas heurísticas Deterministic-FIH com o operador *TQ*, seguida de Randomized-FIH com MT.

O algoritmo *IMLS-DM-TQ-MT* está descrito no algoritmo 4.11 e o algoritmo 4.10 esboça o pseudocódigo da heurística de busca local usada por este algoritmo. Os demais algoritmos *IMLS-MT-TQ*, *IMLS-TQ-MT* e suas heurísticas de busca local são desenvolvidos de forma análoga.

Algoritmo 4.12 Algoritmo *VNS-DM-TQ-MT*

```
VNS-DM-TQ-MT(Z_0, t_max)
 1  k_max = 8
 2  Z* = Z_0
 3  repeat
 4       k = 1
 5       repeat
 6            Z' = N-Random-Perturbation(Z*, TQ, 1)
 7            if   k = 1   then
 8                 Z*' = Deterministic-FIH(Z', DM)
 9            elseif k = 2   then
10                 Z*' = Deterministic-FIH(Z', TQ)
11            else
12                 Z*' = Randomized-FIH(Z', k − 2)
              // critério de aceitação e troca de operador
13            Neighborhood-Change(Z*, Z*', k)
14       until (k > k_max)
15       t = CpuTime()
16  until (t > t_max ou f(Z*) = 0)
17  return   Z*
```

***Variable Neighborhood Search*:** Os algoritmos deste grupo usam diferentes sequências N_k $(1 \leq k \leq k_{max})$ baseadas nos operadores da seção 4.4. Segue uma descrição destes algoritmos.

1. ***VNS-DM-TQ-MT*:** *VNS* explora o espaço de busca usando a sequência de operadores *DM*, *TQ* e *MT*. A heurística Randomized-FIH, baseada no operador MT, executa seis buscas locais diferentes. Em cada busca, a cada iteração é explorado um subconjunto $MT_{k-2} \subseteq MT(Z)$ de vizinhos da vizinhança definida por *MT*, com tamanho $|MT_{k-2}| = (k-2) \times |T|$.
2. ***VNS-MT-TQ*:** *VNS* explora o espaço de busca usando a sequência de operadores *TQ* e *MT*. Os dois procedimentos de busca local utilizados são os mesmos usados no algoritmo *VNS-DM-TQ-MT*.

3. ***VNS-TQ-MT*:** é o algoritmo obtido, invertendo-se a ordem dos procedimentos de busca local utilizados no algoritmo *VNS-MT-TQ*.

O algoritmo 4.12 descreve o pseudocódigo do algoritmo *VNS-DM-TQ-MT*. Os algoritmos *VNS-MT-TQ* e *VNS-TQ-MT* são desenvolvidos de forma análoga, mudando apenas a ordem de chamada aos procedimentos de busca local.

4.6 O benchmark de teste proposto

Esta seção descreve uma base de dados com 102 instâncias de teste para o *PHE* da seção 2.5. As instâncias são provindas de treze escolas públicas de ensino médio brasileiras. O conjunto de instâncias é constituído por 34 casos reais e 68 casos semi-reais[4]. Os 34 casos reais são instâncias provindas das escolas. Os outros 68 casos, dois grupos de 34 instâncias, foram introduzidos de forma a facilitar e a dificultar a resolução do problema. Estes dois últimos grupos foram criados a partir dos casos reais. Em um a resolução foi facilitada e no outro a resolução foi dificultada. Deste modo as 102 instâncias ficaram divididas em três grupos com níveis diferentes de dificuldade de resolução: fácil, moderado e difícil.

A definição destes níveis de dificuldade é baseada no conceito de taxa de esparsidade, equação 4.13, introduzido por Souza et al. (2003).

$$sr = 1 - \frac{\sum_{i=1}^{|L|} \theta + |I|}{|P| \times |D| \times |H|} \tag{4.13}$$

Esta expressão mensura o quanto uma instância do problema está restringida em relação a disponibilidade dos professores.

Quando todos os professores estão disponíveis, $|I| = 0$ e a taxa de esparsidade tem valor máximo dado por $sr_{max} = 1 - \frac{\sum_{i=1}^{|L|} \theta}{|P| \times |D| \times |H|}$ indicando um problema com maior flexibilidade de resolução. Em contrapartida, quando o número de horários de indisponibilidade dos professores é máximo, $\sum_{i=1}^{|L|} \theta + |I| = |P| \times |D| \times |H|$ e a taxa de esparsidade tem valor mínimo dado por $sr_{min} = 0$, configurando um problema de difícil solução por métodos heurísticos.

[4]Instâncias reais com algumas modificações.

Com base neste conceito, os três grupos de instâncias foram denominados: grupo de esparsidade máxima (fácil), grupo de esparsidade real (moderado) e grupo de esparsidade mínima (difícil).

Grupo 1 - Esparsidade real: este grupo contém as 34 instâncias reais cujos conjuntos L (requerimento de aulas) e I (horários indisponíveis) são os originalmente obtidos nas escolas. Estas instâncias são consideradas de dificuldade moderada de resolução por apresentarem taxa de esparsidade $sr_{min} \leq sr \leq sr_{max}$.

Os outros dois grupos são cópias do grupo 1. Nestes grupos o conjunto L de cada instância foi mantido inalterado e o conjunto I foi alterado para introduzir dois níveis diferentes de esparsidade.

Grupo 2 - Esparsidade máxima: este grupo contém as mesmas 34 instâncias do grupo 1. Nas instâncias deste grupo $I = \emptyset$, ou seja, todo professor está disponível em qualquer horário da semana. Estas instâncias são consideradas de fácil resolução devido apresentarem taxa de esparsidade $sr = sr_{max}$.

Grupo 3 - Esparsidade mínima: neste grupo as instâncias tem taxa de esparsidade $sr = sr_{min}$ e o conjunto I é máximo, o que torna estas instâncias de difícil solução. Neste grupo, o conjunto I de cada instância foi definido a partir das melhores soluções das instâncias do grupo 2, que foram obtidas por um algoritmo baseado em *ILS* (Saviniec e Constantino, 2012).

Para entender como este grupo foi criado, considere o seguinte.

Suponha que i_{G1} seja uma instância do grupo 1 e que i_{G2} e i_{G3} sejam as suas cópias nos grupos 2 e 3, respectivamente. Além disso, suponha que Z_{iG2} seja a melhor solução de i_{G2} obtida pelo algoritmo mencionado anteriormente e que Ω_{iG2} seja o conjunto de horários ociosos na agenda dos professores em Z_{iG2}. Então o conjunto I_{iG3} (horários indisponíveis) de i_{G3} é definido por $I_{iG3} = \Omega_{iG2}$.

Note que Z_{iG2} também é uma solução de i_{G3} sem causar nenhuma violação na restrição a_4. Pois se considerarmos Z_{iG2} como solução de i_{G3}, nenhuma aula está alocada em horários indisponíveis (I_{iG3}).

Nosso objetivo neste último grupo, foi construir um conjunto de instâncias com maior dificuldade de resolução, que por outro lado, tivesse boas soluções conhecidas.

A Tabela 4.1 apresenta as características das instâncias: identificador, nome, quantidade de turmas, professores, dias, horários por dia, total de aulas e total de aulas duplas

consecutivas requeridas por professores. A Tabela 4.2 mostra a taxa de esparsidade e o número total de horários indisponíveis por instância em cada grupo.

Tabela 4.1: Características das instâncias

ID	Instância	$\|T\|$	$\|P\|$	$\|D\|$	$\|H\|$	Total de Aulas	Aulas Duplas
1	CL-CEASD-2008-V-A	12	27	5	5	300	132
2	CL-CEASD-2008-V-B	12	27	5	5	300	132
3	CL-CECL-2011-M-A	13	31	5	5	325	144
4	CL-CECL-2011-M-B	13	31	5	5	325	143
5	CL-CECL-2011-N-A	9	28	5	5	225	107
6	CL-CECL-2011-V-A	14	29	5	5	350	164
7	CM-CECM-2011-M	20	51	5	5	500	234
8	CM-CECM-2011-N	8	30	5	5	200	96
9	CM-CECM-2011-V	13	34	5	5	325	142
10	CM-CEDB-2010-N	5	17	5	5	125	60
11	CM-CEUP-2008-V	16	35	5	5	400	192
12	CM-CEUP-2011-M	16	38	5	5	400	192
13	CM-CEUP-2011-N	3	15	5	5	75	36
14	CM-CEUP-2011-V	16	34	5	5	400	169
15	FA-EEF-2011-M	4	12	5	5	100	42
16	JNS-CEDPII-2011-M	8	19	5	5	200	85
17	JNS-CEDPII-2011-V	7	21	5	5	175	73
18	JNS-CEJXXIII-2011-M	5	18	5	5	125	60
19	JNS-CEJXXIII-2011-N	4	15	5	5	100	48
20	JNS-CEJXXIII-2011-V	5	18	5	5	125	60
21	MGA-CEDC-2011-M	19	37	5	5	475	210
22	MGA-CEDC-2011-V	12	31	5	5	300	131
23	MGA-CEGV-2011-M	31	62	5	5	775	352
24	MGA-CEGV-2011-V	32	75	5	5	800	357
25	MGA-CEJXXIII-2010-V	16	35	5	5	400	192
26	MGA-CEVB-2011-M	10	21	5	5	250	108
27	MGA-CEVB-2011-V	9	20	5	5	225	97
28	NE-CESVP-2011-M-A	18	45	5	5	450	212
29	NE-CESVP-2011-M-B	18	44	5	5	450	212
30	NE-CESVP-2011-M-C	18	45	5	5	450	211
31	NE-CESVP-2011-M-D	18	45	5	5	450	211
32	NE-CESVP-2011-V-A	16	44	5	5	400	183
33	NE-CESVP-2011-V-B	16	43	5	5	400	184
34	NE-CESVP-2011-V-C	16	43	5	5	400	182

Tabela 4.2: Esparsidade das instâncias

Grupo 1 - real			**Grupo 2 - máxima**			**Grupo 3 - mínima**		
ID	Hor. Indisp.	*sr*	ID	Hor. Indisp.	*sr*	ID	Hor. Indisp.	*sr*
1	108	0,4	35	0	0,56	69	375	0
2	108	0,4	36	0	0,56	70	375	0
3	23	0,55	37	0	0,58	71	450	0
4	8	0,57	38	0	0,58	72	450	0
5	25	0,64	39	0	0,68	73	475	0
6	21	0,49	40	0	0,52	74	375	0
7	648	0,1	41	0	0,61	75	775	0
8	489	0,08	42	0	0,73	76	550	0
9	455	0,08	43	0	0,62	77	525	0
10	41	0,61	44	0	0,71	78	300	0
11	345	0,15	45	0	0,54	79	475	0
12	498	0,05	46	0	0,58	80	550	0
13	284	0,04	47	0	0,8	81	300	0
14	382	0,08	48	0	0,53	82	450	0
15	160	0,13	49	0	0,67	83	200	0
16	91	0,39	50	0	0,58	84	275	0
17	101	0,47	51	0	0,67	85	350	0
18	50	0,61	52	0	0,72	86	325	0
19	12	0,7	53	0	0,73	87	275	0
20	52	0,61	54	0	0,72	88	325	0
21	382	0,07	55	0	0,49	89	450	0
22	412	0,08	56	0	0,61	90	475	0
23	588	0,12	57	0	0,5	91	775	0
24	857	0,12	58	0	0,57	92	1075	0
25	309	0,19	59	0	0,54	93	475	0
26	167	0,21	60	0	0,52	94	275	0
27	214	0,12	61	0	0,55	95	275	0
28	156	0,46	62	0	0,6	96	675	0
29	167	0,44	63	0	0,59	97	650	0
30	153	0,46	64	0	0,6	98	675	0
31	267	0,36	65	0	0,6	99	675	0
32	181	0,47	66	0	0,64	100	700	0
33	192	0,45	67	0	0,63	101	675	0
34	218	0,43	68	0	0,63	102	675	0

5

Experimentos computacionais

Este capítulo reporta os resultados de experimentos realizados com os algoritmos propostos na seção 4.5, sobre a base de dados da seção 4.6.

Tabela 5.1: Pesos utilizados na primeira fase

Peso	Valor
α_{a_3}	1
α_{a_4}	1

Tabela 5.2: Pesos utilizados na segunda fase

Peso	Valor
α_{a_3}	100.000
α_{a_4}	10.000
α_{b_1}	100
α_{b_2}	25
α_{b_3}	10
α_{b_4}	10
α_{b_5}	10
α_{b_6}	10

Os algoritmos foram implementados em *Visual Basic* 6 e os experimentos realizados no *Windows Server* 2008-R2, rodando em uma máquina virtual *KVM*, configurada para ocupar 30GB de RAM e 50 núcleos de um servidor com 4 processadores *Intel Xeon* E7-4860 (24MB de *Cache* - 2.26 GHz) com o sistema operacional *Linux CentOS* 6.

Na primeira fase de resolução do *PHE* a função objetivo foi ponderada com os pesos da Tabela 5.1. Os pesos utilizados na segunda fase estão dispostos na Tabela 5.2. Estes pesos refletem em geral, a prioridade dada pelas escolas a cada restrição.

5.1 Planejamento e execução dos experimentos

Os experimentos foram organizados e executados da seguinte forma:

Amostras: Todas as instâncias foram processadas 50 vezes por cada algoritmo. O processamento foi realizado em 9 etapas. Em cada etapa um único algoritmo foi experimentado executando-se 50 processos[1] concorrentemente. Todo processo era encarregado de processar as instâncias da base de dados uma vez. Deste modo, ao final das etapas, todas as instâncias haviam sido processadas 50 vezes pelo algoritmo sendo experimentado. Uma vez por cada processo. Para garantir uma distribuição uniforme de processamento entre os processos, durante a execução dos experimentos haviam apenas os 50 processos em execução, com exceção de processos do sistema[2].

Tempo de execução: Na primeira fase o processamento foi executado até que uma solução viável fosse obtida. Na segunda fase o processamento foi executado durante um tempo fixo de 900 segundos (15 minutos).

5.2 Metodologia de análise dos resultados

Na análise dos resultados apresentada na seção 5.3, os algoritmos são avaliados sobre duas variáveis:

Tempo médio de execução: Compara os algoritmos quanto ao tempo médio de execução, em segundos, despendido na primeira fase. Esta análise permite avaliar a capacidade dos algoritmos, em tempo de execução, para encontrar soluções viáveis.

Qualidade das soluções: Compara os algoritmos quanto a qualidade das soluções obtidas na fase de melhoramento. Neste tipo de análise a comparação entre os algoritmos é realizada com base na medida denominada **distância relativa**, definida a seguir:

[1]Diferentes instâncias, em tempo de execução, de um mesmo código executável.

[2]Este procedimento foi seguido para evitar que outros processos de usuário compartilhassem a *CPU* usada pelos 50 processos do experimento. Isto poderia fazer com que alguns processos executassem por menos tempo que outros e desta forma os experimentos estariam propensos a erros grosseiros.

Definição 5.1 (Distância relativa) *Dada uma instância i do PHE. Sejam, $Z_i^{BestKnown}$ a melhor solução conhecida para a instância i e Z_{ijk} uma solução j obtida pelo algoritmo k, para a instância i. Denota-se por distância relativa de Z_{ijk} em relação a $Z_i^{BestKnown}$, a medida dada por:*

$$dr_{ijk} = \frac{f(Z_{ijk})}{f(Z_i^{BestKnown})} \tag{5.1}$$

Esta medida permite mensurar a distância, em proporção, de uma solução de uma dada instância em relação a melhor solução conhecida (limite superior[3]) para ela. Logo, se:

- $dr_{ijk} < 1$: significa que a solução Z_{ijk} é um novo limite superior para a instância i.
- $dr_{ijk} = 1$: significa que a solução Z_{ijk} é uma solução que tem o mesmo valor de função objetivo, que a melhor solução conhecida para a instância i.
- $dr_{ijk} > 1$: significa que a solução Z_{ijk} é uma solução pior que a melhor solução conhecida para a instância i.

Quanto a qualidade das soluções, são avaliados três tipos de distribuições. Seja W um conjunto com n instâncias[4] e Z_{ijk} a solução j $(1 \leq j \leq 50)$ obtida dentre 50 execuções do algoritmo k, para a instância i de W $(1 \leq i \leq n)$. Define-se:

1. **Distribuição das melhores soluções:** Os algoritmos são avaliados por meio da análise da distribuição do conjunto das melhores soluções obtidas para um dado conjunto de instâncias. O conjunto das melhores soluções de um algoritmo k para um conjunto W de instâncias é dado pelo conjunto $MIN_k = \{Z_{ik}^{min} | 1 \leq i \leq n\}$, onde Z_{ik}^{min} é a solução com menor valor de função objetivo, dentre as j's soluções para a instância i, obtidas pelo algoritmo k.
2. **Distribuição das soluções médias:** Os algoritmos são avaliados por meio da análise da distribuição do conjunto das soluções médias obtidas para um dado conjunto de instâncias. O conjunto das soluções médias de um algoritmo k para um conjunto W de instâncias é dado pelo conjunto $MEAN_k = \{Z_{ik}^{mean} | 1 \leq i \leq$

[3]Embora o termo "limite superior" tenha um significado mais abrangente em seu sentido real. De agora em diante, quando nos referirmos ao termo "limite superior" ou "melhor solução conhecida" para uma dada instância, estaremos nos referindo a "solução com menor valor de função objetivo conhecida" para a referida instância.

[4]Por exemplo, o conjunto formado pelas instâncias do grupo esparsidade real.

$n\}$, onde Z_{ik}^{mean} é a solução estimada pela média das 50 soluções obtidas pelo algoritmo k, para a instância i.

3. **Distribuição de todas as soluções:** Os algoritmos são avaliados por meio da análise da distribuição do conjunto de todas as soluções obtidas para um dado conjunto de instâncias. O conjunto de todas as soluções de um algoritmo k para um conjunto W de instâncias é dado pelo conjunto $WHOLE_k = \{Z_{ijk} | 1 \leq i \leq n,\ 1 \leq j \leq 50\}$.

As melhores soluções conhecidas para as instâncias da base de dados utilizada nos experimentos deste capítulo estão tabuladas na Tabela 5.3[5]. Estas soluções foram obtidas, anteriormente, por um algoritmo *ILS* descrito em Saviniec e Constantino (2012), que é um trabalho parcial do projeto de pesquisa desta dissertação.

5.3 Análise dos Resultados

A análise dos resultados está dividida em duas partes. Na primeira parte, seção 5.3.1, é conduzida uma comparação entre as três versões de *ILS* propostas na seção 4.5.1. Na segunda parte, seção 5.3.2, é realizada uma comparação de eficiência entre todos os algoritmos propostos, os da seção 4.5.1 e os da seção 4.5.2.

5.3.1 Algoritmos Mono-Operadores

Esta seção compara a eficiência das três versões de *ILS* propostas na seção 4.5.1, em que cada versão "roda" apenas um dos operadores propostos na seção 4.4. O objetivo desta comparação é analisar o quanto o desempenho de *ILS* é afetado por cada operador.

[5]Para algumas instâncias do grupo 1 (casos reais) estas soluções foram comparadas com a qualidade das soluções utilizadas pelas escolas e se mostraram melhores ou iguais as soluções das escolas. Por outro lado, isto não foi possível para todas as instâncias reais, devido a falta de acesso a todas as soluções feitas pelas escolas.

Tabela 5.3: Melhores soluções conhecidas

Grupo 1		**Grupo 2**		**Grupo 3**	
ID	Melhor Solução	ID	Melhor Solução	ID	Melhor Solução
1	430	35	280	69	260
2	400	36	310	70	270
3	340	37	380	71	290
4	350	38	340	72	280
5	300	39	280	73	270
6	380	40	340	74	290
7	910	41	650	75	540
8	570	42	270	76	220
9	440	43	340	77	300
10	110	44	100	78	100
11	860	45	550	79	450
12	900	46	550	80	480
13	180	47	70	81	70
14	600	48	410	82	330
15	190	49	100	83	60
16	200	50	160	84	120
17	150	51	150	85	110
18	150	52	130	86	130
19	90	53	90	87	80
20	180	54	160	88	160
21	740	55	490	89	420
22	470	56	350	90	260
23	1620	57	1080	91	890
24	1540	58	1030	92	840
25	690	59	560	93	480
26	270	60	210	94	190
27	240	61	180	95	160
28	690	62	680	96	530
29	700	63	600	97	500
30	840	64	630	98	520
31	720	65	700	99	550
32	600	66	510	100	410
33	570	67	570	101	400
34	620	68	570	102	430

Tempo médio de execução na primeira fase

A Figura 5.1 mostra, para as instâncias da base de dados proposta, o tempo médio de execução de cada versão de *ILS* na primeira fase. Conforme especificado na seção 4.6, as instâncias desta base de dados estão classificadas em três grupos com níveis diferentes de esparsidade. Nesta e nas seções que seguem, os gráficos apresentados organizam os resultados dos experimentos por grupo de instância em ordem decrescente de esparsidade, grupo 2 - **esparsidade máxima**, grupo 1 - **esparsidade real** e grupo 3 - **esparsidade mínima**.

Para as instâncias do grupo 2, mostradas na Figura 5.1(a), não houve diferença no tempo de execução dos três algoritmos. O tempo médio despendido por cada algoritmo, em cada instância, foi de menos de 0,5 segundo. Isto se deve ao fato de não haver professores com horários indisponíveis neste grupo, o que facilita a resolução do problema na primeira fase.

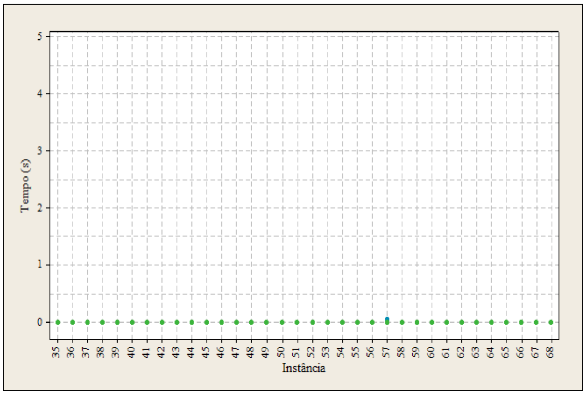
O gráfico da Figura 5.1(b) plota os tempos médios para o grupo 1. Neste grupo de instâncias são observadas diferenças de alguns segundos entre os três algoritmos para algumas instâncias, com os melhores tempos médios obtidos pelo algoritmo *ILS-TQ*. Porém, em termos práticos estas diferenças são insignificantes[6].

Os gráficos da Figura 5.1(c) e da Figura 5.1(d) mostram os resultados para as instâncias do grupo 3. Neste grupo de instâncias, percebemos que o tempo médio do algoritmo *ILS-MT* cresceu de forma desordenada para metade das instâncias do grupo, enquanto o tempo médio de *ILS-DM* cresceu desordenadamente apenas para as instâncias 91 e 92, que são as maiores instâncias. Já o tempo médio de *ILS-TQ* não sofreu grandes alterações.

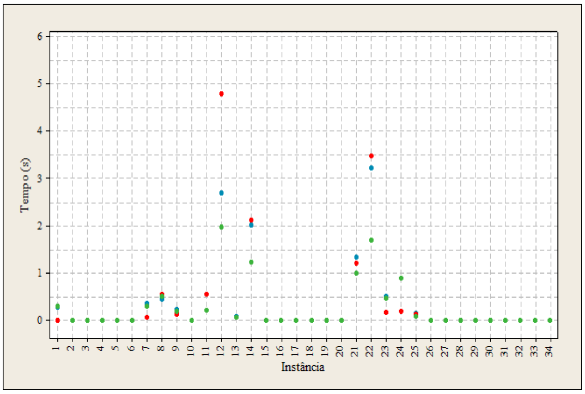
Tendo em vista estes resultados, notamos que quanto a obtenção de soluções na primeira fase, os três algoritmos tiveram maior dificuldade para encontrar soluções conforme a esparsidade dos grupos foram diminuindo. Isto se deve ao fato do número de indisponibilidades aumentar para níveis de esparsidade menores, dificultando a resolução do problema.

Como conclusão desta análise, percebemos que nesta fase de resolução do problema o algoritmo *ILS-MT* não é tão robusto quanto os algoritmos *ILS-DM* e *ILS-TQ*, pois é largamente afetado por instâncias com baixa taxa de esparsidade. Os algoritmos *ILS-DM* e *ILS-TQ* apresentaram bons resultados nos três grupos de instâncias, exceto para a

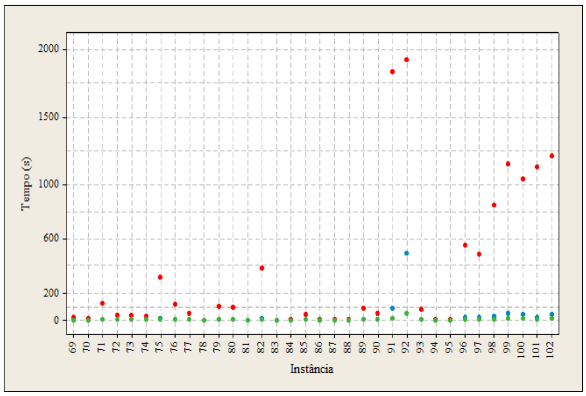
[6]Burke et al. (2010) mencionam que tempo não é um problema crítico em casos reais de escalonamento de horários em instituições de ensino, visto que os quadros de horários são construídos algumas semanas antes de serem usados. Logo, se dois métodos são capazes de produzirem os mesmos resultados com diferença de tempo de apenas algumas unidades de segundos, podemos considerar tal diferença como insignificante em termos práticos.



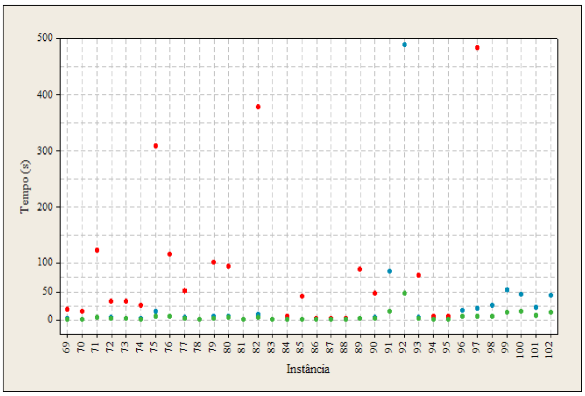
(a) Grupo 2



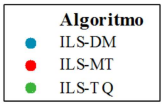
(b) Grupo 1



(c) Grupo 3



(d) Grupo 3 - gráfico ampliado



(e) Legenda.

Figura 5.1: Tempo médio de execução na primeira fase

instância 92 do grupo 3, em que *ILS-DM* apresentou um tempo médio muito ruim. Logo, dentre os três algoritmos testados, o mais adequado para esta fase do problema é o algoritmo *ILS-TQ*.

Distribuição das melhores soluções

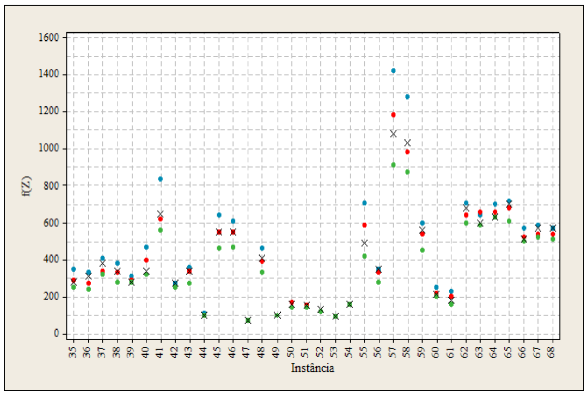
A Figura 5.2 esboça a distribuição das melhores soluções de cada algoritmo. Os gráficos no lado esquerdo da Figura 5.2 plotam as melhores soluções de cada algoritmo por instância. Os gráficos *boxplot's* nas figuras da direita, plotam as mesmas soluções apresentadas nas figuras da esquerda. Os valores da soluções plotadas nestes gráficos estão agora, representados em **distância relativa**. Nos gráficos das figuras à esquerda, as melhores soluções conhecidas para as instâncias, Tabela 5.3, estão plotadas por pontos representados pelo símbolo "×". Nos gráficos da direita, os valores destas soluções estão situados sobre a linha vertical que cruza o eixo das abcissas em 1.

O gráfico da Figura 5.2(b) mostra que *ILS-TQ* encontrou soluções com função objetivo menor ou igual a melhor solução conhecida, para todas as instâncias do grupo 2. *ILS-MT* e *ILS-DM* também melhoraram os limites superiores de algumas instâncias, entretanto em menor quantidade que *ILS-TQ*. Estas observações podem ser facilmente visualizadas, instância por instância na Figura 5.2(a).

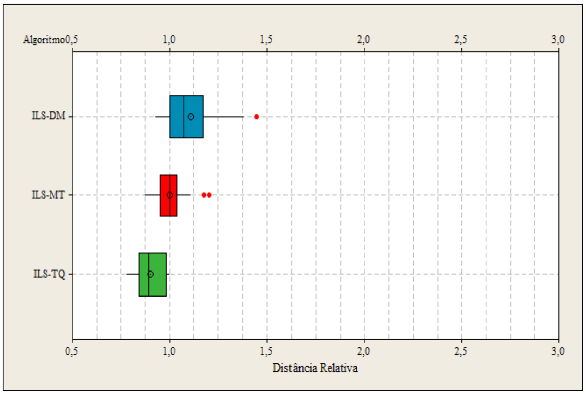
No grupo 1 observamos, pela Figura 5.2(d), que o algoritmo *ILS-TQ* também foi capaz de encontrar soluções com função objetivo menor ou igual a melhor solução conhecida, para todas as instâncias do grupo. E da mesma forma que no grupo 2, *ILS-MT* e *ILS-DM* também encontraram novos limites superiores para as instâncias, mas em menor proporção que *ILS-TQ*.

No grupo 3, Figura 5.2(f), nenhum dos três algoritmos foi capaz de melhorar os limites superiores conhecidos para as instâncias. As melhores soluções encontradas por *ILS-TQ* são entre 1 e 2,8 vezes maiores que os limites superiores. As soluções encontradas por *ILS-MT* e *ILS-DM*, são piores ainda. 75% das melhores soluções encontradas por *ILS-MT* são 3,9 vezes maiores que os limites superiores, ao passo que 75% das melhores soluções de *ILS-DM* são 5,3 vezes maiores que os limites superiores das instâncias.

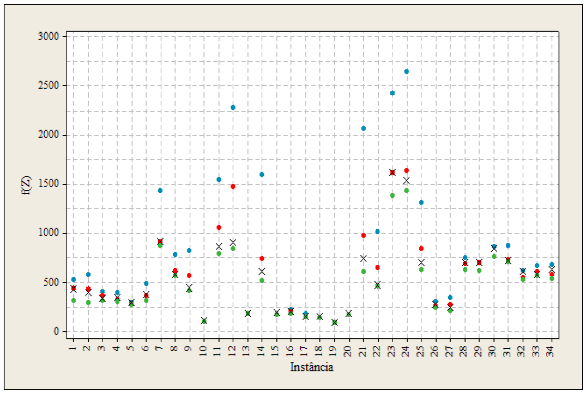
De modo geral, em relação a capacidade de obter melhores soluções, *ILS-TQ* foi o algoritmo que apresentou os melhores resultados em todos os grupos de instâncias, enquanto *ILS-MT* obteve uma leve diferença de desempenho em relação a *ILS-DM*.



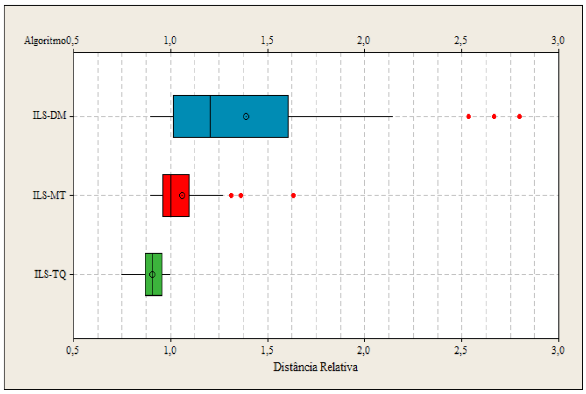
(a) Grupo 2 - melhores soluções por instância



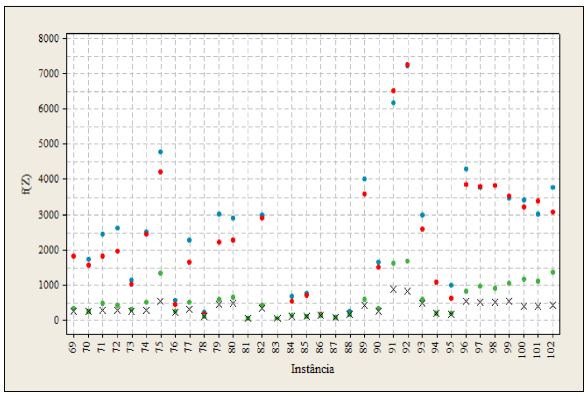
(b) Grupo 2 - conjunto das melhores soluções representadas em distância relativa



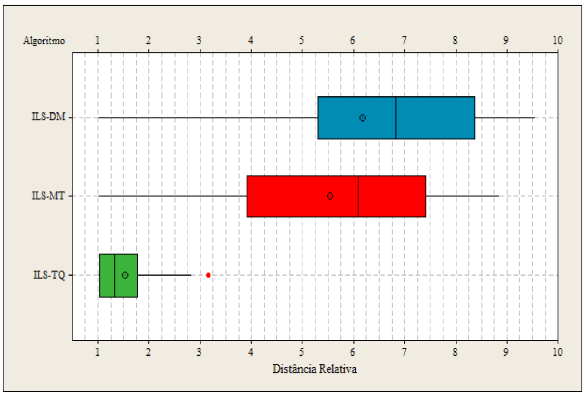
(c) Grupo 1 - melhores soluções por instância



(d) Grupo 1 - conjunto das melhores soluções representadas em distância relativa



(e) Grupo 3 - melhores soluções por instância



(f) Grupo 3 - conjunto das melhores soluções representadas em distância relativa

Figura 5.2: Distribuição das melhores soluções

Distribuição das soluções médias

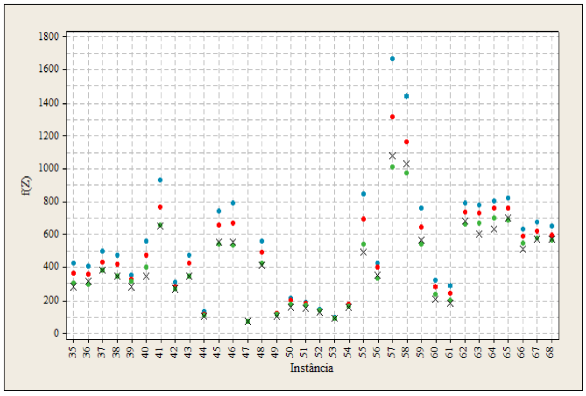
A Figura 5.3 exibe a distribuição das soluções médias de cada algoritmo. Como na seção anterior, os gráficos situados no lado esquerdo da Figura 5.3 plotam para cada instância, as soluções médias de cada algoritmo e os limites superiores que estão representados por símbolos "×". Os gráficos no lado direito esboçam estas mesmas soluções representadas em **distância relativa**.

No grupo 2, Figura 5.3(b), as soluções médias de *ILS-TQ* tem valores de função objetivo entre 0,93 e 1,17 vezes os valores dos limites superiores, apresentando para mais de 25% das instâncias, soluções médias menores que os limites superiores. As soluções médias de *ILS-MT* variam entre 1 e 1,41 vezes os valores dos limites superiores, enquanto as médias de *ILS-DM* estão entre 1 e 1,72.

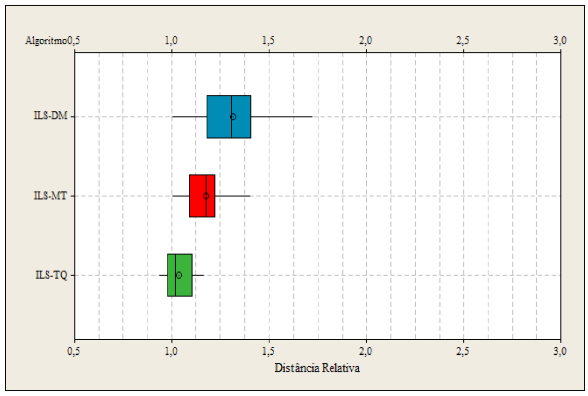
No grupo 1, Figura 5.3(d), as soluções médias de *ILS-TQ* tem no mínimo 0,91 e no máximo 1,22 vezes os valores dos limites superiores. Para aproximadamente 25% das instâncias, as soluções médias são menores ou iguais que os limites superiores. As soluções médias de *ILS-MT* são no mínimo 1,02 e no máximo 1,7 vezes maiores que os limites superiores. As médias de *ILS-DM* são no mínimo 1,07 e no máximo 2,76 vezes maiores que os limites superiores.

No grupo 3, Figura 5.3(f), as piores médias de *ILS-TQ* são no máximo 5,08 vezes maiores que os limites superiores. Já os algoritmos *ILS-MT* e *ILS-DM* apresentaram médias muito ruins, no mínimo 1,65 e no máximo 14 vezes maiores que os limites superiores.

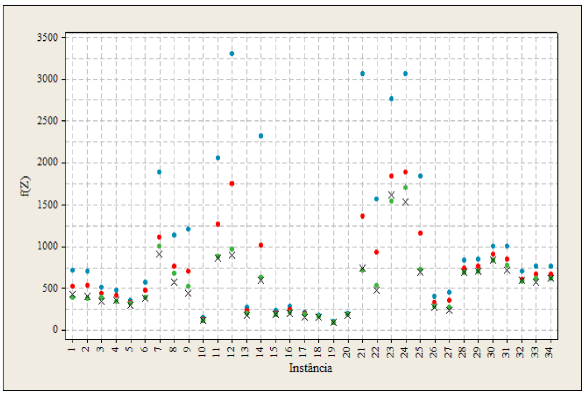
Como conclusão em relação as soluções médias, podemos observar que para instâncias do grupo 2, cujos professores não possuem indisponibilidade de horários, os três algoritmos funcionam igualmente bem. A medida que o número de indisponibilidades aumenta do grupo 2 para o grupo 1, e depois para o grupo 3. A qualidade das soluções médias de *ILS-TQ* são deterioradas lentamente. Enquanto que para os algoritmos *ILS-MT* e *ILS-DM* as solução médias são deterioradas rapidamente, tornando-os extremamente ineficientes para resolver instâncias com baixa taxa de esparsidade.



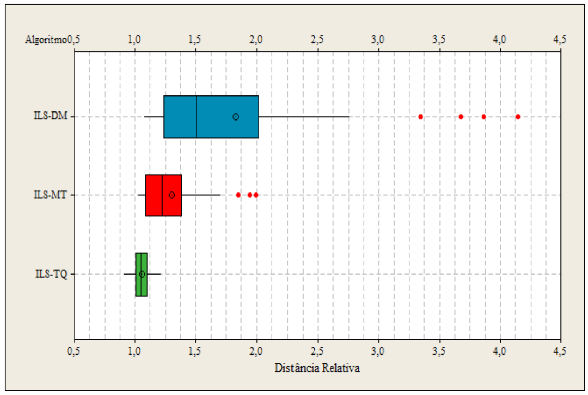
(a) Grupo 2 - soluções médias por instância



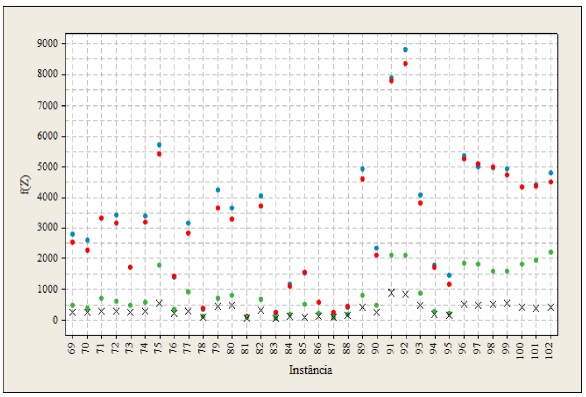
(b) Grupo 2 - conjunto das soluções médias representadas em distância relativa



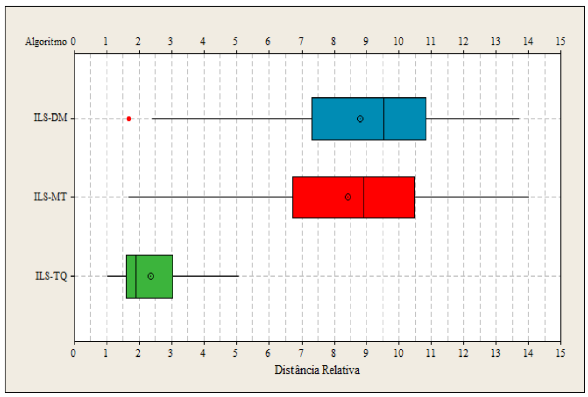
(c) Grupo 1 - soluções médias por instância



(d) Grupo 1 - conjunto das soluções médias representadas em distância relativa



(e) Grupo 3 - soluções médias por instância

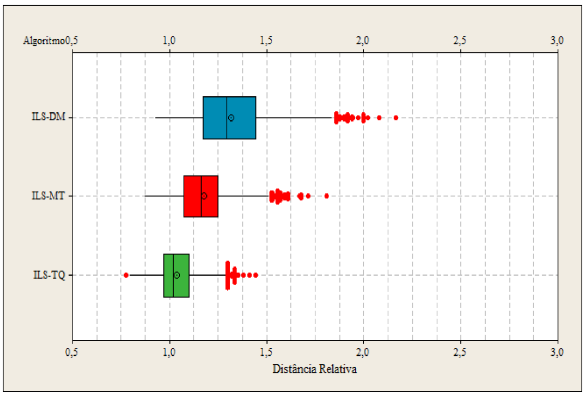


(f) Grupo 3 - conjunto das soluções médias representadas em distância relativa

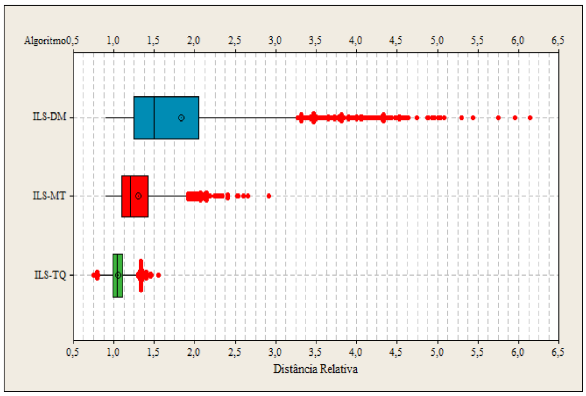
Figura 5.3: Distribuição das soluções médias

Distribuição de todas as soluções

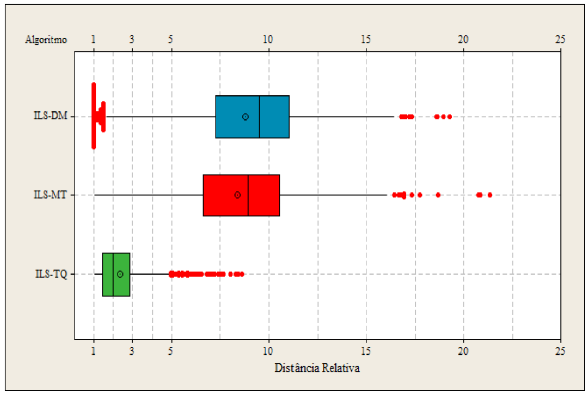
A Figura 5.4 apresenta a distribuição de todas as soluções de cada algoritmo. Nesta distribuição, a sequência de gráficos dispostas na Figura 5.4(a), Figura 5.4(b) e Figura 5.4(c) mostram a diferença de eficiência entre os três algoritmos no quesito qualidade das soluções. As distribuições das soluções plotadas nos gráficos mostram que conforme a taxa de esparsidade das instâncias diminui, a qualidade das soluções geradas pelos algoritmos *ILS-MT* e *ILS-DM* se tornam muito ruins, enquanto *ILS-TQ* continua gerando soluções boas.



(a) Grupo 2 - conjunto de todas as soluções representadas em distância relativa



(b) Grupo 1 - conjunto de todas as soluções representadas em distância relativa



(c) Grupo 3 - conjunto de todas as soluções representadas em distância relativa

Figura 5.4: Distribuição de todas as soluções

No grupo 2, os três algoritmos produziram soluções de boa qualidade para todas as instâncias. Neste grupo, *ILS-DM* foi o algoritmo que gerou os piores resultados. Mesmo

assim, as soluções atípicas ruins deste algoritmo não estão tão distantes dos limites superiores, pois possuem função objetivo no máximo 2,17 vezes maior que os limites superiores.

No grupo 1, cujas instâncias são casos reais, *ILS-TQ* permaneceu gerando soluções de boa qualidade para todas as instâncias. *ILS-MT* perdeu um pouco de eficiência, embora a distribuição de suas soluções continuaram sendo de boa qualidade, menores que 1,42 vezes os limites superiores para 75% das soluções e menor que 1,2 para 50%. Já *ILS-DM* se tornou um algoritmo instável, pois os 25% das soluções acima do terceiro quartil estão muito dispersas e longe dos limites superiores das instâncias. E além disso, há muitas soluções atípicas ruins.

No grupo 3, cujas instâncias possuem taxa de esparsidade igual a zero. Podemos ver que *ILS-MT* e *ILS-DM* são totalmente ineficientes para resolver estas instâncias. Pois 75% das soluções destes algoritmos possuem função objetivo maiores que 6,6 vezes os limites superiores e somente algumas soluções são boas. Já *ILS-TQ* possui uma distribuição com 75% de soluções que são no máximo 2,85 vezes maiores que os limites superiores.

Como conclusão, *ILS-DM* apresentou bons resultados apenas para as instâncias com professores sem indisponibilidade de horários no grupo 2. *ILS-MT* apresentou bons resultados para o grupo 2 e resultados razoáveis para os casos reais do grupo 1. Já *ILS-TQ* apresentou bons resultados para os grupos 1 e 2, e resultados razoáveis para o grupo 3, onde os professores possuem número máximo de horários indisponíveis.

Considerações finais

Como observado nas seções anteriores, quanto ao tempo médio de execução na primeira fase, o desempenho do algoritmo *ILS* sofreu maiores variações apenas no grupo de instâncias com esparsidade mínima. Neste grupo o desempenho de *ILS* foi largamente diminuído com uso do operador *MT*. Com o operador *DM* houve pouca perda de desempenho e com o operador *TQ* foram observados bons resultados.

Quanto a qualidade das soluções, o desempenho de *ILS* variou significativamente dependendo do operador empregado. O operador *TQ* foi o operador que possibilitou os melhores resultados em todos os grupos de instâncias. Pois em geral, as soluções de *ILS-TQ* são melhores e estão menos dispersas que as soluções dos demais algoritmos. Em outras palavras, *ILS-TQ* gera as melhores soluções e estas possuem sempre a mesma qualidade. O fato das soluções de *ILS-TQ* estarem pouco dispersas, mostra que a combinação de *ILS* com o operador *TQ*, gera um algoritmo estável e robusto. Tal comportamento não é observado usando os demais operadores.

Em linhas gerais, nesta etapa de análise concluímos que diferentes operadores podem afetar significativamente o desempenho de uma boa estratégia de busca. Portanto, operadores de vizinhança são um ponto crítico no desenvolvimento de bons algoritmos de busca local e devem ser bem definidos para que se possa obter o melhor desempenho.

5.3.2 Algoritmos Poli-Operadores

Na seção 5.3.1 avaliamos três versões de *ILS* e concluímos que o método pode apresentar melhor ou pior desempenho dependendo do operador empregado. Nesta seção, conduziremos o mesmo tipo de análise desenvolvida na seção 5.3.1, de modo a avaliar os demais algoritmos da seção 4.5.2. O objetivo agora, é confrontar os resultados obtidos por todos os algoritmos propostos, de forma a identificar as versões mais eficientes para resolução do *PHE*.

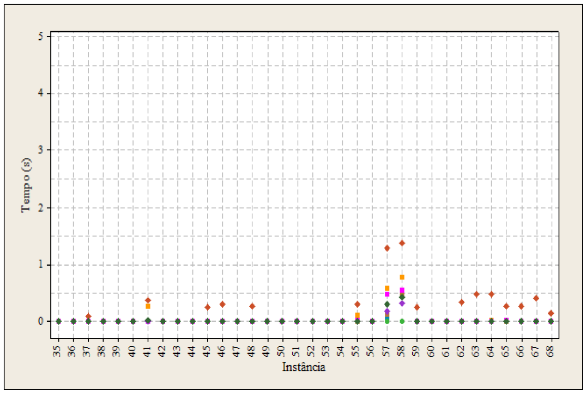
Tempo médio de execução na primeira fase

A Figura 5.5 mostra o tempo médio de execução de cada algoritmo para obtenção de soluções na primeira fase.

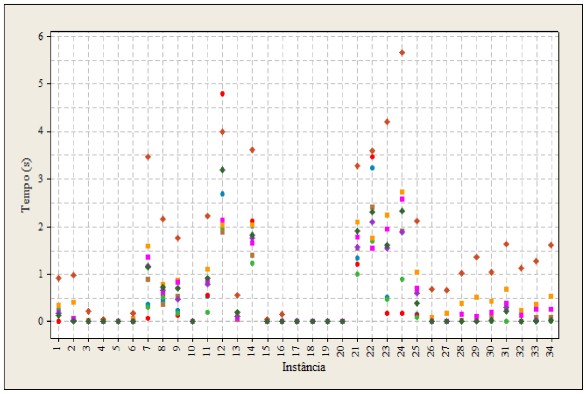
Nos grupos 1 e 2 pode ser observado que não há diferenças significativas, em termos práticos, entre os algoritmos. Pois todos executaram em tempo médio menor que 1,5 segundos para as instâncias do grupo 2, Figura 5.5(a), e em tempo médio inferior a 6 segundos para as instâncias do grupo 1, Figura 5.5(b).

No grupo 3, Figura 5.5(c), *ILS-MT* foi o algoritmo que apresentou os piores tempos médios para a maioria das instâncias, chegando a tempos próximos de 2000 segundos para as instâncias 91 e 92. *ILS-DM* apresentou um tempo médio próximo de 500 segundos para resolver a instância 92 na primeira fase. Este tempo pode ser considerado discrepante em relação aos tempos gastos nas demais instâncias. O restante dos algoritmos executaram em tempo médio inferior a 110 segundos.

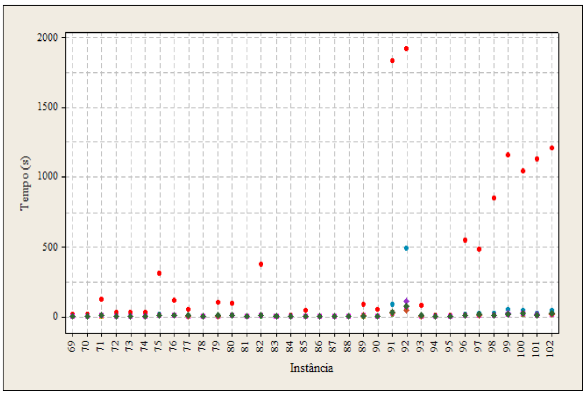
De forma mais detalhada, excluindo da análise os algoritmos *ILS-MT*, *ILS-DM* e a instância 92, que é a maior instância, vemos que para o restante das instâncias os demais algoritmos executaram em tempo médio menor que 30 segundos. Este fato é melhor visualizado no gráfico da Figura 5.5(d).



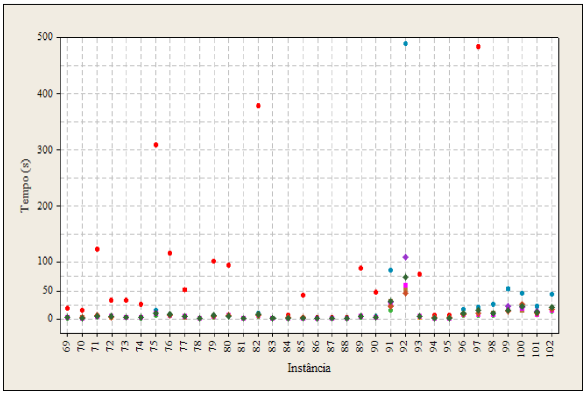
(a) Grupo 2



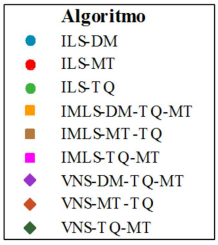
(b) Grupo 1



(c) Grupo 3



(d) Grupo 3 - gráfico ampliado



(e) Legenda.

Figura 5.5: Tempo médio de execução na primeira fase

Distribuição das melhores soluções

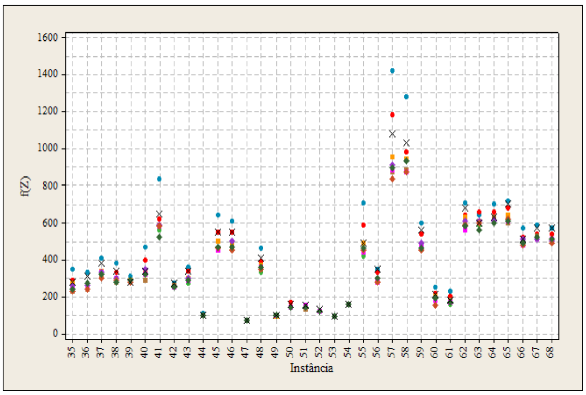
De forma semelhante a apresentação dos dados realizada na Figura 5.2 da seção 5.3.1, para as três versões de *ILS*, a Figura 5.6 exibe a distribuição das melhores soluções para os nove algoritmos analisados nesta seção.

No grupo 2, a Figura 5.6(b) mostra que *ILS-TQ*, *IMLS-TQ-MT* e *VNS-MT-TQ* foram os algoritmos que conseguiram novos limites superiores para um maior percentual de instâncias. Para algumas instâncias localizadas abaixo do primeiro quartil, *VNS-MT-TQ* encontrou as soluções com os melhores valores de função objetivo.

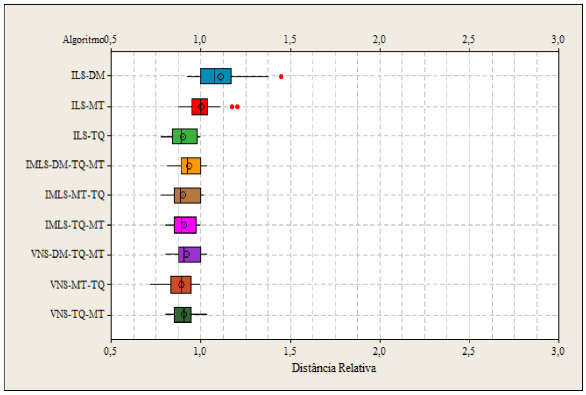
No grupo 1, a Figura 5.6(d) mostra que os algoritmos *ILS-TQ*, *VNS-MT-TQ* e *VNS-TQ-MT* conseguiram melhores limites superiores para um maior percentual de instâncias. Entretanto, com exceção de *ILS-DM* e *ILS-MT*, os demais algoritmos também apresentaram bons resultados.

No grupo 3, Figura 5.6(f), *ILS-TQ*, *IMLS-TQ-MT* e *VNS-MT-TQ* foram os algoritmos que tiveram as distribuições com melhor qualidade. Entretanto, nenhum dos algoritmos foi capaz de melhorar os limites superiores das instâncias. Isto se deve ao fato de que os limites superiores mencionados na Tabela 5.3, para este grupo de instâncias, foram obtidos sem levar em consideração as indisponibilidades dos professores, que foram definidas posteriormente, de forma que estes limites continuassem válidos. Isto foi mencionado na definição do grupo 3, na seção 4.6.

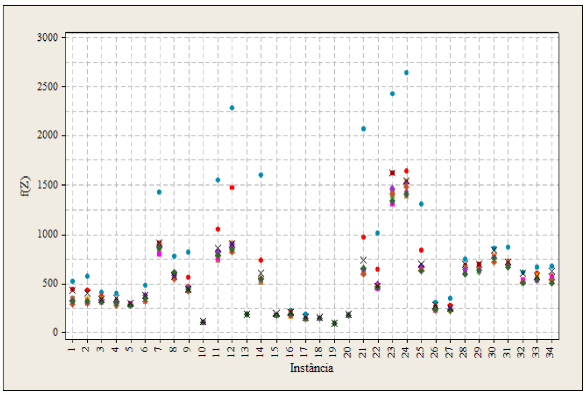
Como conclusão desta análise, as versões de *ILS* e *VNS* baseadas nas combinações *MT-TQ* e *TQ-MT* são os algoritmos mais indicados para encontrar limites superiores para instâncias do PHE, cujas soluções ótimas (mínimos globais) que desejamos conhecer, sejam praticamente impossíveis de serem obtidos por métodos exatos.



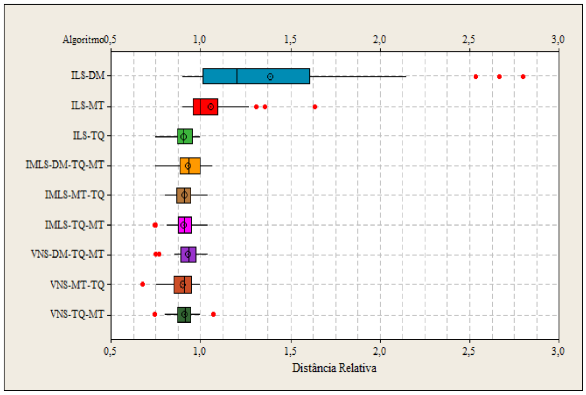
(a) Grupo 2 - melhores soluções por instância



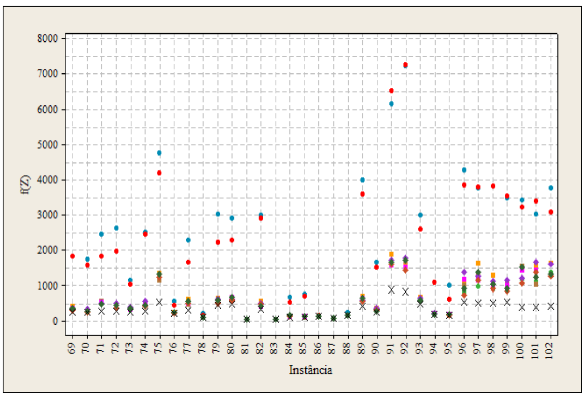
(b) Grupo 2 - conjunto das melhores soluções representadas em distância relativa



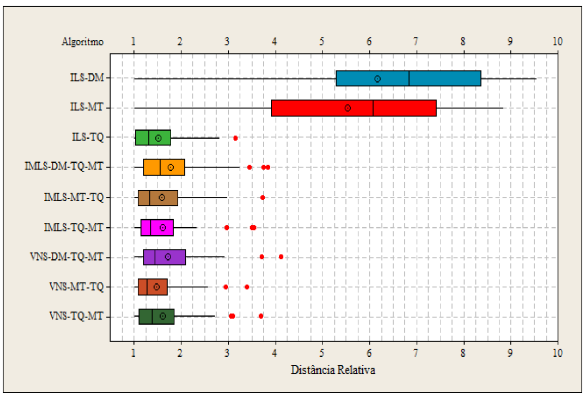
(c) Grupo 1 - melhores soluções por instância



(d) Grupo 1 - conjunto das melhores soluções representadas em distância relativa



(e) Grupo 3 - melhores soluções por instância



(f) Grupo 3 - conjunto das melhores soluções representadas em distância relativa

Figura 5.6: Distribuição das melhores soluções

Distribuição das soluções médias

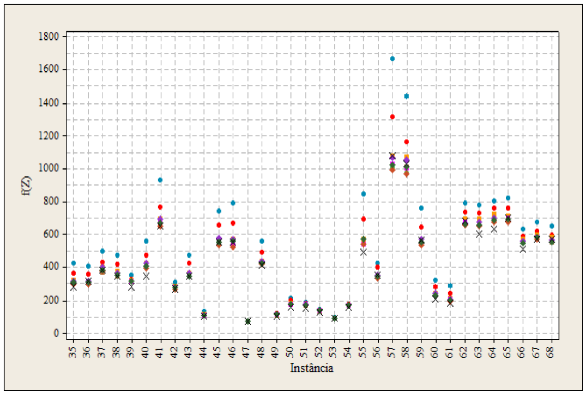
A Figura 5.7 esboça a distribuição das soluções médias de cada algoritmo.

O gráfico da Figura 5.7(b) mostra que os algoritmos *ILS-TQ*, *IMLS-MT-TQ*, *IMLS-TQ-MT*, *VNS-MT-TQ* e *VNS-TQ-MT* foram os algoritmos que apresentaram as melhores soluções médias no grupo 2.

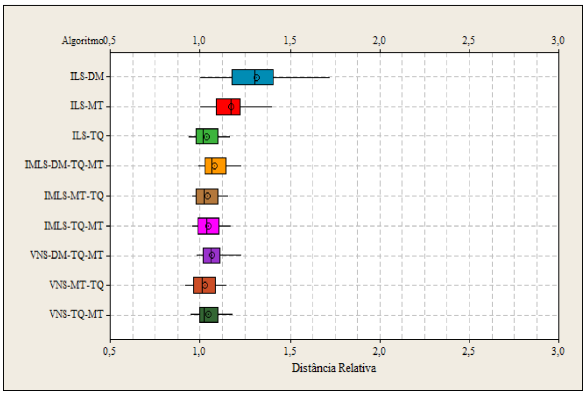
No grupo 1, pelo gráfico da Figura 5.7(d), os melhores resultados médios foram obtidos por *ILS-TQ*, *IMLS-MT-TQ* e *IMLS-TQ-MT* que apresentaram distribuições mais compactas em relação aos demais algoritmos. Além disso, as versões de *VNS* geraram soluções médias atípicas muito ruins para algumas instâncias.

No grupo 3, pelo gráfico da Figura 5.7(f), as melhores soluções médias foram obtidas pelo algoritmo *ILS-TQ*.

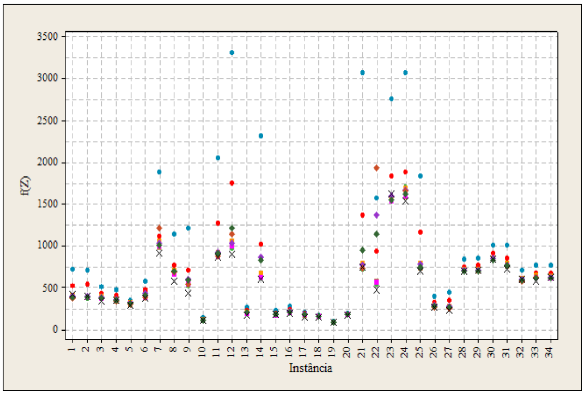
Como conclusão, notamos que em termos médios os algoritmos *ILS-TQ*, *IMLS-MT-TQ*, *IMLS-TQ-MT*, *VNS-MT-TQ* e *VNS-TQ-MT* apresentaram bons resultados e acredita-se que não existam diferenças significativas entre eles.



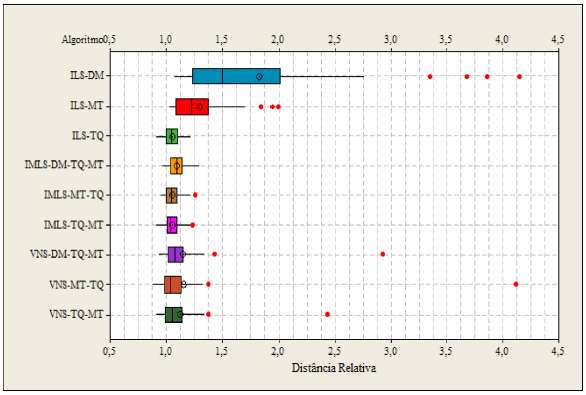
(a) Grupo 2 - soluções médias por instância



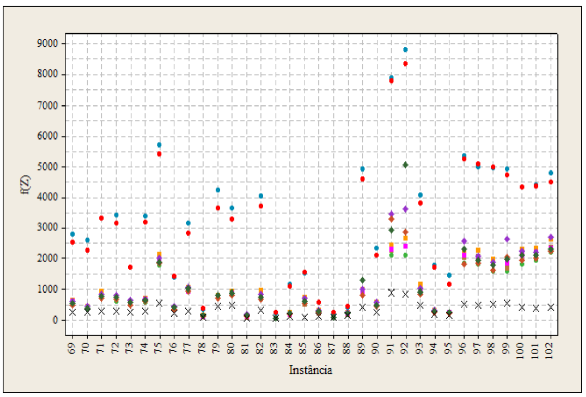
(b) Grupo 2 - conjunto das soluções médias representadas em distância relativa



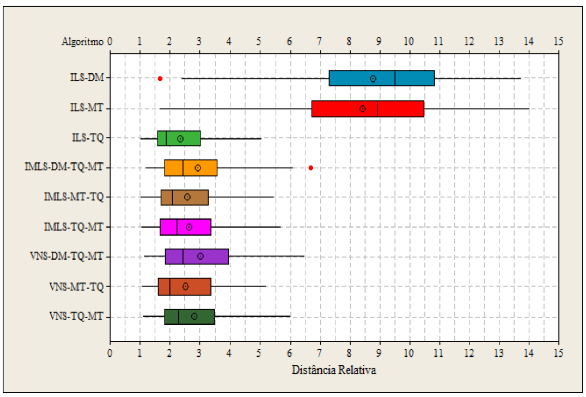
(c) Grupo 1 - soluções médias por instância



(d) Grupo 1 - conjunto das soluções médias representadas em distância relativa



(e) Grupo 3 - soluções médias por instância



(f) Grupo 3 - conjunto das soluções médias representadas em distância relativa

Figura 5.7: Distribuição das soluções médias

Distribuição de todas as soluções

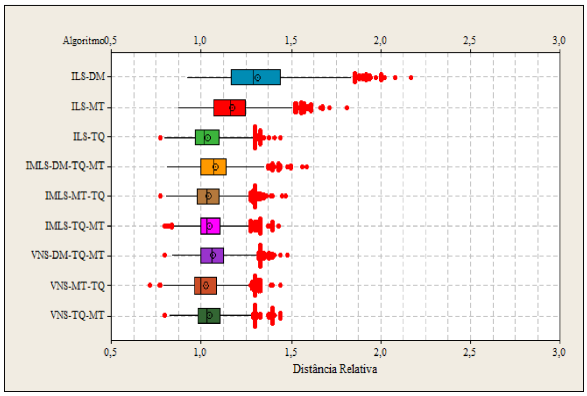
Nesta seção são analisadas as distribuições de todas as soluções, Figura 5.8, dos nove algoritmos estudados.

Pelo gráfico da Figura 5.8(a), bons resultados foram obtidos por todos os algoritmos no grupo 2, com exceção de *ILS-DM* e *ILS-MT*. Pode ser notado ainda, pelas distribuições, um pequeno ganho de desempenho pelos algoritmos *ILS-TQ*, *IMLS-MT-TQ* e *VNS-MT-TQ* em relação aos algoritmos *IMLS-DM-TQ-MT*, *IMLS-TQ-MT*, *VNS-DM-TQ-MT* e *VNS-TQ-MT*.

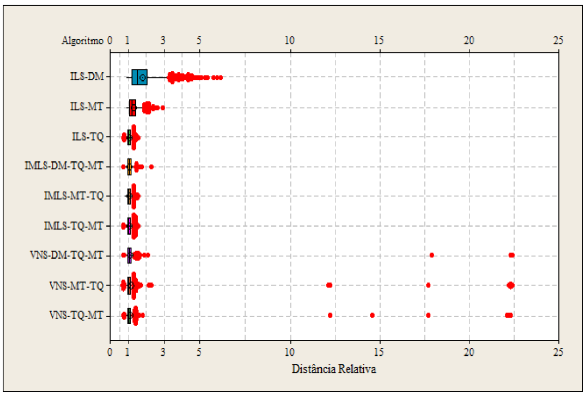
No grupo 1, Figura 5.8(b) e Figura 5.8(c), os algoritmos *ILS-TQ*, *IMLS-MT-TQ*, *IMLS-TQ-MT*, *VNS-MT-TQ* e *VNS-TQ-MT* apresentaram bons resultados. Embora poder ser notado que *ILS-TQ* apresenta uma distribuição, de soluções atípicas ruins, melhor em relação aos demais. Além disso, notamos que embora os algoritmos baseados em *VNS* apresentem bons resultados, eles são um tanto instáveis, devido apresentarem soluções atípicas muito ruins para algumas instâncias.

No grupo 3, como pode ser observado nos gráficos da Figura 5.8(d) e Figura 5.8(e), *ILS-TQ* apresentou os melhores resultados, visto que a distribuição de suas soluções é mais compacta em relação aos demais algoritmos.

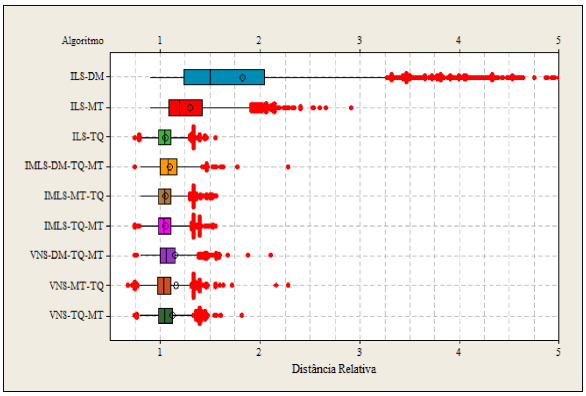
Como conclusão da análise da distribuição de todas as soluções dos algoritmos, *ILS-TQ* é o algoritmo mais indicado para os três grupos de instâncias.



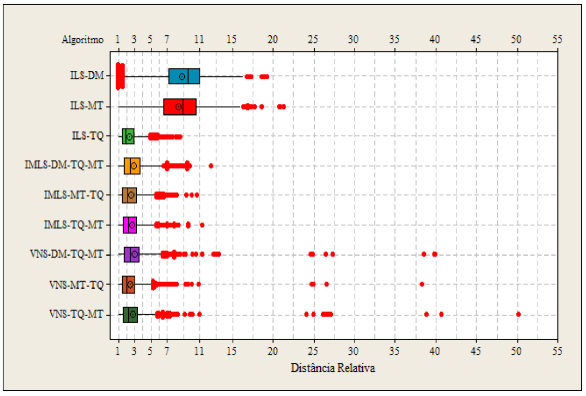
(a) Grupo 2 - conjunto de todas as soluções representadas em distância relativa



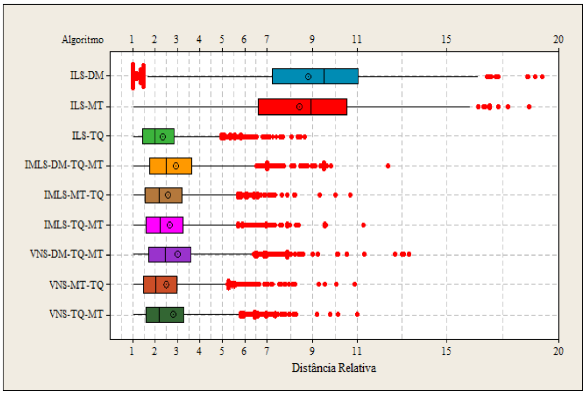
(b) Grupo 1 - conjunto de todas as soluções representadas em distância relativa



(c) Grupo 1 - conjunto de todas as soluções representadas em distância relativa - gráfico ampliado



(d) Grupo 3 - conjunto de todas as soluções representadas em distância relativa



(e) Grupo 3 - conjunto de todas as soluções representadas em distância relativa - gráfico ampliado

Figura 5.8: Distribuição de todas as soluções

Considerações finais

Em relação a comparação de todos os algoritmos quanto ao tempo médio de execução para resolver o *PHE* na primeira fase, nos níveis **esparsidade máxima** e **real** os nove algoritmos são adequados. Já no nível **esparsidade mínima**, *ILS-DM* e *ILS-MT* não apresentam resultados desejáveis.

Quanto a qualidade das soluções, *ILS-DM* e *ILS-MT* apresentaram desempenho muito ruim e não seriam de grande utilidade prática para resolver o PHE. Os demais algoritmos podem ser considerados eficientes, pois apresentaram bom desempenho e seriam úteis em aplicações práticas. Entretanto, observamos dois fatores importantes entre os algoritmos considerados eficientes. Primeiro, no que se refere a distribuição de soluções não atípicas, notamos que *IMLS-DM-TQ-MT* e *VNS-DM-TQ-MT* apresentaram desempenho um pouco pior que os demais algoritmos. Segundo, em relação as soluções atípicas ruins, notamos que os algoritmos baseados em *VNS* são menos estáveis que os algoritmos baseados em *ILS*. Isto pode ser observado nos gráficos da Figura 5.8. Por estes gráficos, todas as versões baseadas em *VNS* apresentam soluções atípicas ruins muito distantes dos limites superiores para algumas instâncias, tanto no grupo 1 quanto no grupo 3. Nossa hipótese para a causa da instabilidade de *VNS* é o fato de *VNS* ter explorado seis vizinhanças de tamanho diferentes usando o operador *MT*. Talvez isto pode ter causado a exploração de uma mesma região várias vezes e consequentemente diminuído a eficiência de *VNS*.

Como conclusão, para variantes do *PHE* semelhantes ao problema tratado nesta dissertação, os algoritmos mais recomendados seriam as versões baseadas em *ILS*, *ILS-TQ*, *IMLS-MT-TQ* e *IMLS-TQ-MT*.

6

Conclusão

Nesta dissertação, abordamos uma categoria do problema de escalonamento de horários em instituições de ensino denominada **problema de horários em escolas**. Em revisão a literatura, identificamos que esta categoria possui diversas variantes e que sua exploração no meio científico tem acontecido em menor volume que nas demais categorias. Entre os motivos apontados pelos pesquisadores para existência de um menor volume de estudos nesta área, destaca-se a escassez de bases de dados que possam ser compartilhadas entre os pesquisadores da área, para comparação de estudos. Pois uma das vantagens da existência de tais bases, é que elas contribuem para que métodos cada vez mais eficientes sejam desenvolvidos para resolver o problema.

No trabalho que desenvolvemos, estudamos uma variante do **problema de horários em escolas** encontrada em treze escolas de ensino médio da rede pública de ensino brasileira. Com a execução das propostas sugeridas, os seguintes resultados e contribuições foram alcançados.

Primeiro, construímos uma base de dados com 102 instâncias de teste para o problema. As instâncias foram definidas para formar três níveis de dificuldade de resolução do problema. Esta base de dados nos permitiu avaliar a eficiência de algoritmos para a resolução do problema estudado, em diferentes níveis de dificuldade.

Segundo, projetamos dois novos operadores de uso dedicado para o problema. Os dois operadores foram avaliados juntamente com um terceiro operador clássico e de uso geral, disponível na literatura. Nesta etapa foram propostos três algoritmos *ILS*, um para rodar com cada operador. Nossos experimentos mostraram que diferentes operadores podem proporcionar a um algoritmo de busca obter diferentes níveis de desempenho.

Isto confirma o que muitos pesquisadores mencionam na literatura. Além disso, os dois operadores que projetamos proporcionaram ao método *ILS* obter melhor desempenho na resolução de 100% das instâncias do problema, do que executando com o operador clássico da literatura.

Terceiro, desenvolvemos mais seis algoritmos para o problema. Três baseados no método *ILS* e três em *VNS*. Nesta etapa, assim como na anterior, não propomos qualquer alterações nas estratégias de busca destes métodos, pois usamos estruturas básicas já apresentadas na literatura. Nossa proposta consistiu em combinar os três operadores testados anteriormente, para projetar diferentes heurísticas de busca local para usar com estes métodos. Nos experimentos realizados sobre a base de dados proposta, avaliamos todos os nove algoritmos propostos e identificamos que entre eles, três baseados em *ILS* e dois baseados em *VNS*, foram os que apresentaram os melhores resultados. Contudo, observamos uma deficiência dos algoritmos baseados em *VNS* se comparado aos baseados em *ILS*. Pois para algumas instâncias, os algoritmos baseados em *VNS* apresentaram soluções atípicas ruins, muito distante das melhores soluções conhecidas. Tal comportamento também foi observado para os algoritmos baseados em *ILS*. No entanto, em menor escala e apenas para o grupo de instâncias com maior dificuldade de resolução.

Como conclusão geral, podemos dizer que os resultados obtidos neste trabalho apontam para uma promissora área de pesquisa em otimização combinatória, que é a investigação e desenvolvimento de operadores de vizinhança para algoritmos de busca local.

6.1 Trabalhos futuros

Em trabalhos futuros, pretendemos avaliar os algoritmos propostos nesta dissertação, sobre um tempo maior de computação para averiguar possíveis diferenças de eficiência entre eles. Além disso, investigaremos em mais detalhes, as causas que levaram *VNS* a apresentar instabilidade. Também pretendemos usar os operadores propostos nesta dissertação, para desenvolver algoritmos de busca local baseados em outras metaheurísticas como Busca tabu, *Simulated Annealing* e *Great Deluge* para aplicar ao *PHE*.

Atualmente estamos investigando a generalização do operador *TQ* para obter um operador capaz de explorar vizinhanças mais complexas. Pois foi observado que o grafo do modelo desenvolvido para este operador, permite que os vértices tenham mais que duas aulas. Deste modo, usando uma estratégia similar a usada no operador *billiard movement* de Kaneko et al. (1999) (Figura 3.4 página 32), para deslocar três ou mais aulas de uma turma entre horários quaisquer, muitos conflitos na restrição a_3 podem ser gerados.

Portanto, o operador TQ generalizado permitiria que estes conflitos fossem reparados por meio de um posterior deslocamento das aulas de outras turmas em conflito.

Outro trabalho que pretendemos desenvolver, é utilizar os dois operadores propostos, em algoritmos de busca local, para resolver instâncias do **Problema de Horários de Cursos** e do **Problema de Alocação de Salas**, pois foi observado que estes operadores podem ser adaptados para aplicar nestes dois problemas.

REFERÊNCIAS

Abdullah, S.; Ahmadi, S.; Burke, E.; Dror, M. Investigating ahuja–orlin´s large neighbourhood search approach for examination timetabling. *OR Spectrum*, v. 29, n. 2, p. 351–372, 2007.

Abramson, D.; Abela, J.; of Information Technology, C. A. D. *A parallel genetic algorithm for solving the school timetabling problem.* Citeseer, 1991.

Ahuja, R.; Orlin, J.; Sharma, D. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, v. 91, n. 1, p. 71–97, 2001.

Aladag, C.; Hocaoglu, G.; Basaran, M. The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem. *Expert Systems with Applications*, v. 36, n. 10, p. 12349–12356, 2009.

Avella, P.; D´Auria, B.; Salerno, S.; Vasil´ev, I. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, v. 13, n. 6, p. 543–556, 2007.

Beligiannis, G.; Moschopoulos, C.; Kaperonis, G.; Likothanassis, S. Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Research*, v. 35, n. 4, p. 1265–1280, 2008.

Bello, G.; Rangel, M.; Boeres, M. An approach for the class/teacher timetabling problem using graph coloring. *The Practice and Theory of Automated Timetabling VII*, 2008.

Birbas, T.; Daskalaki, S.; Housos, E. School timetabling for quality student and teacher schedules. *Journal of scheduling*, v. 12, n. 2, p. 177–197, 2009.

Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, v. 35, n. 3, p. 268–308, 2003.

Brito, S.; Fonseca, G.; Toffolo, T.; Santos, H.; Souza, M. A sa-vns approach for the high school timetabling problem. *Electronic Notes in Discrete Mathematics*, v. 39, p. 169–176, 2012.

Burke, E.; De Causmaecker, P.; Petrovic, S.; Berghe, G. Variable neighborhood search for nurse rostering problems. *Metaheuristics: computer decision-making*, p. 153–172, 2004.

Burke, E.; Eckersley, A.; McCollum, B.; Petrovic, S.; Qu, R. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, v. 206, n. 1, p. 46–53, 2010.

Caldeira, J.; Rosa, A. School timetabling using genetic search. *Practice and Theory of Automated Timetabling, Toronto*, 1997.

Carpaneto, G.; Toth, P. Primal-dual algorithms for the assignment problem. *Discrete applied mathematics*, v. 18, n. 2, p. 137–153, 1987.

Cerdeira-Pena, A.; Carpente, L.; Farina, A.; Seco, D. New approaches for the school timetabling problem. In: *Artificial Intelligence, 2008. MICAI'08. Seventh Mexican International Conference on*, IEEE, 2008, p. 261–267.

Clark, M.; Henz, M.; Love, B. Quikfix a repair-based timetable solver. 2008.

Coelho, A.; de Souza, S. Um algoritmo híbrido baseado em algoritmos meméticos e reconexão por caminhos para resolução do problema de horário escolar. *Anais do SPOLM, Rio de Janeiro*, 2006.

Constantino, A. A.; de Melo, E. L.; Rizzato, D. B.; Romão, W. Um algoritmo heurístico para otimização do problema de escalonamento de enfermeiros. *Anais: XLI SBPO, Porto Seguro*, 2009.

De Haan, P.; Landman, R.; Post, G.; Ruizenaar, H. A four-phase approach to a timetabling problem in secondary schools. 2006.

Dell'Amico, M.; Trubian, M. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, v. 41, n. 3, p. 231–252, 1993.

Di Gaspero, L.; Schaerf, A. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modelling and Algorithms*, v. 5, n. 1, p. 65–89, 2006.

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

Filho, G.; Lorena, L. A constructive evolutionary approach to school timetabling. *Applications of Evolutionary Computing*, p. 130–139, 2001.

da Fonseca, G.; Ribeiro, R.; Martins, F. Uma abordagem híbrida de sat e busca tabu para o problema da programação de horários escolares. *Pré-anais: XLIII SBPO, Ubatuba*, 2011.

Fonseca, G.; Santos, H.; Toffolo, T.; Brito, S.; Souza, M. A sa-ils approach for the high school timetabling problem. *9th International Conference on the Practice and Theory of Automated Timetabling, Son, Norway*, v. 1, p. 493–496, 2012a.

Fonseca, G.; Toffolo, T.; Brito, S.; Santos, H. Técnicas de busca local para o problema da programação de horários escolares. *Pré-anais: XVI CLAIO - XLIV SBPO - LIA-SGT, Rio de Janeiro*, 2012b.

Hansen, P.; Mladenović, N.; Moreno Pérez, J. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, v. 175, n. 1, p. 367–407, 2010.

Jacobsen, F.; Bortfeldt, A.; Gehring, H. Timetabling at german secondary schools: tabu search versus constraint programming. In: *Proceedings 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT2006), Brno, Czech Republic*, Citeseer, 2006, p. 439–442.

Kaneko, K.; Yoshikawa, M.; Nakakuki, Y. Improving a heuristic repair method for large-scale school timetabling problems. In: *Principles and Practice of Constraint Programming–CP'99*, Springer, 1999, p. 275–288.

Lourenço, H.; Martin, O.; Stützle, T. Iterated local search. *Handbook of metaheuristics*, p. 320–353, 2003.

Lü, Z.; Hao, J.; Glover, F. Neighborhood analysis: a case study on curriculum-based course timetabling. *Journal of Heuristics*, v. 17, n. 2, p. 97–118, 2011.

Melicio, F.; Caldeira, J.; Rosa, A. Two neighbourhood approaches to the timetabling problem. *Proceedings of the practice and theory of automated timetabling (PATAT'04)*, p. 267–282, 2004.

Moura, A.; Scaraficci, R. A grasp strategy for a more constrained school timetabling problem. *International Journal of Operational Research*, v. 7, n. 2, p. 152–170, 2010.

Moura, A.; Scaraficci, R.; Silveira, R.; SANTOS, V. Técnicas metaheurísticas aplicadas à construção de grades horárias escolares. *Simpósio Brasileiro de Pesquisa Operacional*, v. 36, p. 1319–1330, 2004.

Nguyen, K.; Pham, T.; Le, N.; Dang, N.; Tran, N. Simulated annealing-based algorithm for a real-world high school timetabling problem. In: *Knowledge and Systems Engineering (KSE), 2010 Second International Conference on*, IEEE, 2010, p. 125–130.

Osogami, T.; Imai, H. Classification of various neighborhood operations for the nurse scheduling problem. *Algorithms and Computation*, p. 72–83, 2000.

Pillay, N. An overview of school timetabling research. *Practice and Theory of Automated Timetabling (PATAT 2010), August 11-13, Queen's University, Belfast, UK*, 2010a.

Pillay, N. A study into the use of hyper-heuristics to solve the school timetabling problem. In: *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, ACM, 2010b, p. 258–264.

Post, G.; Ahmadi, S.; Daskalaki, S.; Kingston, J.; Kyngas, J.; Nurmi, C.; Ranson, D. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, v. 194, n. 1, p. 385–397, 2012.

Post, G.; Ahmadi, S.; Daskalaki, S.; Kingston, J.; Kyngas, J.; Nurmi, C.; Ranson, D.; Ruizenaar, H. An xml format for benchmarks in high school timetabling. *Practice and Theory of Automated Timetabling (PATAT 2008), August 18 - 22, Universite de Montreal*, 2008.

Post, G.; Kingston, J.; Ahmadi, S.; Daskalaki, S.; Gogos, C.; Kyngas, J.; Nurmi, C.; Santos, H.; Rorije, B.; Schaerf, A. An xml format for benchmarks in high school timetabling ii. *Practice and Theory of Automated Timetabling (PATAT 2010), August 11-13, Queen's University, Belfast, UK*, 2010.

Poulsen, C.; Bandeira, D. Aplicação de um modelo para construção de grades horárias escolares baseado na meta-heurística simulated annealing. *Pré-anais: XVI CLAIO - XLIV SBPO - LIA-SGT, Rio de Janeiro*, 2012.

RAGHAVJEE, R.; PILLAY, N. An application of genetic algorithms to the school timetabling problem. In: *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, ACM, 2008, p. 193–199.

RAGHAVJEE, R.; PILLAY, N. Evolving solutions to the school timetabling problem. In: *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, IEEE, 2009, p. 1524–1527.

RAGHAVJEE, R.; PILLAY, N. An informed genetic algorithm for the high school timetabling problem. In: *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, ACM, 2010a, p. 408–412.

RAGHAVJEE, R.; PILLAY, N. Using genetic algorithms to solve the south african school timetabling problem. In: *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*, IEEE, 2010b, p. 286–292.

RAGHAVJEE, R.; PILLAY, N. The effect of construction heuristics on the performance of a genetic algorithm for the school timetabling problem. In: *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*, ACM, 2011, p. 187–194.

RAHOUAL, M.; SAAD, R. Solving timetabling problems by hybridizing genetic algorithms and tabu search. *Burke and Rudová (2006)*, p. 467–472, 2006.

SANTOS, H.; OCHI, L.; SOUZA, M. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *Journal of Experimental Algorithmics (JEA)*, v. 10, p. 2–9, 2005.

SAVINIEC, L.; CONSTANTINO, A. Applying ils algorithm with a new neighborhood operator to solve a large benchmark of the high school timetabling problem. *Pré-anais: XVI CLAIO - XLIV SBPO - LIA-SGT, Rio de Janeiro*, 2012.

SCHAERF, A. Tabu search techniques for large high-school timetabling problems. In: *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, 1996, p. 363–368.

SCHAERF, A. Local search techniques for large high school timetabling problems. *systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, v. 29, n. 4, p. 368–377, 1999a.

Schaerf, A. A survey of automated timetabling. *Artificial Intelligence Review*, v. 13, n. 2, p. 87–127, 1999b.

Sousa, V.; Moretti, A.; Podestá, V. Programação da grade de horário em escolas de ensino fundamental e médio. *Pesquisa Operacional*, v. 28, n. 3, p. 399–421, 2008.

Souza, M.; Costa, F.; Guimarães, I. Um algoritmo evolutivo híbrido para o problema de programação de horários em escolas. *XXII Encontro Nacional de Engenharia de Produção, Curitiba*, 2002a.

Souza, M.; Martins, A.; Araújo, C.; Costa, F. Métodos de pesquisa em vizinhança variável aplicados ao problema de alocação de salas. *XXII Encontro Nacional de Engenharia de Produção, Curitiba*, 2002b.

Souza, M.; Ochi, L.; Maculan, N. A grasp-tabu search algorithm for solving school timetabling problems. *Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, Boston*, p. 659–672, 2003.

Srndic, N.; Pandzo, E.; Dervisevic, M.; Konjicija, S. The application of a parallel genetic algorithm to timetabling of elementary school classes: A coarse grained approach. In: *Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on*, IEEE, 2009, p. 1–5.

Subramanian, A.; Medeiros, J. M. F.; Cabral, L. A. F.; Souza, M. J. F. Aplicação da metaheurística busca tabu na resolução do problema de alocação de salas do centro de tecnologia da ufpb. *XXVI ENEGEP*, 2006.

Valouxis, C.; Housos, E. Constraint programming approach for school timetabling. *Computers & Operations Research*, v. 30, n. 10, p. 1555–1572, 2003.

Van Laarhoven, P.; Aarts, E.; Lenstra, J. Job shop scheduling by simulated annealing. *Operations research*, p. 113–125, 1992.

Yuri, K.; Polina, K.; Mikhail, P. Formulation space search approach for the teacher/class timetabling problem. *Yugoslav Journal of Operations Research*, v. 18, 2008.

Zhang, D.; Liu, Y.; M'Hallah, R.; Leung, S. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, v. 203, n. 3, p. 550–558, 2010.