CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

ALGORITMOS BASEADOS NA METAHEURÍSTICA VNS PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS

ULISSES REZENDE TEIXEIRA

Orientador: Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto (UFOP)

Coorientador: Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

BELO HORIZONTE
MAIO DE 2019

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

ULISSES REZENDE TEIXEIRA

ALGORITMOS BASEADOS NA METAHEURÍSTICA VNS PARA RESOLUÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS EM ESCOLAS

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional

Linha de pesquisa: Sistemas Inteligentes

Orientador: Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto (UFOP)

Coorientador: Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

BELO HORIZONTE
MAIO DE 2019

Teixeira, Ulisses Rezende
T266a Algoritmos baseados na metaheurística VNS para resolução do problema de programação de horários em escolas. / Ulisses Rezende Teixeira. – – Belo Horizonte, 2019.
xiii, 55 f. : il.

Dissertação (Mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais, Programa de Pós-Graduação em Modelagem Matemática e Computacional, 2018.
Orientador: Prof. Dr. Marcone Jamilson Freitas Souza
Coorientador: Prof. Dr. Sérgio Ricardo de Souza

Bibliografia

1. Programação Heurísticas. 2. Algoritmos Computacionais. 3. Metaheurística. I. Souza, Marcone Jamilson Freitas. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título

CDD 519.6

Elaboração da ficha catalográfica pela Bibliotecária Elisângela Gonçalves Barbieri CRB-6: 2624 / CEFET-MG

Para minha mãe Ruth, minha amada esposa
Ana Paula e filhas Ana Lívia e Bianca.

Agradecimentos

Dedicação, foco e perseverança. Estas três palavras definem minha caminhada para obtenção deste título de mestre. Há tanto o que agradecer que é difícil descrever em tão poucas palavras. Em primeiro lugar, ao Pai Celestial, que me deu forças para iniciar, permanecer e concluir esta caminhada. Esta força superior é que nos move e nos mantêm motivados em busca de nossos sonhos e objetivos. À minha mãe, pela criação, educação e incentivos dados desde criança, me mostrando a importância de estudar e buscar nossos sonhos. À meu pai (em memória), que está sempre comigo, me apoiando e direcionando em todas as minhas decisões como meu anjo da guarda. À minha esposa Ana Paula, que sempre me deu forças e me ajudou a continuar motivado e em frente, mesmo com todas as dificuldades e distrações. Às minhas filhas Ana Lívia e Bianca, pela compreensão, mesmo pequenas e sem entenderem o motivo, ajudaram, a seu modo, me permitindo muitas noites, fins de semana e feriados de isolamento. Ao final desta caminhada, a ausência e a privação de brincadeiras e passeios valerá a pena. Ao meu orientador, prof. Dr. Marcone, e ao meu co-orientador, prof. Dr. Sérgio, por terem me aceito como orientando e por terem fornecido todo o apoio que precisei para a realização deste trabalho. São exemplos de profissionais e pessoas. Não posso deixar de agradecer também àquelas pessoas que contribuíram diretamente para que eu pudesse ingressar e concluir este projeto: profs. Dr. Ronald Zanetti e Guilherme Bastos pelas recomendações durante o processo de seleção. Desde a graduação, há mais de 15 anos, tenho-os como referência no meio acadêmico e profissional. Ao amigo e ex-colega de trabalho Júlio Alves (dentre outros), que trilhou o caminho de um curso de pós graduação stricto sensu paralelamente ao trabalho e serviu de exemplo e incentivo para que eu pudesse permanecer firme neste objetivo. Aos colegas de trabalho, em especial Marcus (CTO), e toda a minha equipe da *Starline*, por terem permitido conciliar meus estudos ao ambiente de trabalho. Aos colegas de mestrado que compartilharam disciplinas e sempre ajudaram nos momentos de dificuldade e dúvidas. Por fim, agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES e à instituição Centro Federal de Educação Tecnológica - CEFET - MG, por terem possibilitado a realização deste sonho.

“Todas as vitórias ocultam uma abdicação.” (Simone de Beauvoir)

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes.” (Marthin Luther King)

Resumo

Este trabalho tem seu foco no Problema de Programação de Horários em Escolas. Este problema consiste em produzir um quadro de horários, tipicamente semanal, para os encontros entre professores e turmas de instituições de ensino, satisfazendo a restrições de várias naturezas. Dada a sua natureza combinatória, foram desenvolvidos cinco algoritmos heurísticos baseados na metaheurística *Variable Neighborhood Search* (VNS) para resolvê-lo. Todos os algoritmos exploram o espaço de soluções do problema por meio dos seguintes tipos de movimento: troca de eventos, troca de recursos, troca de bloco de recursos, mudança de horário de recurso, mudança de horário de evento e cadeia Kempe. Os três primeiros algoritmos desenvolvidos consistem em variações do método *General Variable Neighborhood Search* (GVNS), que diferem entre si pelo método de busca local usado na fase de refinamento. O quarto algoritmo desenvolvido, nomeado VNS Adaptativo, consiste em aplicar a busca local em uma vizinhança escolhida probabilisticamente, sendo que a probabilidade é maior para as vizinhanças cujas buscas locais foram mais bem sucedidas em iterações anteriores. O quinto algoritmo desenvolvido, nomeado *Skewed General Variable Neighborhood Search* (SGVNS), explora o espaço de soluções do problema permitindo, também, a geração de soluções intermediárias que sejam piores que a solução corrente. Todos os algoritmos usam a biblioteca KHE (*Kingston High School Engine*), especializada para implementação de problemas de programação de horários. Os algoritmos foram testados em problemas teste da Competição Internacional de Programação de Horários ocorrida em 2011 e seus resultados foram comparados com aqueles gerados pelo algoritmo *Goal Solver*, vencedor da competição. Todos os cinco algoritmos implementados apresentaram bons resultados, superando, inclusive, o vencedor da competição em vários problemas-teste. Com exceção do algoritmo VNS Adaptativo, todos os demais se mostraram estatisticamente equivalentes quando comparados ao *Goal Solver*.

Palavras-chave: *Programação de Horários, Programação de Horários em Escolas, Metaheurísticas, Busca em Vizinhança Variável, Descida em Vizinhança Variável*

Abstract

This work focus on the school timetabling problem. This problem consists of producing timetabling, usually weekly, for the meetings between teachers and classes in educational institutions, satisfying constraints. In view of its combinatorial nature, it were developed five heuristics algorithms based on VNS metaheuristic to solve it. All algorithms explore the solution space with the following types of moves: event swap, resource swap, resource block swap, resource change, event change, and Kempe's chain. The first three algorithms developed consists of variations from the General Variable Neighborhood Search (GVNS) method, differing among them by the local search method on the refinement phase. The fourth algorithm developed, called Adaptive VNS, consists of choosing, probabilistically, a neighborhood to apply the local search, in such a way that the neighborhoods most successful in previous iterations are more likely to be chosen. The fifth algorithm developed, called Skewed General Variable Neighborhood Search (SGVNS), explores the solution space of the problem allowing the generation of intermediate solutions that are worse than the current solution. All algorithms use the KHE (Kingston High School Engine) library, specialized in timetabling problems. The algorithms were tested in instances from the International Timetabling Competition of 2011 and their results were compared with those of the Goal Solver, winner of this competition. All of the five algorithms implemented presented good results, overcoming the winner of the competition in some instances. Excepted the Adaptive VNS algorithm, all of the others were statistically equivalent to Goal Solver.

Keywords: *Timetabling, School Timetabling, Metaheuristics, Local Search, Variable Neighborhood Search, Variable Neighborhood Descent*

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

CEFET	Centro Federal de Educação Tecnológica
DECOM	Departamento de Computação
VNS	*Variable Neighborhood Search*
VND	*Variable Neighborhood Descent*
GVNS	*General Variable Neighborhood Search*
rVND	*Random Variable Neighborhodd Descent*
BLVA	Busca Local Vizinhança Variável
SGVNS	*Skewed General Variable Neighborhood Search*
PATAT	*Practice and Theory of Automated Timetabling*
ITC	*International Timetabling Competition*
GOAL	*Group of Optimization and Algorithms*
KHE	*Kingston High School Engine*
XHSTT	*XML HighSchool Timetabling*

Sumário

Capítulo 1

Introdução

Um quadro de horários é uma tabela em que são alocados recursos em espaços de tempo previamente determinados. No contexto de instituições de ensino e de forma mais específica, um quadro de horários escolar é uma tabela que registra a sequência de matérias (Português, Matemática, Física, Química, etc) que cada turma (1º ano A, 1º ano B, 2º ano A, etc) terá semanalmente.

Montar este quadro é uma tarefa comum a todas as instituições de ensino a cada início de período letivo. Esta tarefa, de forma bastante simplificada, consiste em combinar horários, turmas e professores de forma a atender ao maior número possível de restrições. Essas restrições podem ser de vários tipos, como: indisponibilidade de professores em alguns horários, redução máxima de janelas de horários de professores, restrição a aulas da mesma disciplina de forma sequencial, restrição de alocação de somente uma única turma por professor no mesmo horário.

Não foram encontrados trabalhos nem dados oficiais a respeito do assunto, mas acredita-se que a grande maioria das instituições de ensino, em especial no Brasil, realizem a atividade de montagem de horários de forma completamente manual. Este trabalho de encontrar uma solução, conforme Bardadym (1995), pode levar horas, ou até mesmo dias, dependendo do tamanho da instituição e da quantidade de turmas envolvidas. Assim, a solução utilizada normalmente é a primeira solução viável encontrada, dado o grande esforço para melhorá-la.

Com toda esta complexidade envolvida nesse trabalho, bem como seu grande interesse prático, este tipo de problema vem sendo estudado pela academia desde a década de 1960 (GOTLIEB, 1963), atraindo a atenção de vários pesquisadores de diversas áreas, em especial, daqueles que utilizam métodos de otimização.

Esta dissertação concentra-se na subclasse do problema de programação de horários conhecida como *School Timetabling*. São propostos algoritmos que usam a metaheurística *Variable Neighborhood Search* (VNS) e a biblioteca *Kingston High School timetabling engine*

(KHE), que são detalhados ao longo deste texto.

1.1 Objetivos

O objetivo geral desta dissertação é apresentar alternativas de solução para problemas de programação de horários em escolas, implementando algoritmos baseados na metaheurística *Variable Neighborhood Search* (VNS).

Como objetivos específicos, é possível elencar:

- Revisar a literatura relacionada ao problema *school timetabling* e à metaheurística VNS;
- Avaliar a adequação da metaheurística VNS para a solução desta classe de problemas de programação de horários;
- Avaliar formas alternativas de implementação da etapa de refinamento do algoritmo;
- Testar os algoritmos desenvolvidos usando dados da última competição internacional de programação de horários e comparar seus resultados com os do algoritmo vencedor dessa competição.

1.2 Organização da dissertação

O restante desta dissertação está organizado como segue. No Capítulo 2 o problema abordado é descrito. No Capítulo 3 é realizada uma revisão da literatura, descrevendo os principais e mais recentes trabalhos relacionados ao tema.

No Capítulo 4 apresenta-se a abordagem escolhida para a solução do problema, detalhando os principais pontos e estruturas usados, além de exibir os algoritmos implementados.

No Capítulo 5 são apresentados os resultados encontrados com as comparações de desempenho dos métodos desenvolvidos com o algoritmo *Goal Solver*, vencedor da última competição internacional, o ITC [1] (*International Timetabling Competition*) 2011.

No último capítulo são feitas as considerações finais e apresentadas as sugestões de trabalhos futuros.

[1] https://www.utwente.nl/en/eemcs/dmmp/hstt/itc2011/

Capítulo 2

Descrição do problema

Neste capítulo serão apresentadas as características gerais de um problema de quadro de horários. Será abordada a classe deste problema relacionada ao ambiente escolar mostrando suas especificidades.

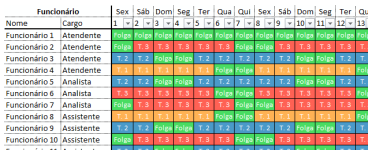
2.1 O problema de quadro de horários

Problemas de programação de horários (*timetabling problems*) são bastante comuns em nossa vida cotidiana. Eles podem ser encontrados em várias situações, como:

- Grade de horários escolares: aulas de uma determinada turma em cada dia da semana;
- Escala de trabalho de enfermeiras: distribuição de profissionais de enfermagem em horários previamente definidos para garantir o bom atendimento dos pacientes;
- Programação de horários em eventos esportivos: alocação dos competidores em horários considerando cada modalidade esportiva;
- Programação de horários no sistema de transporte coletivo: distribuição dos veículos em horários definidos pela demanda de passageiros.

A Figura 1 exibe exemplos de quadros de horários envolvendo as mais diversas áreas e aplicações. Nela, pode ser visto um quadro de horários escolar mostrando as disciplinas de cada horário em cada dia da semana; a especialidade desportiva de cada horário em uma competição; os horários de uma determinada linha de ônibus e os horários de uma determinada equipe alocada para trabalhar em um hospital.

Apesar de parecer simples em um primeiro momento, montar um quadro de horários não é uma tarefa tão trivial e rápida quanto pode parecer à primeira vista. Em cenários com muitos



Funcionário		Sex	Sáb	Dom	Seg	Ter	Qua	Qui	Sex	Sáb	Dom	Seg	Ter	Qu
Nome	Cargo	1	2	3	4	5	6	7	8	9	10	11	12	13
Funcionário 1	Atendente	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Folga	Fol
Funcionário 2	Atendente	Folga	T.3	T.3	T.3	T.3	T.3	Folga	Folga	T.3	T.3	T.3	T.3	T.
Funcionário 3	Atendente	T.2	T.2	Folga	Folga	T.2	T.2	T.2	T.2	T.2	Folga	Folga	T.2	T.
Funcionário 4	Atendente	T.1	T.1	T.1	T.1	T.1	T.1	Folga	Folga	T.1	T.1	T.1	T.1	Fol
Funcionário 5	Analista	T.2	T.2	Folga	Folga	T.2	T.2	T.2	T.2	T.2	Folga	Folga	T.2	T.
Funcionário 6	Analista	T.3	T.3	T.3	T.3	T.3	Folga	Folga	T.3	T.3	T.3	T.3	T.3	Fol
Funcionário 7	Analista	Folga	T.3	T.3	T.3	T.3	T.3	Folga	Folga	T.3	T.3	T.3	T.3	T.
Funcionário 8	Assistente	T.1	T.1	T.1	T.1	T.1	Folga	Folga	T.1	T.1	T.1	T.1	T.1	Fol
Funcionário 9	Assistente	T.2	T.2	Folga	Folga	T.2	T.2	T.2	T.2	T.2	Folga	Folga	T.2	T.
Funcionário 10	Assistente	Folga	T.3	T.3	T.3	T.3	T.3	Folga	Folga	T.3	T.3	T.3	T.3	T.

(a) Enfermaria

HORÁRIOS	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA
07:00	Legislação	Dir. Defensiva	Primeiros Socorros	Meio Ambiente	Mecânica
08:00	Dir. Defensiva	Legislação	Legislação	Dir. Defensiva	Primeiros Socorros
09:00	Dir. Defensiva	Legislação	Legislação	Dir. Defensiva	Meio Ambiente
16:00	Meio Ambiente	Primeiros Socorros	Mecânica	Dir. Defensiva	Legislação
17:00	Legislação	Legislação	Legislação	Primeiros Socorros	Dir. Defensiva
18:00	Primeiros Socorros	Meio Ambiente	Meio Ambiente	Legislação	Dir. Defensiva
19:00	Dir. Defensiva	Legislação	Legislação	Primeiros Socorros	Meio Ambiente
20:00	Mecânica	Legislação	Legislação	Dir. Defensiva	Dir. Defensiva

(b) Escola

QUADRO DE HORÁRIOS PARA O FESTIVAL INAUGURAL - Sábado dia 05/3/16

HORÁRIO	QUADRA	ÁREA COMUM
8h00 as 8h40	HANDEBOL 1 (12 anos acima)	TAEKWONDO KIDS (3 a 5 anos – Ens. Inf.)
8h45 as 9h25	FUTSAL KIDS (3 a 5 anos – Ens. Inf.)	TAEKWONDO 1 (6º ao 9º ano)
9h30 as 10h10	FUTSAL 1 (6 a 8 anos)	TAEKWONDO 2 (06 a 10 anos)
10h15 as 10h45	HANDEBOL 2 (11 anos abaixo)	TAEKWONDO 3 (11 anos acima)
10h50 as 11h30	FUTSAL 2 (9 a 12 anos)	

(c) Evento exportivo

NOVO QUADRO DE HORÁRIOS TRÊS RIOS X ALÉM PARAÍBA			NOVO QUADRO DE HORÁRIOS TRÊS RIOS X ALÉM PARAÍBA		
SAÍDA DE TRÊS RIOS			SAÍDA DE ALÉM PARAÍBA		
2ª A 6ª FEIRA	SABADO	DOMINGO	2ª A 6ª FEIRA	SABADO	DOMINGO
6:15	7:30	9:00	4:15	5:10	6:30
7:30	9:15	13:30	5:10	6:00	11:15
9:15	11:00	15:30	6:00	7:15	13:30
11:00	13:00	17:40	7:15	9:30	15:30
13:00	14:30	19:30	09:45	12:00	17:30
14:30	16:15	22:45	12:00	13:30	20:30
16:15	17:45		14:00	15:30	
17:45	19:45		15:30	17:45	
20:30	22:30		18:15	20:30	
22:30			20:30		

(d) Transporte

Figura 1 – Exemplos de quadros de horários.

horários e recursos, o tempo necessário para avaliar as combinações possíveis aumenta exponencialmente.

Esta dissertação tem seu foco em uma categoria deste problema relacionado ao ambiente educacional, mais especificamente no ensino regular (fundamental e médio no Brasil).

Este problema, conhecido como programação de horários em escolas (em inglês, *School timetabling*) é, na realidade, um subproblema de programação de horários (em inglês, *Timetabling*). Segundo Schaerf (1999), este último pode ser classificado em três classes diferentes de problemas:

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

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

Problemas de programação de horários são bastante estudados pela academia desde o início da década de 60, iniciando-se por Gotlieb (1963). Os trabalhos relatam aplicações de várias técnicas diferentes ao longo de todos estes anos para resolver uma grande variedade de problemas. Isso motivou a criação de conferências internacionais exclusivas para tratar do tema, como a Conferência Internacional de Automação de Problemas de Horário (*PrActice and Theory on Automated Timetabling* - PATAT). Também foram organizadas competições específicas, como a Competição Internacional de Programação de Horários (*International Timetabling Competition* - ITC).

Todo este interesse se baseia em três pontos principais, de acordo com Schaerf (1999):

- Dificuldade de encontrar uma solução: devido à grande quantidade de restrições, encontrar uma solução viável se torna uma tarefa bastante árdua e que pode levar dias de trabalho manual, de acordo com a quantidade de recursos (turmas, professores, horários) envolvidos;
- Importância prática: ter um quadro de horários é uma necessidade básica de todas as instituições de ensino. Um bom quadro de horários pode impactar a vida de uma grande quantidade de indivíduos, sejam eles alunos ou professores e pode, inclusive, interferir na eficiência de professores e no desempenho de alunos;
- Importância teórica: é demonstrado, de acordo com Even, Itai e Shamir (1975), que problemas de programação de horários pertencem à classe de problemas NP-difíceis, sendo objeto de constantes estudos para se obter métodos que gerem boas soluções em tempos computacionais compatíveis com aqueles necessários para a tomada de decisão, independentemente de sua dimensão.

Segundo Souza (2000), a classe de problemas conhecida como *High School Timetabling* ou Problema Classe-Professor tem, como característica básica, a repetição da programação semanal. Cada turma apresenta um único conjunto de disciplinas para todos os seus alunos e suas aulas são distribuídas em um mesmo turno (manhã, tarde ou noite).

O conceito de turma neste cenário é definido como sendo um conjunto de alunos que possuem uma mesma grade curricular e assistem suas aulas em uma mesma sala de aula. A alocação das salas de aula não faz parte do problema, já que é associada diretamente à turma. Schaerf (1999) descreve variações deste problema, como a ocorrência de aulas envolvendo mais de uma turma, disciplinas ministradas por mais de um professor e alocação de salas de aula específicas de acordo com a disciplina. Este trabalho, entretanto, se concentra na solução do problema básico, sem variações.

Mesmo no problema básico, há uma série de pontos envolvendo necessidades e desejos que acabam tornando esta atividade bastante complexa. Por exemplo: entender as necessidades de cada disciplina e respeitar a disponibilidade de cada professor. Todos estes pontos são conhecidos como restrições.

As restrições costumam ser divididas em dois grupos, segundo Santos, Ochi e Souza (2005):

- Restrições fortes: são responsáveis pela viabilidade ou não de uma determinada solução. Como exemplo, podemos citar a alocação de um professor a apenas uma turma no mesmo horário. A alocação de um mesmo professor em duas turmas diferentes no mesmo horário representa o não atendimento desta restrição. Soluções viáveis ou factíveis possuem todas as restrições deste tipo atendidas, ao passo que uma solução que possui uma restrição forte não atendida é uma solução inviável;
- Restrições fracas: são responsáveis por melhorar uma determinada solução que já era viável, tornando-a ainda melhor. Quanto mais restrições deste tipo forem atendidas, melhor será a solução. Entretanto, o não atendimento a estas restrições não inviabiliza uma determinada solução. Diferentes tipos de restrições fracas podem apresentar pesos variados. Como exemplo, podemos citar a existência de horários vagos (conhecidos como janelas) no quadro de um determinado professor. É esperado que se reduza ao máximo a ocorrência de janelas, mas sua existência não inviabiliza a solução.

O objetivo de um bom quadro de horários é atender a todas as restrições fortes e ao maior número possível de restrições fracas. Desta forma, garante-se uma correta distribuição das disciplinas e a satisfação do maior número possível de envolvidos.

A qualidade de um quadro de horários afeta também o corpo discente, o qual, tendo um quadro com as matérias bem distribuídas, poderá ser beneficiado no processo de aprendizagem, tendo como resultado uma melhoria no desempenho escolar.

2.2 O problema abordado

Esta dissertação aborda a classe de quadro de horários por turma, conhecida na literatura como *school timetabling*.

Nesta classe do problema de quadro de horários, são definidos três itens básicos:

- Dias e horários: dado um número de dias úteis na semana (normalmente cinco ou seis), cada dia é dividido em um número de períodos. Um horário é representado pelo par composto por um dia e um período. O número total de horários disponíveis para alocação é o somatório de todos os períodos de todos os dias da semana;
- Turmas: uma turma representa um grupo de alunos que realizam as mesmas disciplinas;
- Disciplinas e Professores: cada disciplina é composta por um número fixo de aulas (agendadas em períodos distintos), podendo ser ministrada por professores específicos.

Salas de aula não são consideradas nesta classe do problema, pois estão diretamente ligadas à turma, e todas as aulas são ministradas no mesmo local.

De modo geral, o problema consiste na combinação destes itens básicos. O resultado desta combinação é denominado "aula". Assim, uma aula é representada pela alocação de uma turma e professor para a ocorrência de uma disciplina em um horário (dia da semana e período do dia).

Estas alocações devem seguir as restrições fortes e fracas, que são definidas de acordo com cada cenário. Um horário válido deve garantir o atendimento de todas as restrições fortes definidas. Como exemplo de restrições fortes comuns para este tipo de problema tem-se: (i) garantir que cada professor seja alocado no máximo em um horário; (ii) garantir que a carga horária semanal de cada disciplina e de todas as turmas, seja respeitada.

A alocação realizada para todos os dias da semana e horários se repete semanalmente e todos os estudantes da mesma turma seguem esta mesma alocação.

Capítulo 3

Revisão de literatura

Neste capítulo serão apresentados alguns dos principais trabalhos relacionados ao problema abordado. Será feito, também, uma explicação sobre a abordagem utilizada na solução do problema, apresentando os conceitos principais envolvidos na abordagem, bem como os algoritmos descritos na literatura que foram empregados para a solução do mesmo.

3.1 Trabalhos relacionados

Os primeiros trabalhos dedicados ao estudo de problemas de programação de horários em escolas remontam ao final da década de 1960, com Appleby, Blake e Newman (1961) e Csima e Gotlieb (1964). Nestes trabalhos foram utilizadas heurísticas construtivas, simulando a forma manual de resolver o problema. Essas soluções são construídas de forma gradual, alocando aula a aula e resolvendo os conflitos à medida que eles aparecem.

Posteriormente, os pesquisadores começaram a aplicar técnicas mais elaboradas, como programação inteira, fluxo em grafos e coloração de grafos. Essas técnicas podem ser vistas em trabalhos de Wood (1969) e Tripathy (1984), em que são usados programação inteira e coloração de grafos, respectivamente.

Em 1985, Werra (1985) utilizou modelos matemáticos para o problema, apresentando resultados com abordagens utilizando coloração de grafos e fluxo em rede. Cerca de uma década depois, Werra (1997) apresentou uma revisão sobre os mesmos modelos matemáticos e também discorreu sobre a utilização de métodos exatos e de coloração de grafos na solução destes problemas.

Na década de 1990, um importante trabalho de Cooper e Kingston (1995) apresenta a comprovação de que problemas de alocação de horários apresentam, em seu problema de decisão, complexidade NP-Completo, no caso geral, apresentando novas perspectivas de pesquisas. Esse trabalho confirma o que havia sido apresentado anteriormente por Even,

Itai e Shamir (1975).

Ainda na década de 1990, mais especificamente em 1996, o interesse nessa classe de problemas levou à criação de um grupo de trabalho denominado *EWG PATAT* (*Euro Working Group on Automated Timetabling*). Este grupo organiza desde então uma conferência bianual na qual são apresentados os avanços na área e discutidos assuntos gerais relacionados ao tema. Esporadicamente, o grupo organiza também competições com temas específicos. A última dessas competições ocorreu em 2011 e apresentou, como tema central, a classe de problemas objeto deste estudo (*High School Timetabling*).

No final dos anos 2000 foi apresentado pelo EWG PATAT um importante avanço nos estudos da área, qual seja, a definição de um padrão para representação do problema e sua respectiva solução. Este padrão foi denominado de XHSTT ou *XML HIGH SCHOOL TIMETABLING*. Além do padrão propriamente dito, esse formato se tornou um repositório público e global de problemas teste criados por vários estudiosos de várias partes do mundo, com problemas reais e fictícios dos mais variados tipos. A partir da definição deste padrão, já no início dos anos 2010, Kingston (2012) publicou um importante trabalho descrevendo sua biblioteca para manipulação dos arquivos XHSTT e sua ferramenta de avaliação de soluções, elevando o nível dos estudos desta categoria de problemas.

As metaheurísticas começaram a ser amplamente utilizadas para a solução desta classe de problemas a partir da década de 1990 e continuam até os dias atuais, sejam aquelas com abordagem de busca local ou evolucionárias. Vários estudos utilizando *Simulated Annealing*, Algoritmos Genéticos, Busca Tabu e GRASP, dentre outros, foram publicados por diversos autores desde então, como, por exemplo, Zhang et al. (2010), Hertz (1991) e Thompson e Dowsland (1998).

No Brasil, podemos citar os trabalhos de Souza (2000) e Santos, Ochi e Souza (2005). O primeiro apresentou características e classificações sobre o problema e a aplicação dos principais algoritmos heurísticos disponíveis na literatura da época. Já o segundo apresentou abordagens híbridas utilizando técnicas heurísticas e de programação linear inteira mista, focando na hibridização entre heurísticas e métodos exatos.

A competição ITC ocorrida em 2011 (última edição) foi vencida pela equipe brasileira *Goal*, organizada pelo Departamento de Computação (DECOM) da Universidade Federal de Ouro Preto (UFOP). O algoritmo implementado, descrito em Fonseca et al. (2016), combina as metaheurísticas *Simulated Annealing* e ILS (*Iterated Local Search*).

O uso de algoritmos híbridos e hiperheurísticas tem sido cada vez maior em estudos mais recentes, sendo estes os métodos mais atuais de estudos em problemas de *school timetabling*.

Pillay (2014) apresenta um amplo trabalho de *survey*, com uma visão bastante abrangente sobre os trabalhos mais recentes relacionados a esta classe de problemas.

3.2 Heurísticas

Uma heurística pode ser definida como sendo um algoritmo que encontra uma solução factível de forma empírica, não necessariamente a melhor solução, para um determinado problema, com uma determinada função objetivo em um tempo computacional razoável (DÍAZ et al., 2000).

Dada esta definição, é esperado que esta abordagem seja bastante utilizada para resolver problemas da classe NP-difícil, pois heurísticas apresentam um desempenho médio interessante. Entretanto, nenhum método heurístico, por definição, garante solução ótima.

Em geral, esses métodos utilizam combinação de escolhas aleatórias e conhecimento histórico dos resultados já alcançados anteriormente para se guiarem e realizarem buscas pelo espaço de soluções do problema, no sentido de evitar paradas prematuras em ótimos locais.

Metaheurísticas, por sua vez, são heurísticas genéricas, normalmente mais sofisticadas, que partem de analogias físicas, biológicas ou etológicas (BLUM; ROLI, 2003). Como exemplo, podemos citar *Simulated Annealing*, Algoritmos Genéticos e ILS (*Iterated Local Search*).

As metaheurísticas se diferenciam pelos seguintes aspectos:

- Critério de escolha da solução inicial;
- Definição da vizinhança de uma solução;
- Critério de seleção de uma solução vizinha;
- Estratégia adotada para não a busca não ficar presa em ótimos locais, sendo esta a principal característica de diferenciação.

As heurísticas podem ser classificadas em construtivas e de refinamento (BLUM; ROLI, 2003), conforme será descrito nas duas próximas subseções.

3.2.1 Heurísticas Construtivas

Heurísticas construtivas são aquelas em que uma solução é construída elemento a elemento, seguindo um critério previamente definido, até que se tenha chegado a uma solução viável.

Normalmente, estes métodos se mostram do tipo guloso, ou seja, o elemento escolhido a ser inserido na solução a cada passo é o melhor elemento naquele momento.

Heurísticas construtivas, normalmente, são empregadas como métodos para a geração de uma solução inicial para uma metaheurística.

3.2.2 Heurísticas de refinamento

Heurísticas de refinamento, por outro lado, partem de uma solução inicial previamente construída e tentam melhorá-la ao máximo possível.

As heurísticas de refinamento utilizam o conceito de vizinhança, que está associado ao conceito de movimento. Um movimento nada mais é do que uma determinada modificação na solução atual.

As heurísticas de refinamento ficam presas no primeiro ótimo local encontrado, o que restringe a busca no espaço de soluções e limita a qualidade da solução final.

As técnicas de refinamento podem ser subdivididas basicamente em três estratégias, caracterizando-se pelo critério de parada ao percorrer o espaço de soluções:

- *Best improvement* ou técnica do melhor resultado: percorre todos os vizinhos de uma determinada solução e retorna o melhor entre todos eles. Este método tem, como pontos positivos, o retorno da melhor solução possível dada uma determinada vizinhança. Por outro lado, caso a quantidade de vizinhos seja muito alta, o método acaba perdendo desempenho;
- *First improvement* ou técnica da primeira melhora: percorre os vizinhos de uma determinada solução até encontrar uma solução que seja melhor que a solução corrente, situação em que a busca é interrompida. Este método tem como principal ponto positivo evitar uma busca exaustiva, pois apenas no pior caso toda a vizinhança será explorada;
- *Random Descent*: neste caso, cada vizinho é escolhido aleatoriamente. Se ele representar uma melhora em relação à solução corrente, ele é aceito e passa a ser a nova solução. Caso contrário, outro vizinho é testado. Se, ao final de um número predeterminado de iterações, nenhum vizinho for aceito, o método é interrompido e retorna a melhor solução encontrada durante a busca.

3.2.3 VNS, GVNS e SVNS

Conforme os próprios autores definem, *Variable Neighborhood Search* (VNS) é uma metaheurística baseada em mudanças sistemáticas de vizinhanças, tanto na fase de busca

local, quanto na fase de perturbação, para diversificar as soluções. Foi proposta por Mladenović e Hansen (1997) e tem como foco a solução de problemas de otimização combinatória.

O Algoritmo 1 apresenta o pseudocódigo da metaheurística VNS em sua forma original, publicada por Mladenović e Hansen (1997). O primeiro laço de repetição (linha 4) do algoritmo define um critério de parada, que pode ser um tempo de execução ou um número determinado de iterações. O segundo laço de repetição (linha 6) representa a quantidade de estruturas de vizinhança disponíveis. Sempre que o algoritmo encontra uma solução de melhora, o algoritmo força a execução de volta à primeira estrutura de vizinhança (linha 12). Já o terceiro e último laço de repetição (linha 7) percorre todos os vizinhos de uma determinada vizinhança, comparando o resultado de sua função objetivo com a solução corrente.

Algoritmo 1: VNS ORIGINAL

Entrada: Conjunto $\mathcal{N}_k$ de estruturas de vizinhança, k = 1, ..., k_{max}; solução inicial $x \in S$

Saída: Uma solução x^* aproximada do problema

```
1  início
2      x* ← x;
3      f* ← f(x);
4      enquanto condição de parada não encontrada faça
5          k ← 1;
6          enquanto k <= k_max faça
7              para cada vizinho y ∈ N_k(x*) faça
8                  aplique um método de busca local em y para obter um ótimo local y';
9                  se f(y') < f* então
10                     x* ← y';
11                     f* ← f(y');
12                     retorne ao passo da linha 5
13                 fim
14             fim
15             k ← k + 1;
16         fim
17     fim
18 fim
19 retorna x*
```

O algoritmo GVNS (*General Variable Neighborhood Search*) é uma metaheurística baseada em busca local apresentada por Mladenović et al. (2008). O algoritmo é baseado em uma versão generalizada da metaheurística VNS (*Variable Neighborhood Search*) proposta anteriormente pelos mesmos autores (MLADENOVIĆ; HANSEN, 1997), na qual o método

de busca local é o VND (*Variable Neighborhood Descent*).

Por sua vez, o algoritmo SVNS, ou *Skewed* VNS, também é uma metaheurística baseada em busca local e também foi criada por Mladenović et al. (2008). Seu principal diferencial é usar um parâmetro que define uma margem para aceitar soluções que são piores que a solução atual. A concepção deste método baseia-se no fato de que as melhores soluções podem estar muito distantes da solução atual. Por isso, faz-se necessário passar por soluções intermediárias e, às vezes, piores para alcançá-las.

A Seção 4.6, apresentada no próximo Capítulo, detalha melhor o funcionamento desses algoritmos e suas respectivas implementações.

Capítulo 4

Metodologia

Neste Capítulo são apresentados os algoritmos implementados como solução para o problema apresentado, assim como os recursos empregados na implementação desses algoritmos.

Inicialmente é mostrado o formato XHSTT para representar as restrições, os eventos e os recursos existentes nos problemas-teste. Em seguida, é mostrada a biblioteca KHE para manipular essas informações. Posteriormente são apresentados os movimentos utilizados para explorar o espaço de soluções do problema e, finalmente, os algoritmos propostos.

4.1 Representação do problema e da solução: formato XHSTT

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

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

Como o próprio nome diz, o XHSTT é um formato baseado em XML (*eXtensive Markup Language*) e estabelece estruturas específicas para tratar recursos, horários e suas respectivas restrições. Já o repositório público criado no mesmo projeto foi composto por problemas-testes com situações reais e outros simulados, coletados de diversos estudos realizados ao redor do mundo, como por exemplo, Inglaterra (WRIGHT, 1996), Finlândia (NURMI; KYNGAS, 2007), Grécia (VALOUXIS; HOUSOS, 2003), Holanda (HAAN et al.,

2006) e Brasil (SOUZA, 2000).

O modelo XHSTT é dividido em 3 entidades básicas, sendo:

- Tempo e recursos: a entidade tempo é composta por horários e os recursos são subdivididos em 3 subcategorias: turmas, professores e salas;
- Eventos: um evento é a unidade básica de associação, representando uma aula. Um evento pode ser uma tarefa, quando está vinculado a um recurso, ou um encontro, quando está vinculado a um horário;
- Restrições: orientam a distribuição dos recursos nos eventos. Podem ser definidas como sendo fortes ou fracas, de acordo com os critérios esperados para que uma determinada solução seja viável ou não. Além disso, são subdivididas em 3 subcategorias: restrições básicas de agendamento, restrições de eventos e restrições de recursos.

As restrições básicas de agendamento podem ser:

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

As restrições de eventos podem ser:

- Vincular eventos: atribui um conjunto de eventos a um mesmo horário de início;
- Espalhar eventos: distribui eventos de forma uniforme nos horários disponíveis;
- Evitar quebra de atribuições: para cada evento, atribui um dado recurso a todos os seus horários;
- Distribuir quebra de eventos: garante que o número de aulas consecutivas estejam entre um número mínimo e um número máximo de horários;
- Quebrar eventos: impõe limites no número de encontros não consecutivos criados por um evento e sua duração.

As restrições de recursos podem ser:

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

4.2 Função de avaliação: biblioteca KHE

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

Essa biblioteca implementa toda a estrutura da função de avaliação necessária para avaliar cada solução gerada. A função de avaliação é dada pelo somatório de todas as restrições não atendidas e é calculada por dois fatores distintos: o primeiro é a quantidade de restrições fortes não atendidas e o segundo é a quantidade de restrições fracas não atendidas.

O resultado da função de avaliação deve ser zero ou o mais próximo possível de zero para os dois valores, sendo que, caso ambos os valores sejam iguais a zero, significa que a solução é ótima, ou seja, todas as restrições foram atendidas. Quando o primeiro valor for

diferente de zero, significa que é uma solução infactível, isto é, não foi possível encontrar uma solução que atenda a todas as restrições fortes. Quando o primeiro valor é zero e o segundo diferente de zero, a solução é factível, mas pelo menos uma restrição fraca deixou de ser atendida.

4.3 Movimentos

Um movimento, em uma metaheurística, é uma operação definida por uma regra que, quando aplicada, modifica a solução corrente gerando uma nova solução. A definição de cada movimento não está condicionada à viabilidade da solução encontrada pela sua aplicação. A qualidade desta solução deverá ser avaliada através do resultado de sua função objetivo, definindo, então, se a nova solução será aceita ou descartada.

Na implementação dos algoritmos foram especificados seis movimentos diferentes encontrados em trabalhos anteriores e descritos amplamente na literatura especializada. Nas próximas subseções serão detalhados cada um desses movimentos.

4.3.1 Cadeia de Kempe

O movimento da cadeia de Kempe foi proposto por Johnson et al. (1991) para o problema de coloração de vértices em grafos e baseia-se no conceito de que determinadas mudanças podem gerar soluções inviáveis, criando conflitos. Para resolver esses conflitos, é necessária uma cadeia de outros movimentos. O conjunto dessas modificações sequenciais realizadas em uma determinada solução é denominado Cadeia de Kempe (*Kempe Chain Interchanges - KCI*).

O algoritmo busca e retorna a melhor cadeia de movimentos possível, considerando as cadeias mais longas como sendo mais promissoras. Este movimento apresenta um papel importante na diversificação de soluções, o que o capacita como uma boa escolha para ser usado na etapa de *shaking* do algoritmo VNS.

Seu funcionamento é descrito em Fonseca e Santos (2014) e, em linhas gerais, é baseado na criação de um grafo não direcionado, bipartido, e na aplicação de uma busca em profundidade nesse grafo.

A Figura 2 ilustra um exemplo de mudanças geradas pela aplicação deste movimento.

O primeiro passo gerado originalmente pelo movimento: dado um professor, dois recursos do tipo turma/horário são escolhidos e trocados entre si. Após esta primeira modificação, o algoritmo verifica se a mudança gerou alguma inconsistência. No exemplo apresentado na Figura 2, o recurso Professor 1 é escolhido. Então, a Turma 4 de quinta-feira foi trocada

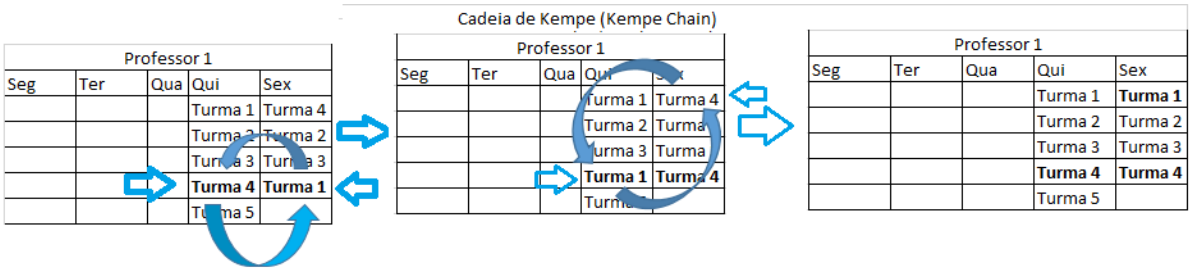


Figura 2 – Movimento Cadeia de Kempe.

pela Turma 1 de sexta-feira. Após esta troca a Turma 4 ficou com 2 aulas na sexta-feira em horários não consecutivos. Neste momento o algoritmo avalia todas as possibilidades de troca e retorna a que gera o melhor resultado, em especial, no sentido de resolver a inconsistência. É então realizada nova troca (Turma 1 de quinta-feira pela Turma 4 do primeiro horário de sexta-feira). Com esta nova mudança, a solução passou a ser viável.

4.3.2 Troca de eventos (*Meet Swap*)

Este movimento consiste em selecionar duas aulas de uma turma e realizar a troca de horários entre elas.

Dois recursos são selecionados e trocados entre si, gerando uma nova solução. A Figura 3 ilustra a operação realizada neste movimento. O recurso Turma 1 é escolhido. O recurso do segundo horário da segunda-feira foi trocado pelo recurso do terceiro horário também da segunda-feira, desta mesma turma.

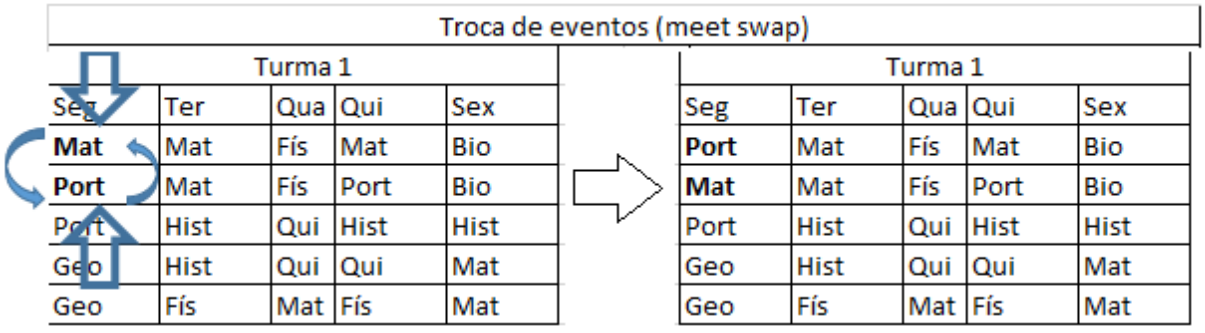


Figura 3 – Movimento de troca de eventos

4.3.3 Mudança de eventos (*Event move*)

Este movimento consiste em selecionar uma aula de uma respectiva turma e movê-la para um outro horário que esteja disponível.

Tanto um recurso quanto um horário vazio são selecionados. O recurso é, então, movido para o horário disponível, gerando-se uma nova solução. A Figura 4 mostra a operação realizada neste movimento. O recurso Professor 1 é escolhido. A Turma 5 de quinta-feira deste professor foi movida para o horário vago existente no quarto horário de sexta-feira, deste mesmo professor.

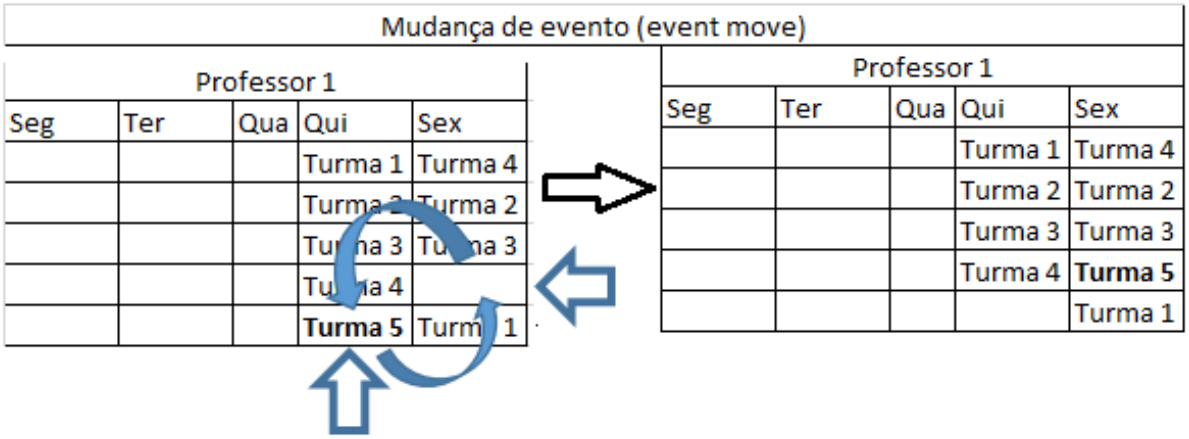


Figura 4 – Movimento de mudança de eventos.

4.3.4 Troca de eventos em bloco (*Meet Block Swap*)

Este movimento é similar ao movimento de troca de eventos, em que são selecionadas duas aulas de uma turma e seus horários são trocados. A diferença neste movimento ocorre se, para uma das aulas selecionadas, existir outra aula em um horário adjacente a ela. Se isso ocorrer, a troca envolve não somente a aula escolhida, mas, sim, suas adjacências.

Dois recursos são selecionados para serem trocados entre si. Entretanto, o algoritmo verifica se esses recursos pertencem a um bloco, ou seja, se os recursos adjacentes são os mesmos. Como fazem parte de um bloco, o algoritmo realiza a troca do bloco inteiro como mostrado. A Figura 5 mostra a troca de eventos em bloco. A Turma 1 é escolhida dentre os recursos. Então, o recurso Matemática, alocado no primeiro horário de segunda-feira, foi escolhido para ser trocado pelo recurso Português, alocado no segundo horário também da segunda-feira. Entretanto, o recurso Português possui um adjacente no terceiro horário. Assim a troca passa a envolver o bloco inteiro.

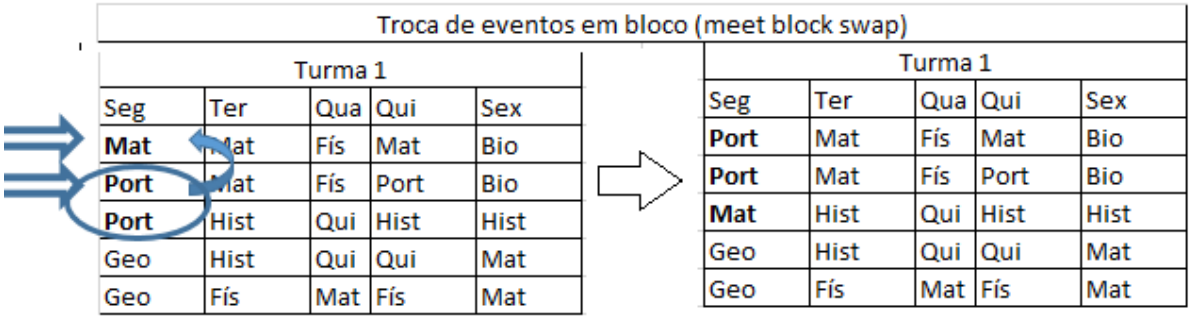


Figura 5 – Movimento de troca de eventos em bloco.

4.3.5 Troca de recursos (*Resource swap*)

Este movimento consiste em selecionar duas aulas e realizar a troca do recurso professor associado entre elas.

O algoritmo, neste movimento, primeiro seleciona dois recursos do tipo professor e, em seguida, seleciona um horário para cada um, podendo ser o mesmo horário ou horários diferentes entre si. Após, realiza a troca do recurso turma alocado entre os dois recursos professores previamente selecionados. A Figura 6 ilustra o movimento de troca de recursos. Primeiramente, Professor 1 e Professor 2 são selecionados. Em seguida, a Turma 4 de quinta-feira do Professor 1 é escolhida para ser trocada pela Turma 8 também do quarto horário, mas do Professor 2.

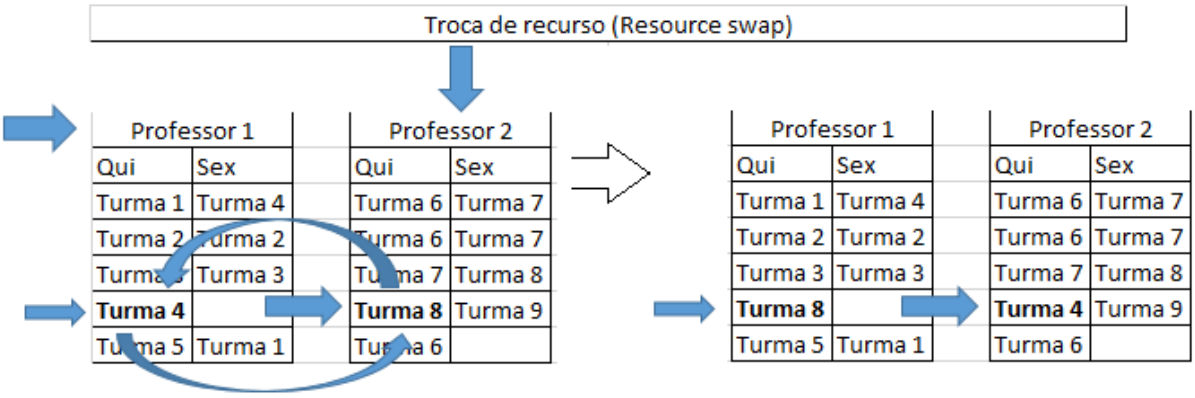


Figura 6 – Movimento de troca de recursos.

4.3.6 Mudança de recursos (*Resource move*)

Este movimento consiste em selecionar uma aula e fazer a troca de recurso dela para um outro recurso que esteja disponível no horário.

O algoritmo seleciona dois recursos do tipo professor e, em seguida, seleciona um horário para cada um, sendo o primeiro com recurso turma alocado e outro com horário vago. Em seguida, realiza a mudança do recurso turma alocado no primeiro professor para o segundo.

O movimento de mudança de recursos é apresentado na Figura 7. Os recursos Professor 1 e Professor 2 são selecionados. Em seguida, a Turma 1, alocada no quinto horário de sexta-feira do Professor 1, foi escolhida para ser movida para o horário vago disponível no quarto horário, também de sexta-feira, mas do Professor 2.

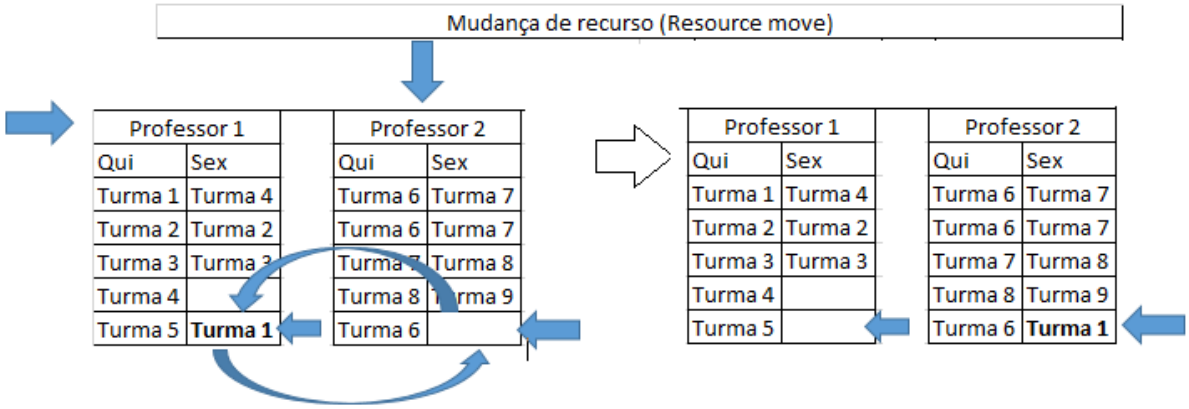


Figura 7 – Movimento de mudança de recursos.

4.4 Solução inicial

O algoritmo implementado é da classe das metaheurísticas de busca local. Desta forma, faz-se necessário utilizar como variável de entrada uma solução inicial.

Cada problema-teste utilizado oriundo da Fase 1 do ITC2011 já continha sua própria solução inicial. Além disso, a biblioteca KHE utilizada apresenta um método nativo para construção de uma solução inicial. Em geral, e respeitando algumas exceções, essa solução gerada automaticamente pela biblioteca não apresenta bons valores de função objetivo.

Em todos os testes e coletas de dados foi utilizada a solução inicial com o melhor valor de função objetivo dentre a solução fornecida junto ao problema-teste, dentro do próprio arquivo XHSTT, e a gerada pela biblioteca KHE.

Para os problemas-teste da Fase 1, a solução inicial utilizada foi a fornecida pela organização do evento, com exceção dos três problemas-teste oriundos da Austrália. Para estes e para todos os problemas-teste da Fase 2, foi utilizada a solução inicial gerada automaticamente pela biblioteca.

4.5 Abordagem proposta

A abordagem proposta neste trabalho para resolver o problema de programação de horários em escolas é o uso da metaheurística *Variable Neighborhood Search* (VNS), que é baseada em busca local.

Foram desenvolvidos cinco algoritmos, sendo quatro deles baseados na metaheurística GVNS. O funcionamento do algoritmo GVNS é bastante similar ao do VNS padrão, sendo a principal diferença entre eles a etapa de refinamento da solução, ou descida, como é conhecida, na qual é realizada a busca de ótimos locais a partir de uma dada solução. No método VNS, é utilizado um método de busca local com uma vizinhança específica, caracterizado por duas estratégias, sendo elas a pesquisa em todos os vizinhos (*best improvement*) ou busca até encontrar alguma solução de melhora (*first improvement*). Por outro lado, o algoritmo GVNS utiliza o algoritmo VND (*Variable Neighborhood Descent*), em que são utilizadas várias estruturas de vizinhança. O algoritmo VND retorna, portanto, um ótimo local com relação a todas essas estruturas.

O algoritmo GVNS gera uma nova solução a partir da solução atual e executa um procedimento de busca local. Ao concluir a busca local, é verificado se a solução encontrada é melhor que a solução atual. Se for, a solução atual é atualizada e repete-se o processo enquanto não for atingido o critério de parada. Caso contrário, a solução encontrada é descartada e repete-se o processo de geração de nova solução.

As soluções vizinhas são geradas utilizando o movimento da Cadeia de Kempe. A cada iteração em que a solução não é melhorada, o algoritmo incrementa o número de vezes que o movimento da Cadeia de Kempe é executado, até o limite máximo de dez execuções (variável $Kempe_{\max}$). Quando uma solução de melhora é encontrada, o algoritmo volta a executar o movimento apenas uma vez. Este comportamento tem como objetivo buscar soluções melhores, escapando de ótimos locais.

Nos cinco métodos foi acrescentado um limite máximo de iterações para busca local: *MaxViz*. Esta variável foi definida com o valor 1.000.000. Como a geração de vizinhos na busca local é realizada de forma determinística, em problemas-teste de pequena e média dimensão o algoritmo pode avaliar todas as possíveis soluções, retornando, assim, o ótimo local em relação à todas as vizinhanças. Entretanto, em problemas-teste de grande dimensão, o limitador de número de iterações evita que o método gaste todo o tempo disponível buscando soluções em uma única vizinhança, podendo, assim, reduzir sua efetividade.

4.6 Algoritmos desenvolvidos

O pseudocódigo do algoritmo GVNS implementado é apresentado pelo Algoritmo 2. A primeira ação do algoritmo é realizar uma busca local na solução corrente (linha 2). Em seguida, entra no primeiro laço de repetição, cujo critério de parada é o tempo de execução determinado pela variável de entrada *MaxTime* (linha 4). Dentro do laço de repetição, o algoritmo realiza k_{atual} movimentos da Cadeia de Kempe (linha 7), iniciando em 1, e em seguida realiza a busca local nesta nova solução gerada (linha 9). A solução encontrada é

comparada com a melhor solução corrente e, se for melhor, passa a ser a nova solução corrente (linha 11). Quando isso acontece, a variável que controla a quantidade de movimentos da Cadeia de Kempe é reiniciada em 1 (linha 12). Caso contrário, o algoritmo retorna ao laço de repetição, incrementando a quantidade de movimentos da Cadeia de Kempe a serem executados até o limite definido na variável $Kempe_{max}$ (linha 15).

Algoritmo 2: GVNS

Entrada: Solução inicial s_0; Tempo máximo de execução *MaxTime*; Número máximo de movimentos Kempe $Kempe_{max}$.

Saída: Melhor solução s encontrada.

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

Foram utilizadas nesta dissertação três versões diferentes do algoritmo de busca local ou algoritmo de descida, as quais são descritas e apresentadas nas subseções seguintes.

Além dos três algoritmos, que se diferenciam entre si pelo método de busca local usado, foram desenvolvidos outros dois algoritmos: SGVNS (*Skewed General Variable Neighborhood Search*) e VNS Adaptativo, que serão melhor detalhados nas seções a seguir.

4.6.1 GVNS

Esta implementação segue a estrutura do GVNS de Mladenović et al. (2008), que usa o algoritmo VND como algoritmo de descida. O acrônimo vem de *Variable Neighborhood*

Descent. Neste algoritmo, a ordem das vizinhanças é determinística, ou seja, previamente definida, conforme mostrado no Algoritmo 3.

O algoritmo funciona com dois laços de repetição, sendo o primeiro restrito à quantidade de vizinhanças r utilizadas (linha 4). O segundo laço de repetição restringe a quantidade de vizinhos a serem avaliados, de acordo com a variável de entrada *MaxViz* (linha 7). Dentro do laço de repetição, o algoritmo gera um novo vizinho aleatório, usando como movimento a estrutura de vizinhança selecionada (linha 9). Se a solução for melhor, a solução corrente é atualizada; senão, descarta-se o vizinho e retorna-se ao laço de repetição da quantidade de vizinhos. Quando tiver verificado todos os vizinhos da estrutura de vizinhança ou ao atingir o número máximo de vizinhos a serem avaliados, o algoritmo verifica se houve alguma melhora. Em caso positivo, retorna-se à primeira estrutura de vizinhança definida (linha 16) ou passa-se para a próxima estrutura de vizinhança da lista r (linha 19).

Algoritmo 3: VND

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

Saída: Melhor solução s encontrada.

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

4.6.2 GVNS/rVND

Nesse algoritmo, a busca local do GVNS é feita pelo método de busca local rVND. O nome deste algoritmo vem de *Random Variable Neighborhood Descent*. Neste algoritmo, a ordem das vizinhanças é randomizada a cada chamada do método de busca local. A vantagem deste procedimento reside na não necessidade de determinar a ordem das vizinhanças e representa, assim, um parâmetro a menos do algoritmo, conforme mostrado no Algoritmo 4. Esse algoritmo é uma adaptação proposta por Souza et al. (2010) sobre o algoritmo VND de Mladenović e Hansen (1997).

O funcionamento do algoritmo é bastante semelhante ao VND com uma diferença: o conjunto de vizinhanças é randomizado a cada chamada do algoritmo (linha 5). Com este funcionamento, são geradas sequências diferentes de execução das estruturas de vizinhança. Conforme descrito anteriormente, seu funcionamento conta com dois laços de repetição, sendo o primeiro restrito à quantidade de vizinhanças r utilizadas (linha 6). O segundo laço de repetição restringe a quantidade de vizinhos a serem avaliados, de acordo com a variável de entrada *MaxViz* (linha 8). Dentro do laço de repetição, o algoritmo gera um novo vizinho aleatório, usando, como movimento, a estrutura de vizinhança selecionada (linha 10). Se a solução for melhor, atualiza a solução corrente. Se não for, descarta o vizinho e retorna ao laço de repetição da quantidade de vizinhos. Ao atingir o número máximo de vizinhos a serem avaliados, o algoritmo verifica se houve alguma melhora. Em caso positivo, retorna à primeira estrutura de vizinhança definida (linha 16) ou passa para a

próxima estrutura de vizinhança da lista $\mathcal{N}$ (linha 19).

Algoritmo 4: RVND

Entrada: Solução inicial s_0; Conjunto N de r vizinhanças; Número máximo de iterações *MaxViz*.

Saída: Melhor solução s encontrada.

```
1  início
2      s ← s_0;
3      i ← 1;
4      N ← {N^1, N^2, ..., N^r} /* Conj. de vizinhanças*/
5      N_R ← {N_R^1, N_R^2, ..., N_R^r} /*Conj. de vizinhanças randomizado */
6      para cada N_R^i ∈ N_R faça
7          iter ← 1;
8          enquanto existir vizinho s' na vizinhança N_R^i e iter ≤ MaxViz faça
9              iter ← iter + 1;
10             s' ← próximo vizinho de s na vizinhança N_R^i;
11             se f(s') ≤ f(s) então
12                 s ← s';
13             fim
14         fim
15         se encontrou solução melhor então
16             i ← 1;
17         fim
18         senão
19             i ← i + 1;
20         fim
21     fim
22 fim
23 retorna s
```

4.6.3 VNS/BLVA

Nesse algoritmo, nomeado VNS/BLVA (VNS com Busca Local em Vizinhança Aleatória), a busca local é feita em uma única vizinhança a cada iteração, sendo essa vizinhança escolhida aleatoriamente.

O Algoritmo 5 ilustra seu funcionamento. há apenas um laço de repetição (linha 5) definido pela variável *MaxViz*, que é responsável por definir a quantidade máxima de vizinhos a serem avaliados. Dentro do laço de repetição, o algoritmo gera uma nova solução de acordo com uma estrutura de vizinhança escolhida aleatoriamente (linha 8). Caso a solução seja

melhor, atualiza-se a solução corrente (linha 10).

Algoritmo 5: BLVA

Entrada: Solução inicial s_0; Conjunto N de r vizinhanças; Número máximo de iterações *MaxViz*.

Saída: Melhor solução s encontrada.

```
1  início
2      s ← s0;
3      N ← {N^1, N^2, ..., N^r} /* Conj. de vizinhanças*/
4      iter ← 1;
5      enquanto iter ≤ MaxViz faça
6          iter ← iter + 1;
7          i ← número randômico entre 1 e r;
8          s' ← próximo vizinho de s na vizinhança N^i ∈ N;
9          se f(s') ≤ f(s) então
10             s ← s';
11         fim
12     fim
13 fim
14 retorna s
```

4.6.4 SGVNS: *Skewed General Variable Neighborhood Search*

Este algoritmo foi construído mesclando duas variações da metaheurística VNS: SVNS e GVNS.

A metaheurística Skewed VNS (SVNS) é uma variação do VNS proposta por Mladenović e Hansen (1997). Ela usa um parâmetro α para aceitar soluções que são piores que a solução atual. O conceito envolvido é que as soluções melhores podem estar muito longes da solução atual. Assim, é necessário passar por soluções intermediárias (e, talvez, piores) para alcançá-las. O algoritmo GVNS (General VNS), proposto por Mladenović et al. (2008), usa o algoritmo *Variable Neighborhood Descent* (VND) para realizar as buscas locais. O VND (HANSEN; MLADENOVIC; PÉREZ, 2008) é um método de descida que usa mudanças sistemáticas de vizinhanças para explorar o espaço de soluções. Este algoritmo retorna um ótimo local com base em todas as vizinhanças usadas e foi melhor detalhado na Seção 4.6.1.

O SGVNS proposto também usa VND como método de busca local. Adicionalmente, assim como no algoritmo SVNS, um parâmetro α é usado para aceitar soluções intermediárias que são piores que a solução atual.

Algoritmos que aceitam soluções piores podem trazer um problema em sua execução, chamado de ciclagem. Este fenômeno ocorre quando o algoritmo fica preso em uma mesma sequência de soluções. Para evitar este comportamento, foi implementada uma Lista Tabu – uma lista que armazena, de forma temporária, os valores das soluções já visitadas (resultado da função objetivo). Assim, o algoritmo previne que a mesma sequência de soluções sejam geradas novamente.

O conceito da Lista Tabu foi obtido da metaheurística Busca Tabu (GLOVER, 1986). Nesta metaheurística, a busca local utiliza uma estrutura de memória adaptativa para conduzir a busca através do espaço de soluções explorando características de boas soluções. Por outro lado, alguns atributos são registrados para evitar determinadas soluções. Este mecanismo é conhecido como Lista Tabu.

A Lista Tabu foi implementada com um tamanho fixo de elementos (conforme parâmetro de entrada do algoritmo) e usando o protocolo FIFO (*First In First Out*). Isso significa que os elementos são inseridos sequencialmente até atingir o tamanho da lista. Quando o tamanho da lista é atingido, o próximo elemento sobresceverá o primeiro inserido e assim por diante.

O pseudocódigo do SGVNS é descrito no Algoritmo 6. Na linha 12 é verificado se a nova solução será considerada ou não, de acordo com o parâmetro α e a lista de valores de

soluções já geradas.

Algoritmo 6: SGVNS

Entrada: Solução inicial (s_0); Tempo máximo de execução (*MaxTime*); Número máximo de movimentos da cadeia de Kempe (*Kempe*$_{\max}$); Percentual para aceitar soluçoes piores (α); Tamanho da lista Tabu.

Saída: Melhor solução s encontrada.

```
1  início
2      s ← s_0;
3      s' ← s_0;
4      s_temp ← s;
5      k_atual ← 1;
6      Insere f(s) na Lista Tabu;
7      enquanto time ≤ MaxTime faça
8          para k = 0; k < k_atual faça
9              s' ← vizinho de s' usando o movimento da Cadeia de Kempe;
10         fim
11         s' ← Busca local usando o algoritmo VND(s');
12         se f(s') < (1 + α) × f(s) e f(s') ∉ Lista Tabu então
13             s_temp ← s';
14             k_atual ← 1;
15             Insere f(s') na Lista Tabu;
16             if f(s') ≤ f(s) then
17                 s ← s';
18             end
19         fim
20         senão
21             s' ← s_temp;
22             se k_atual < Kempe_max então
23                 k_atual ← k_atual + 1;
24             fim
25         fim
26     fim
27 fim
28 retorna s
```

4.6.5 VNS Adaptativo

Algoritmos adaptativos são aqueles que modificam seu comportamento durante sua execução, levando em conta as mudanças que ocorrem durante seu funcionamento.

O algoritmo VNS Adaptativo proposto é uma variação da metaheurística clássica VNS, na qual as vizinhanças usadas para a busca local são escolhidas de acordo com suas respectivas probabilidades, as quais representam o sucesso na busca local em iterações anteriores.

O princípio envolvido é que as vizinhanças que geram melhores soluções poderiam ter probabilidades maiores que as que não geram boas soluções. Para evitar uma convergência prematura do algoritmo e vício em algumas vizinhanças, quando uma solução melhor não é encontrada após um número pré-determinado de iterações, as probabilidades são reiniciadas.

A implementação é descrita no Algoritmo 7. Na linha 8, as probabilidades iniciais são definidas na variável *probneighborhood* com valor $1/|N|$ ($0,2$ no caso desta dissertação, que são cinco vizinhanças para explorar o espaço de soluções). A vizinhança usada para realizar a busca local, de acordo com as probabilidades atuais, é escolhida na linha 16. As probabilidades de todas as vizinhanças são recalculadas a cada *itercalc* iterações (linha 43 do algoritmo 7).

Algoritmo 7: VNS ADAPTATIVO

Entrada: Solução inicial s_0; Tempo máximo de execução (*MaxTime*); Número máximo de movimentos da cadeia de Kempe ($Kempe_{\max}$); Quantidade de iterações para recalcular as probabilidades (*itercalc*); Conjunto $|N|$ de vizinhanças N; Número de recálculos de probabilidades sem melhora para reiniciar as probabilidades (*IterRestart*).

Saída: Melhor solução s.

```
1  início
2      s ← s_0;
3      s' ← s_0;
4      k_atual ← 1;
5      melhora ← 0;
6      numero_itercalc ← 1;
7      para para cada vizinhança N faça
8          probneighborhood ← 1/|N|;
9      fim
10     iter ← 1;
11     enquanto time ≤ MaxTime faça
12         para k = 0; k < k_atual faça
13             s' ← vizinho de s' aplicando o movimento da cadeia de Kempe;
14         fim
15         prob ← número aleatório entre 0 e 1;
16         l ← vizinhança escolhida de acordo com as probabilidades probneighborhood e número randômico prob;
17         s' ← busca local usando vizinhança N^l(s') ;
18         se f(s') ≤ f(s) então
19             s ← s';
20             k_atual ← 1;
21             melhora ← 1;
22         fim
23         senão
24             s' ← s;
25             if melhora = 1 then
26                 k_atual ← 1;
27                 melhora ← 0;
28             end
29             senão se k_atual ≤ Kempe_max então
30                 k_atual ← k_atual + 1;
31             fim
32         fim
33         se resto da divisão de iter por itercalc igual 0 então
34             numero_itercalc ← numero_itercalc + 1;
35             se numero_itercalc ≥ IterRestartandmelhora = 0 então
36                 para cada vizinhança N^l of N faça
37                     probneighborhood(N^l) ← 1/|N|;
38                 fim
39                 numero_itercalc ← 1;
40             fim
41             senão
42                 for cada vizinhança N^l of N do
43                     Atualizaprobneighborhood(N^l)
44                 end
45             fim
46         fim
47         iter ← iter + 1;
48     fim
49 fim
50 retorna s
```

Capítulo 5

Experimentos computacionais

Neste capítulo, serão apresentados os detalhes em que os experimentos foram realizados bem como os resultados obtidos. Em seguida será mostrado como estes resultados foram avaliados de forma estatística, comparando os algoritmos entre si e com o *Goal Solver*.

5.1 Descrição geral

Os algoritmos foram implementados em C++, utilizando a IDE Code::Blocks. Todos os testes foram realizados em um notebook Intel Core i5, com 8 GB de RAM, rodando sistema operacional Windows 10. A análise dos parâmetros usando a ferramenta iRace foi realizada no mesmo equipamento, mas rodando a plataforma Linux em sua distribuição Ubuntu versão 16.10.

A terceira e mais recente edição da competição internacional de problemas de *timetabling* – ITC foi dedicada a problemas do modelo *High School Timetabling*, no ano de 2011. As duas competições organizadas anteriormente, respectivamente em 2002 e 2007, apresentaram temas específicos e distintos.

O ITC 2011 foi dividido em 3 etapas, sendo elas:

- Fase 1: os próprios competidores eram responsáveis pela geração das melhores soluções para os problemas-teste públicos, sem restrições de tempo e de recursos computacionais;
- Fase 2: os organizadores realizavam a execução de cada algoritmo participante nas mesmas condições, considerando o tempo de 1000 segundos e usando problemas-teste específicos, não divulgados anteriormente;
- Fase 3: os próprios competidores realizaram a geração de soluções em um conjunto oculto de problemas-teste e, como na Fase 1, não foram definidas restrições de tempo

e tecnologias. Somente os 5 melhores da Fase 2 chegaram à esta fase.

Assim como na competição ITC2011, foram utilizadas como base para a validação dos algoritmos implementados os mesmos vinte e um problemas-teste usados na Fase 1 e dezoito problemas-teste usados na Fase 2, todos de domínio público atualmente. Estes são originários de vários países e possuem características bastante diversificadas, sendo alguns simples e outros com alto volume de turmas e professores, tornando-os altamente complexos. Os problemas-teste são apresentados nas Tabelas 1 e 2. Nestas tabelas são apresentados, em cada coluna:

- Problema teste: nome identificador de cada problema teste;
- Horários: quantidade de horários disponíveis que deverão ter professores alocados para cada turma;
- Professores.: quantidade de professores, ou recursos, a serem alocados em cada turma e horário;
- Salas: quantidade de salas de aula que poderão ser alocadas para cada combinação horário/turma. Esta informação não é muito comum de ser vista na classe de problemas *school timetabling*. Nesta classe de problemas, a sala de aula é fixa, associada à turma. Assim como na competição ITC2011 e na implementação do algoritmo *Goal Solver*, não está sendo utilizada nos algoritmos implementados nesta dissertação;
- Turmas: quantidade de turmas que deverão ter professores e horários alocados;
- Aulas: quantidade de eventos, ou aulas, dada pelo somatório da quantidade de horários de cada turma.

Tabela 1 – Problemas-teste da fase 1 - ITC 2011

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

Tabela 2 – Problemas-teste da fase 2 - ITC 2011

	Problema-teste	Horários	Profs	Salas	Turmas	Aulas
1	BrazilInstance2	25	14	-	6	150
2	BrazilInstance3	25	16	-	8	200
3	BrazilInstance4	25	23	-	12	300
4	BrazilInstance6	25	30	-	14	350
5	FinlandElementarySchool	35	22	21	291	445
6	FinlandSecondarySchool2	40	22	21	469	566
7	Aigio1stHighSchool2010-2011	35	37	-	208	532
8	ItalyIntance4	36	61	-	38	1101
9	KosovaInstance1	62	101	-	63	1912
10	Kottenpark2003	38	75	41	18	1203
11	Kottenpark2005A	37	78	42	26	1272
12	Kottenpark2008	40	81	11	34	1118
13	Kottenpark2009	38	93	53	48	1301
14	Woodlands2009	42	40	-	-	1353
15	SpanishSchool	35	66	4	21	439
16	WesternGreeceUniversity3	35	19	-	6	210
17	WesternGreeceUniversity4	35	19	-	12	262
18	WesternGreeceUniversity5	35	18	-	6	184

5.2 Definição dos valores dos parâmetros

Vários experimentos foram realizados com o objetivo de encontrar a melhor configuração de parâmetros para cada um dos algoritmos implementados e testados.

Os algoritmos GVNS, GVNS/rVND e VNS/BLVA não possuem parâmetros variáveis em sua implementação, por isso não há necessidade de validação.

Já os parâmetros dos algoritmos SGVNS e VNS Adaptativo foram definidos com a ajuda do pacote *iRace* (<http://iridia.ulb.ac.be/irace/>). Esta ferramenta implementa o procedimento *Iterated Racing* (LÓPEZ-IBÁNEZ et al., 2011) e é uma extensão do *Iterated F-race (I/F-Race)* proposto por Birattari, Balaprakash e Dorigo (2007). A função principal do *iRace* é automatizar e otimizar a busca pela configuração mais adequada dos parâmetros de um determinado algoritmo no sentido de obter os melhores resultados. *iRace* é implementado como uma biblioteca e extensão do software R (TEAM et al., 2014).

A análise *iRace* foi realizada com um total de 3.000 execuções para cada um dos dois algoritmos (VNS Adaptativo e SGVNS). Devido à extensa duração dos testes, não foi usado o mesmo tempo de 1.000 segundos como critério de parada. Em seu lugar, foram usados três diferentes tempos como critério de parada para cada algoritmo em cada problema-teste: 10 segundos, 30 segundos e 60 segundos, definidos arbitrariamente. As tabelas 3 e 4 mostram os valores dos parâmetros testados pelo *iRace* em relação aos algoritmos SGVNS e VNS Adaptativo, respectivamente.

Tabela 3 – Valores dos parâmetros avaliados pelo *iRace* no algoritmo SGVNS.

Parâmetro	Valores				
α	0.001	0.01	0.025	0.05	0.075
$Kempe_{\max}$	1	5	10	-	-
Tamanho da Lista Tabu	5	7	10	15	-

Tabela 4 – Valores dos parâmetros avaliados pelo *iRace* no algoritmos VNS Adaptativo.

Parâmetro	Valores			
Qtd de iterações para reiniciar probabilidades	5	10	15	20
$Kempe_{\max}$	1	5	10	-
itercalc	100	250	500	750

O *iRace* indicou os melhores valores para esses parâmetros como mostrado na Tabela 5. Esses valores foram encontrados durante a execução com o tempo de 60 segundos.

5.3 Resultados

Como dito anteriormente, tanto na Fase 1 quanto na Fase 2 do ITC 2011, os testes foram realizados sem restrições computacionais de tempo nem de ambiente. Para fins de compa-

Tabela 5 – Melhores valores para os parâmetros indicados pelo *iRace*.

	α	$Kempe_{\max}$	Tamanho da Lista Tabu
SGVNS	0.025	5	10
	itercalc	$Kempe_{\max}$	Qtd de iterações para reiniciar probabilidades
VNS Adaptativo	500	1	5

ração, na presente dissertação foi utilizado o mesmo tempo de 1000 segundos para cada problema-teste, mesmo valor utilizado pelo algoritmo *Goal Solver*, vencedor da competição. Para os problemas teste da primeira fase, além do tempo de execução, foram utilizados, também, as mesmas soluções iniciais de cada problema-teste, sendo essas soluções divulgadas pela organização do evento e disponíveis juntamente com cada problema-teste em seus respectivos arquivos XHSTT. Os três problemas-teste oriundos da Austrália (AustraliaBGHS98, AustraliaSAHS96 e AustraliaTES99) apresentam uma característica em comum: a solução inicial divulgada pelo ITC é inferior à solução inicial obtida através da execução do método disponível na biblioteca KHE. Para esses três problemas-teste em particular, portanto, a solução inicial utilizada (também pelo algoritmo Goal Solver) foi a melhor solução, desconsiderando a solução disponibilizada pela organização da competição. Para os problemas-teste da Fase 2, a solução inicial utilizada foi sempre a gerada pela função disponível na biblioteca KHE.

Dos vinte e um problemas-teste apresentados na Fase 1 do evento, cinco deles já apresentavam solução inicial ótima. A otimalidade foi demonstrada pelo resultado da função objetivo 0/0, ou seja, todas as restrições fortes e fracas foram atendidas. Nestes casos, portanto, não fazia sentido avaliá-las, são eles:

- FinlandArtificialSchool;
- FinlandCollege;
- GreeceHighSchool1;
- GreecePatras3rdHS2010;
- GreecePreveza3dHS2008.

Dos dezesseis problemas-teste da Fase 1 restantes, todos foram avaliados considerando as condições descritas acima. Os resultados são apresentados na Tabela 6. Os dezoito problemas-teste da Fase 2 também foram avaliados nas mesmas condições anteriormente mencionadas e os resultados são apresentados na Tabela 7.

Nestas tabelas são mostrados:

- O valor da solução inicial divulgado pela organização do evento (os valores entre parênteses mostram o valor obtido pela biblioteca KHE, quando este for melhor);
- O valor retornado pelo *Goal Solver* executado na mesma máquina dos testes;
- O valor do algoritmo GVNS;
- O valor do algoritmo GVNS utilizando o algoritmo rVND;
- O valor do algoritmo GVNS utilizando o algoritmo BLVA;
- O valor do algoritmo SGVNS;
- O valor do algoritmo VNS Adaptativo.

Os valores apresentados entre parênteses em cada algoritmo implementado mostram o *gap* percentual entre o algoritmo usado e o *Goal Solver*, calculado pela Equação (1):

$$gap = \frac{f^{\star}(algoritmo) - f^{\star}(Goal)}{f^{\star}(Goal)} \tag{1}$$

sendo $f^{\star}(algoritmo)$ o resultado alcançado pelo algoritmo usado no respectivo problema-teste e $f^{\star}(Goal)$ o alcançado pelo *Goal Solver*. Quando os valores de avaliação das restrições fortes são diferentes, o *gap* é calculado apenas com base nesses valores. Nos demais casos são usados somente os valores das restrições fracas. Resultados negativos de *gap* indicam que o algoritmo implementado foi melhor que o algoritmo *Goal Solver*. Os valores mostrados em negrito indicam qual o algoritmo dentre os seis (*Goal Solver* e os cinco algoritmos implementados) obteve o melhor resultado.

Tabela 6 – Comparação de resultados entre os algoritmos implementados e o GOAL Solver nos problemas teste da Fase 1

Problema-teste	Solução Inicial ITC 2011	*Goal Solver* (SA + ILS)	GVNS	GVNS rVND	VNS BLVA	SGVNS	VNS Adaptativo
AustraliaBGHS98	7/433 (6/450)	6/450	4/370 (-0,33)	4/364 (-0,33)	6/448 (-0,44)	**1/432 (-0,83)**	7/433 (0,16)
AustraliaSAHS96	23/44 (17/550)	14/50	12/51 (-0,14)	**12/47 (-0,14)**	14/49 (-0,02)	13/46 (-0,07)	17/53 (0,17)
AustraliaTES99	26/134 (7/163)	7/161	**7/151 (-0,06)**	**7/151 (-0,06)**	7/161 (0,00)	7/163 (0,01)	7/163 (0,01)
BrazilInstance1	0/24	0/12	**0/11 (-0,08)**	0/12 (0,00)	**0/11 (-0,08)**	0/12 (0,00)	0/12 (0,00)
BrazilInstance4	0/112	0/91	0/94 (0,03)	0/94 (0,03)	0/94 (0,03)	**0/9 (-0,01)**	0/100 (0,09)
BrazilInstance5	0/225	0/164	0/155 (-0,05)	0/161 (-0,01)	0/158 (-0,03)	**0/149 (-0,09)**	0/173 (0,05)
BrazilInstance6	0/209	0/149	0/148 (-0,67)	0/148 (-0,67)	0/137 (-0,08)	**0/131 (-0,11)**	0/162 (0,09)
BrazilInstance7	0/330	0/264	0/249 (-0,05)	0/252 (-0,04)	**0/240 (-0,05)**	0/248 (-0,06)	0/282 (0,06)
EnglandStPaul	0/18.444	0/18.092	0/12.542 (-0,30)	0/14.564 (-0,19)	0/18.418 (0,01)	**0/12.466 (-0,31)**	0/18.418 (0,01)
FinlandHighSchool	0/1	**0/1**	**0/1 (0,00)**	**0/1 (0,00)**	**0/1 (0,00)**	**0/1 (0,00)**	**0/1 (0,00)**
FinlandSecondarySchool	0/106	**0/86**	0/87 (0,01)	0/87 (0,01)	**0/86 (0,00)**	0/90 (0,04)	0/87 (0,01)
ItalyInstance1	0/28	0/19	**0/18 (-0,05)**	0/19 (0,00)	**0/18 (-0,05)**	**0/18 (-0,05)**	0/19 (0,00)
NetherlandsGEPRO	1/566	1/566	**1/434 (-0,23)**	1/464 (-0,18)	1/441 (-0,22)	1/441 (-0,22)	1/545 (-0,03)
NetherlandsKottenpark2003	0/1.410	0/1.409	**0/1.216 (-0,13)**	0/1.248 (-0,11)	0/1.343 (-0,04)	0/1.281 (-0,09)	0/1.372 (-0,02)
NetherlandsKottenpark2005	0/1.078	0/1.078	**0/881 (-0,18)**	0/972 (-0,09)	0/1.016 (-0,05)	0/894 (-0,17)	0/1078 (0,00)
SouthAfricaLewitt2009	0/58	0/22	0/24 (0,09)	0/24 (9,09)	**0/20 (-9,09)**	0/24 (0,09)	0/50 (1,27)

Tabela 7 – Comparação de resultados entre os algoritmos implementados e o GOAL Solver nos problemas-teste da Fase 2

Problema-teste	Solução Inicial KHE	*Goal Solver* (SA + ILS)	GVNS	GVNS rVND	VNS BLVA	SGVNS	VNS Adaptativo
BrazilInstance2	4/96	1/39	1/38 (-2,56)	1/39 (0,0)	**1/35 (-10,26)**	**1/35 (-10,26)**	1/44 (12,82)
BrazilInstance3	1/195	0/90	0/81 (-10,0)	0/69 (-23,33)	0/78 (-13,33)	**0/60 (-33,33)**	0/105 (16,67)
BrazilInstance4	39/144	**17/96**	**17/96 (0,0)**	17/102 (6,25)	17/108 (12,50)	**17/96 (0,0)**	18/102 (6,25)
BrazilInstance6	13/303	**3/220**	4/210 (33,33)	4/213 (33,33)	3/230 (4,55)	3/233 (5,91)	4/237 (33,33)
FinlandElementarySchool	11/4	**0/3**	**0/3 (0,0)**	**0/3 (0,0)**	**0/3 (0,0)**	**0/3 (0,0)**	**0/3 (0,0)**
FinlandSecondarySchool2	3/1319	**0/0**	0/10 (-)	0/10 (-)	0/10 (-)	0/10 (-)	0/15 (-)
Aigio1stHighSchool2010-2011	10/691	0/121	1/115 (-)	**0/80 (-33,88)**	0/110 (-9,09)	**0/80 (-33,88)**	0/110 (-8,26)
ItalyInstance4	36/22762	0/1340	0/598 (-55,37)	0/393 (-70,67)	**0/306 (-77,16)**	0/410 (-69,40)	0/1526 (13,88)
KosovaInstance1	1261/656	**0/102**	27/394 (-)	0/319 (212,75)	0/310 (203,92)	48/540 (-)	152/365 (-)
Kottenpark2003	4/7502	**0/10725**	1/43678 (307,25)	0/24643 (129,77)	0/24799 (131,23)	1/48179 (349,22)	2/38735 (261,17)
Kottenpark2005A	37/30241	30/35454	31/2329 (3,33)	**29/11038 (-3,33)**	33/24077 (10,00)	32/24588 (6,67)	36/29177 (20,00)
Kottenpark2008	61/176009	24/19231	**12/25999 (-50,0)**	12/34859 (-50,0)	26/30884 (8,33)	13/30014 (-45,83)	29/36946 (20,83)
Kottenpark2009	57/23453	31/2536	28/44505 (-9,68)	**20/14855 (-35,48)**	37/28855 (19,35)	31/19805 (0,0)	46/17137 (48,39)
Woodlands2009	19/1	2/13	**2/10 (-23,08)**	**2/10 (-23,08)**	**2/10 (-23,08)**	**2/10 (-23,08)**	**2/10 (-23,08)**
SpanishSchool	3/455	0/1032	**0/1012 (-1,94)**	0/1052 (1,94)	0/1087 (5,33)	0/1081 (4,75)	0/1322 (28,10)
WesternGreeceUniversity3	0/30	**0/5**	0/6 (20,0)	**0/5 (0,0)**	**0/5 (0,0)**	0/6 (20,0)	0/6 (20,0)
WesternGreeceUniversity4	0/41	0/7	0/9 (28,57)	0/9 (28,57)	**0/6(-14,29)**	0/13 (85,71)	0/7 (0,0)
WesternGreeceUniversity5	17/44	**0/0**	**0/0 (0,0)**	**0/0 (0,0)**	**0/0 (0,0)**	**0/0 (0,0)**	**0/0 (0,0)**

Dos dezesseis problemas-teste da Fase 1 e dos dezoito problemas-teste da Fase 2 avaliados, em somente dois da Fase 1 e em três da Fase 2 nenhum dos cinco algoritmos conseguiu vencer o algoritmo *Goal Solver*. São eles:

- Fase 1:
  - Finland High school;
  - Finland Secondary school.
- Fase 2:
  - BrazilInstance6;
  - KosovaInstance1.
  - NetherlandsKottenpark2003

Na análise dos problemas-teste da Fase 1, dos quatorze problemas-teste restantes, o algoritmo GVNS venceu ou igualou seu resultado ao Goal Solver em sete. O algoritmo GVNS/rVND venceu ou igualou seus resultados a outro algoritmo em três problemas-teste. O algoritmo VNS/BLVA venceu em dois problemas-teste e teve o melhor resultado equivalente a outro método em outros quatro. O algoritmo SVGNS venceu ou igualou seus resultados em 6 problemas-teste. Por último, o algoritmo VNS Adaptativo igualou o melhor resultado em somente dois problema-teste.

Já na análise dos problemas-teste da Fase 2, dos quinze problemas-teste em que pelo menos um dos algoritmos foi melhor que o Goal Solver, o algoritmo GVNS obteve sozinho o melhor resultado em três problemas-teste e igualou o melhor resultado em outros três. O algoritmo GVNS/rVND foi o que alcançou os melhores resultados nos problemas-teste desta fase, vencendo ou igualando o melhor resultado em oito problemas-teste. Os algoritmos VNS/BLVA e SGVNS alcançaram, cada um, sete melhores resultados e por último, o algoritmo VNS Adaptativo obteve o melhor resultado somente em um problema-teste e igualou o melhor resultado em outros dois.

O gráfico da Figura 8 mostra o número de problemas-teste em que foi obtido o melhor resultado em cada algoritmo implementado, nos problemas-teste da Fase 1. Em caso de empate, são contabilizados em todos os algoritmos que obtiveram o melhor valor. Nota-se pelos resultados que o método GVNS apresentou resultados superiores aos demais, seguido pelo algoritmo SGVNS. Nota-se também que o procedimento de randomizar a relação de vizinhanças implementadas no método GVNS/rVND não traz muitos benefícios para a busca local, visto que a utilização deste método não conseguiu vencer efetivamente em nenhum problema-teste. Outra observação que pode ser feita com relação a esses

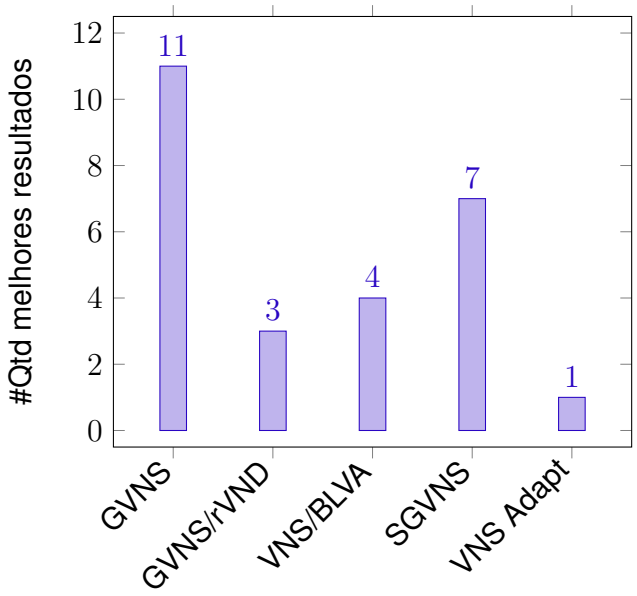


Figura 8 – Resultados por algoritmo: Fase 1

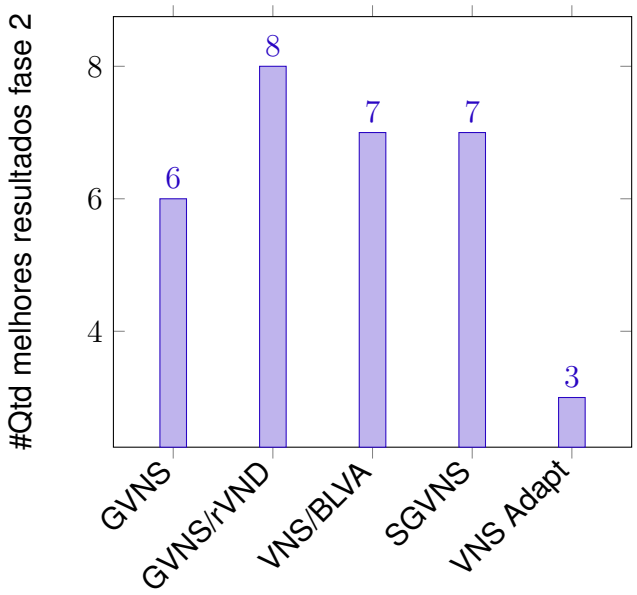


Figura 9 – Resultados por algoritmo: Fase 2

resultados é que o uso das probabilidades para cada vizinhança usado no método VNS Adaptativo também não trouxe grandes benefícios.

O gráfico da Figura 9 mostra o número de problemas-teste onde foi obtido o melhor resultado em cada algoritmo implementado nos problemas-teste da Fase 2. Assim como no Gráfico 8, em caso de empate, são contabilizados em todos os algoritmos que obtiveram o melhor valor. Nos problemas-teste desta fase, nota-se uma equivalência muito grande nos algoritmos GVNS. Com exceção do algoritmo VNS Adaptativo, os demais venceram em praticamente o mesmo número de problemas-teste, com uma leve superioridade do algoritmo GVNS/rVND.

5.4 Análise estatística dos resultados

De posse dos dados coletados utilizando cada um dos algoritmos em todos os dezesseis problemas-testes da Fase 1 e dos dezoito problemas-teste da Fase 2, foram realizados testes estatísticos, com o objetivo de avaliar se existe diferença estatística significativa entre os cinco algoritmos implementados e o algoritmo *Goal Solver*. Para isto, foi utilizado a ferramenta R (*RStudio, versão 1.0.153 2017*).

Cada problema-teste em cada algoritmo foi executado trinta vezes, sendo inicialmente avaliada a normalidade dos dados coletados para decisão sobre qual teste estatístico poderia ser utilizado (paramétrico ou não paramétrico). Esta análise foi realizada utilizando todos os trinta valores coletados e o teste de Shapiro-Wilk (MONTGOMERY, 2017), além da análise gráfica baseada em gráficos do tipo *boxplot*.

O *boxplot* ou diagrama de caixa é uma ferramenta gráfica que permite visualizar a distribuição e valores discrepantes (*outliers*) dos dados. Através desta ferramenta é possível avaliar de forma visual, além de outras coisas, a normalidade dos dados, usando para isso, a simetria do gráfico. A linha dentro do retângulo representa a medida da mediana. Para distribuições normais é esperado que a mediana fique no centro do retângulo.

Os gráficos do tipo *boxplot* dos problemas-teste da fase 1 foram agrupados conforme apresentado nas Figuras 10, 11, 12 e 13. Os gráficos do tipo *boxplot* dos problemas-teste da fase 2 também foram agrupados e podem ser vistos nas Figuras 14, 15, 16, 18 e 18. Os dados não apresentaram distribuição estatística normal como pode ser verificado pelo p-valor retornado na análise de *Shapiro-Wilk* e pela análise gráfica dos *boxplot*.

Em todos os gráficos apresentados nas Figuras 10, 11, 12, 14, 15, 16, 18 e 18, o eixo X representa o algoritmo avaliado e o eixo Y representa os valores obtidos pela função objetivo em cada algoritmo.

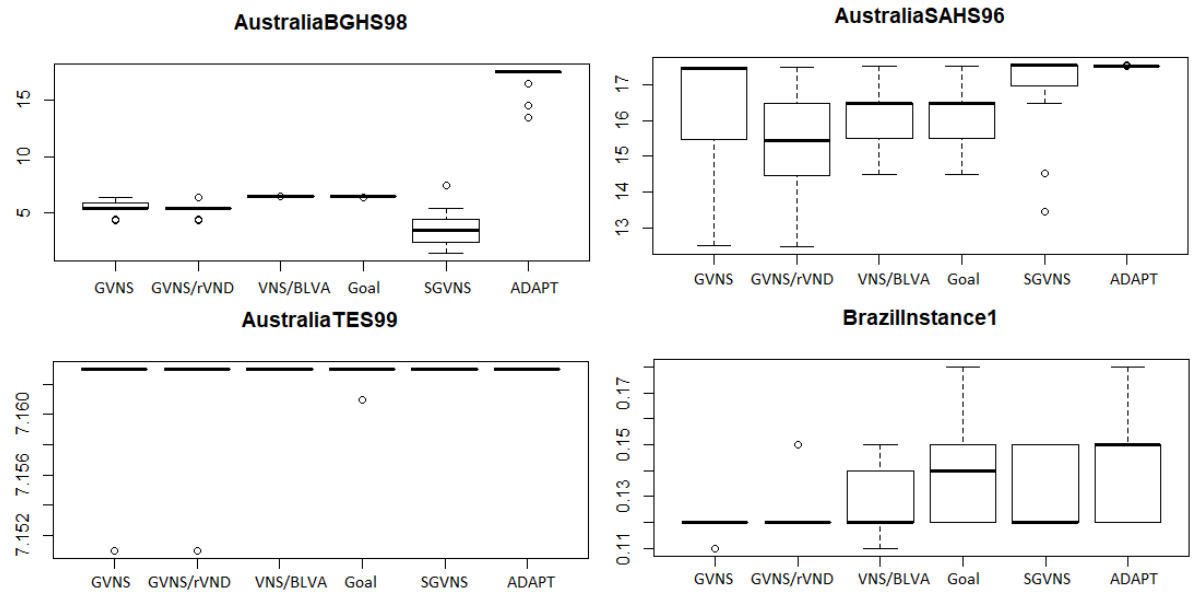


Figura 10 – Teste de normalidade do grupo 1 de problemas-teste da Fase 1

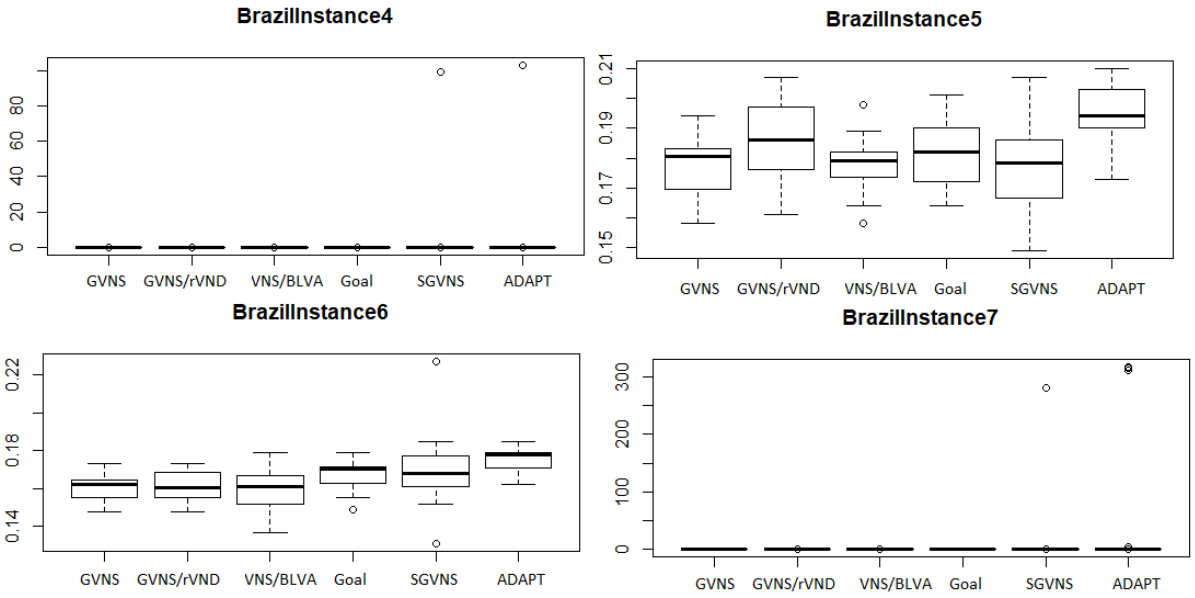


Figura 11 – Teste de normalidade do grupo 2 de problemas-teste da Fase 1

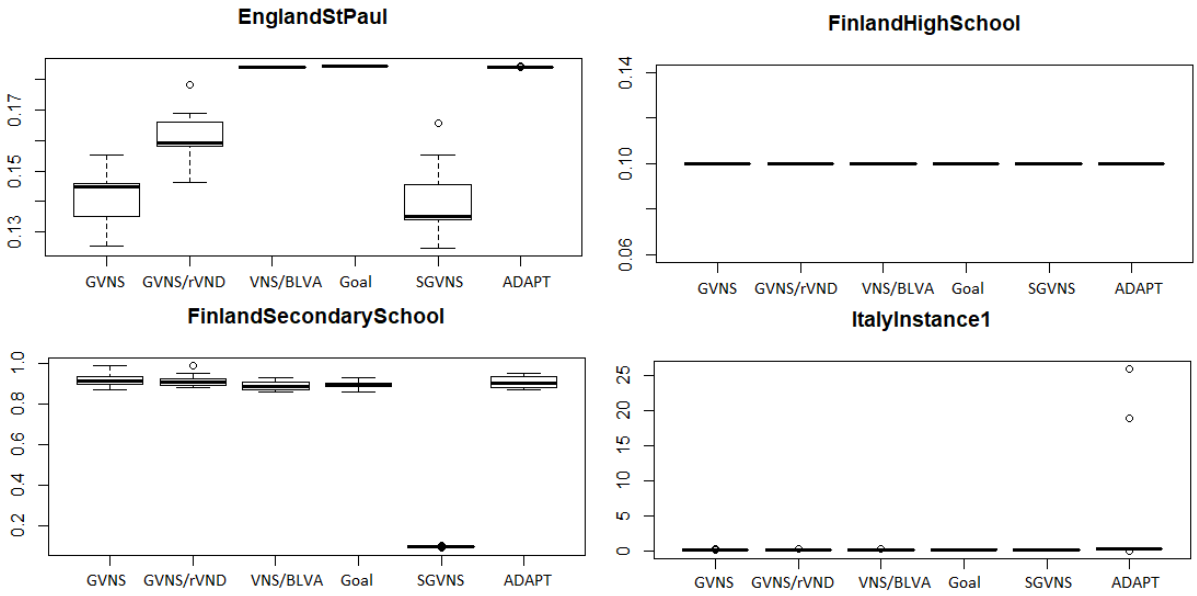


Figura 12 – Teste de normalidade do grupo 3 de problemas-teste da Fase 1

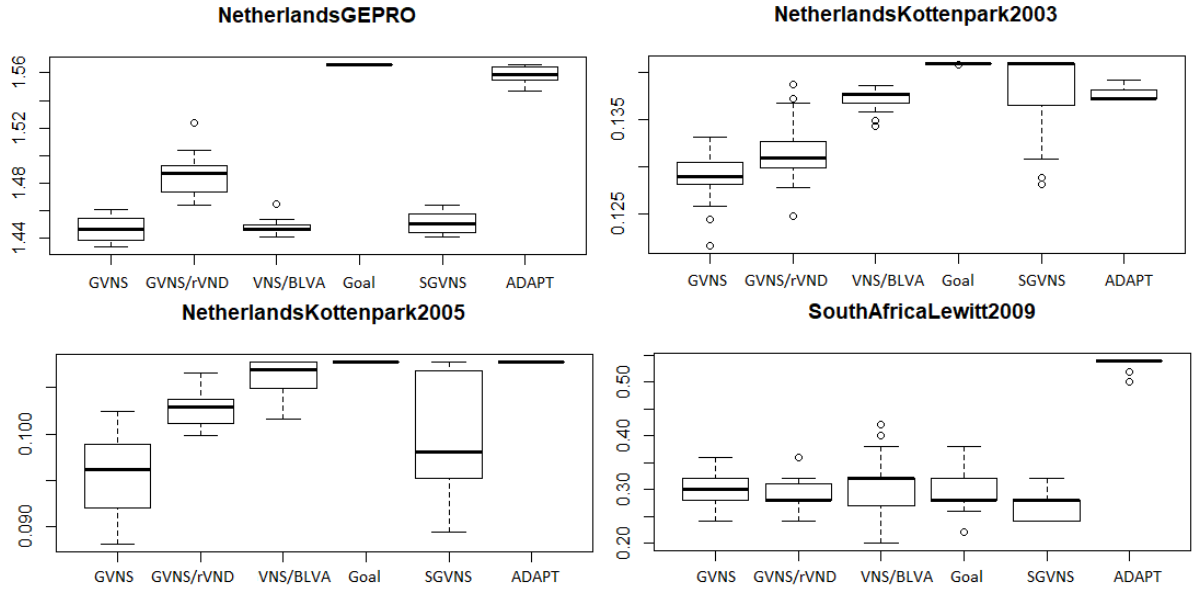


Figura 13 – Teste de normalidade do grupo 4 de problemas-teste da Fase 1

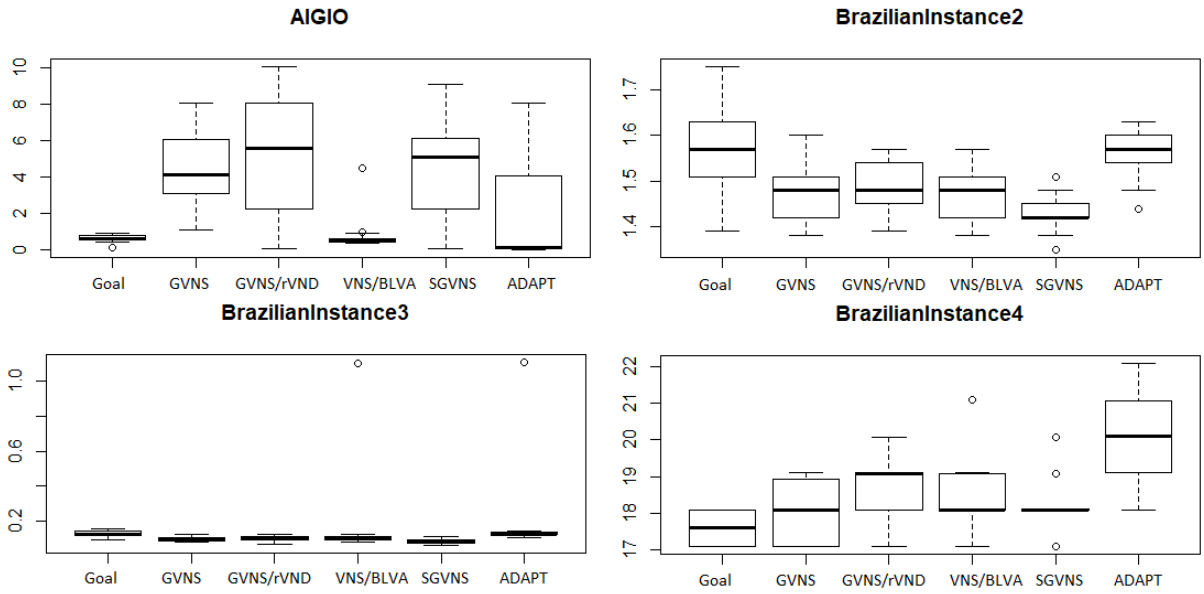


Figura 14 – Teste de normalidade do grupo 1 de problemas-teste da Fase 2

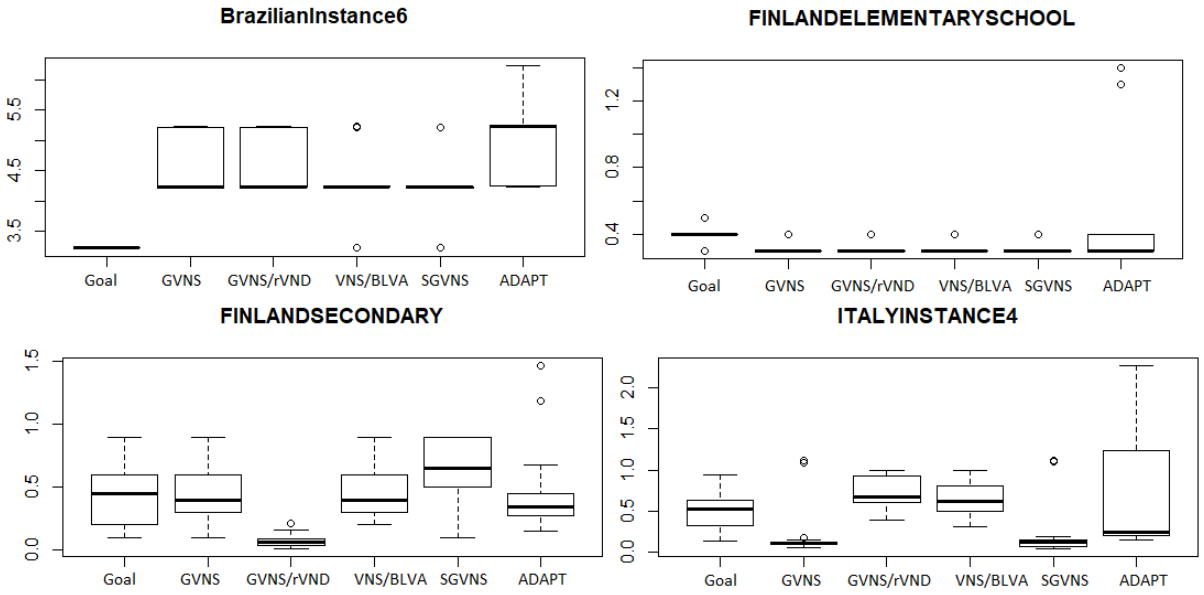


Figura 15 – Teste de normalidade do grupo 2 de problemas-teste da Fase 2

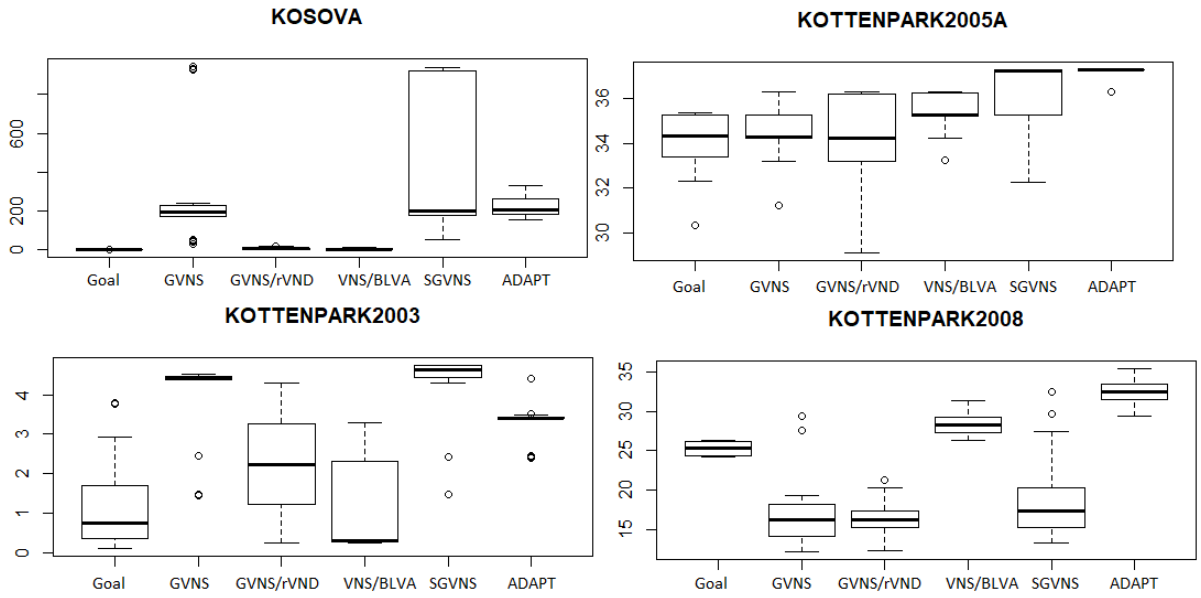


Figura 16 – Teste de normalidade do grupo 3 de problemas-teste da Fase 2

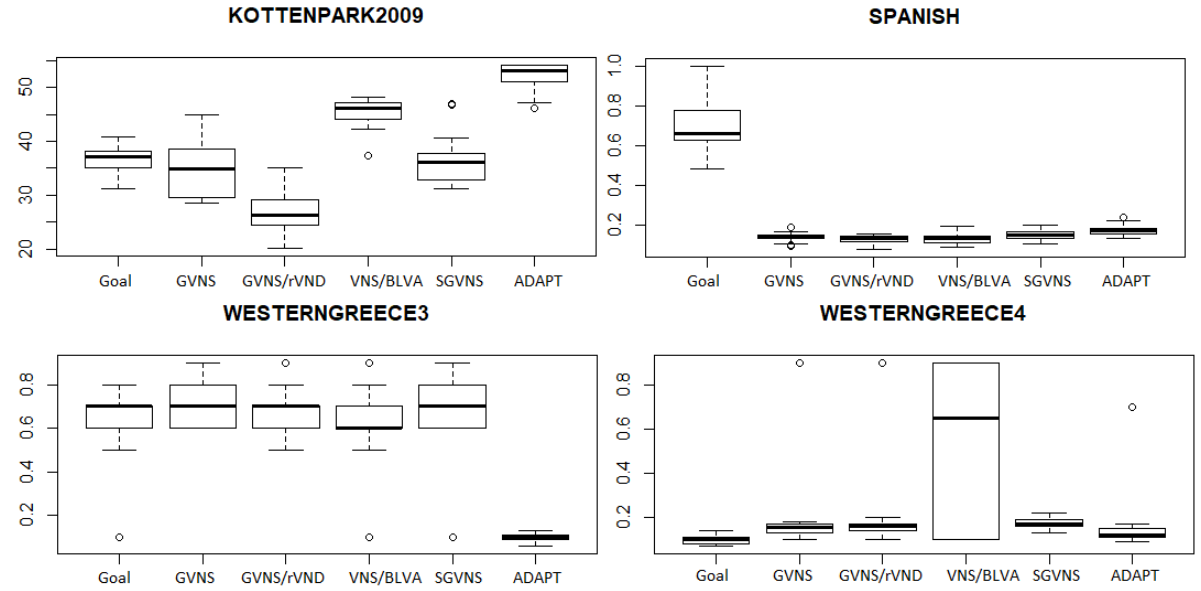


Figura 17 – Teste de normalidade do grupo 4 de problemas-teste da Fase 2

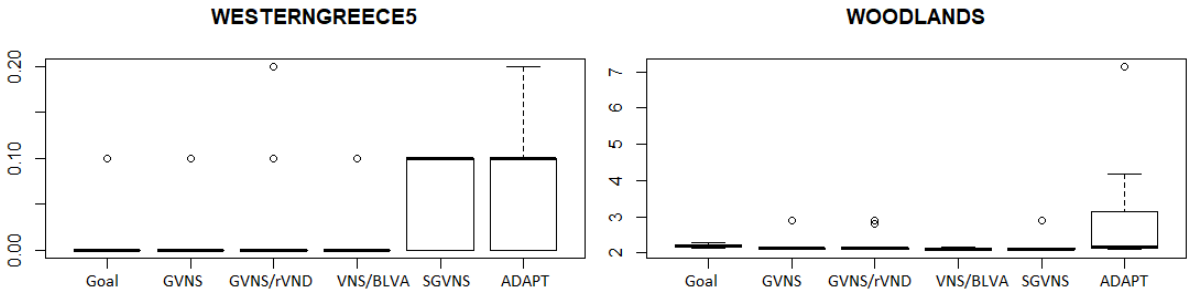


Figura 18 – Teste de normalidade do grupo 5 de problemas-teste da Fase 2

De posse desta informação, foi realizado o teste comparativo utilizando-se o teste de *Friedman* (MONTGOMERY, 2017). Este teste estatístico é semelhante ao teste da análise de variância (ANOVA) para detectar diferenças de tratamentos em experimentos que podem ser agrupados em blocos. Isso significa que o teste estatístico irá avaliar a influência de cada algoritmo (tratamento) em cada problema-teste (bloco) de forma separada. Entretanto, o teste de *Friedman* é um teste não paramétrico, ou seja, não exige como pré-requisito a normalidade dos dados, ao contrário do teste ANOVA.

Foram realizados dois testes a partir dos dados coletados: 1) Utilizando-se as médias das amostras coletadas e 2) Utilizando-se o menor valor entre as amostras coletadas.

Os resultados do teste de *Friedman* (p-valor) são apresentados na Tabela 8 para os problemas-teste da fase 1 e fase 2 em conjunto. Em ambos os casos, seja considerando as médias das amostras, seja considerando o menor valor obtido, o teste estatístico evidencia diferenças significativas entre os algoritmos. Esse fato é comprovado pela análise do p-valor menor que 5% (nível de significância).

Tabela 8 – Resultados do teste de *Friedman* para problemas-teste da fase 1

Amostra	p-valor
Médias	0,000008437
Menor valor	0,0000001748

Dado este resultado, é possível analisar os resultados em pares, identificando assim quais dos algoritmos apresentaram diferença estatística significativa. Essa análise foi realizada utilizando-se o teste de Wilcoxon, ou teste dos postos sinalizados de Wilcoxon ou, ainda, teste de *Mann-Whitney-Wilcoxon* (MONTGOMERY, 2017). Esse teste é realizado com todas as amostras coletadas; portanto, não é possível fazê-lo considerando somente os melhores valores encontrados em cada algoritmo/problema-teste.

Os resultados deste teste são apresentados na Tabela 9, na qual é exibido o p-valor resultante da análise pareada dos algoritmos. Nota-se pelos resultados que somente o

algoritmo SGVNS comparado com os algoritmos VNS Adaptativo e VNS/BLVA apresentaram diferenças estatísticas significativas. Da mesma forma, e por consequência, ressalta-se também uma equivalência estatística do algoritmo *Goal Solver* com todos os algoritmos implementados neste trabalho. Diferenças estatísticas significativas ocorrem quando o p-valor for menor que 0,05 (nível de significância). Quanto mais próximo de 1 (um) for o p-valor, maior é a equivalência estatística.

Tabela 9 – Resultados do teste de *Wilcoxon*

	VNS Adapt.	VNS/BLVA	GOAL	GVNS/rVND	SGVNS
VNS/BLVA	0,569	-	-	-	-
GOAL	0,822	0,371	-	-	-
GVNS/rVND	0,695	0,785	0,891	-	-
SGVNS	0,045	0,011	0,125	0,122	-
GVNS	0,811	0,595	0,740	0,827	0,095

Capítulo 6

Conclusões e trabalhos futuros

Neste Capítulo são apresentadas as conclusões desta dissertação, além de sugestões para trabalhos futuros.

Por fim, são apresentadas as publicações geradas durante a realização deste trabalho.

6.1 Conclusões

A presente dissertação propôs uma abordagem de solução para problemas de *school timetabling*, utilizando, como base, variações do algoritmo VNS. Esta nova abordagem se mostrou bastante adequada a este tipo de problema, em especial para os problemas retratados nos problemas-teste apresentados na competição ITC2011 que não apresentavam solução inicial ótima. Os algoritmos implementados obtiveram resultados superiores ao do vencedor da competição – *Goal Solver* – em vários dos problemas-teste avaliados.

Os algoritmos implementados apresentam, como vantagem, a simplicidade e o fato de terem poucos parâmetros, independentemente do tipo de algoritmo de descida utilizado. Essa é uma vantagem sobre o algoritmo *Goal Solver*, que tem oito parâmetros, sendo quatro de cada um dos dois métodos usados na hibridização (*Simulated Annealing* e ILS). Essa vantagem fica bastante evidente quando o *Goal Solver* é comparado com os algoritmos GVNS/rVND e o VNS/BLVA, que não possuem nenhum parâmetro de configuração.

Dentre os cinco algoritmos implementados, observou-se, também, que a utilização de métodos de vizinhança randômica (GVNS/rVND) não é mais eficiente que o método GVNS e o método de vizinhança totalmente randômica (VNS/BLVA), com vantagem para o GVNS, em especial quando comparamos os resultados obtidos nos problemas-teste da Fase 1 do ITC 2011. A efetividade da randomização das vizinhanças no algoritmo de descida pode variar dependendo do problema-teste avaliado, como pode ser visto pelos resultados obtidos nos problemas-teste da Fase 1 e Fase 2.

Outra observação importante pode ser feita quanto ao algoritmo SGVNS e seu mecanismo de aceitar temporariamente soluções de piora. Os resultados foram positivos, ou seja, alcançou o melhor resultado dentre todos os algoritmos implementados em alguns problemas-teste ou ficou bem próximo do melhor valor obtido. Apesar disso, este algoritmo não se mostrou estatisticamente superior aos demais algoritmos implementados. Também ele não foi o algoritmo que obteve o melhor resultado no maior número de problemas-teste avaliados.

Além disso, observou-se, também, que o uso de probabilidades para escolher a melhor vizinhança para a busca local não resultou em soluções melhores do que as dos demais algoritmos avaliados; pelo contrário, gerou os piores resultados dentre os cinco algoritmos implementados.

6.2 Trabalhos futuros

Como sugestão de trabalhos futuros, sugere-se um estudo para analisar o impacto na alteração da sequência de uso dos movimentos na busca local dos algoritmos GVNS/VND e SGVNS. Essa análise poderia ser feita utilizando a ferramenta *iRace*, tratando a sequência dos movimentos como sendo um parâmetro de entrada destes algoritmos.

Ainda considerando os movimentos utilizados na busca local, uma outra sugestão de trabalho futuro é avaliar a efetividade dos movimentos empregados. Caso algum dos movimentos utilizado não esteja sendo efetivo na busca por melhores soluções, ele poderia ser elimintado do conjunto de movimentos disponível. O tempo economizado na execução deste movimento eliminado seria usado com outro tipo de movimento, que seja mais efetivo e com isso obter soluções ainda melhores.

Outra alternativa para trabalho futuro interessante seria testar o comportamento dos algoritmos implementados sem o limite do número de iterações na busca local ou, ainda, variar o valor definido para esse limite.

Sugere-se também estender os testes para outros problemas-teste, em especial, os utilizados em trabalhos mais recentes, como por exemplo, os de Saviniec e Constantino (2017).

6.3 Publicações originadas

São listadas nesta Seção as publicações produzidas durante o desenvolvimento desta pesquisa.

- **Título:** Um algoritmo GVNS usando o framework KHE para resolver problemas de programação de horários em escolas

  Co-autores: Marcone Jamilson Freitas Souza e Sérgio Ricardo de Souza

  Evento: XIV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC) - 6th Brazilian Conference on Intelligent Systems (BRACIS)

  Local: Uberlândia, Brasil

  Período: 2 a 5 de outubro de 2017

  Link: <http://www2.sbc.org.br/ce-ia/pg/historico/base/ENIAC-2017/Anais-ENIAC-2017.pdf>

  Páginas: 877-888

- **Título:** Uma abordagem de busca local utilizando GVNS para solução de problemas de programação de horários em escolas

  Co-autores: Marcone Jamilson Freitas Souza e Sérgio Ricardo de Souza

  Evento: XXXVIII Ibero Latin American Congress on Computacional Methods in Engineering (CILAMCE)

  Local: Florianópolis, Brasil

  Período: 5 a 8 de novembro de 2017

  Digital Object Identifier: DOI: 10.20906/CPS/CILAMCE2017-1240

  Link: <https://ssl4799.websiteseguro.com/swge5/PROCEEDINGS/PDF/CILAMCE2017-1240.pdf>

- **Título do capítulo:** An Adaptive VNS and Skewed GVNS Approaches for School Timetabling Problems

  Co-autores: Marcone Jamilson Freitas Souza, Sérgio Ricardo de Souza e Vitor Nazário Coelho

  Título do livro: Variable Neighborhood Search

  Editores: Sifaleras A., Salhi S. e Brimberg J.

  Subtítulo do livro: 6th International Conference, ICVNS 2018, Sithonia, Greece, October 4–7, 2018, Revised Selected Papers, Lecture Notes in Computer Science, volume 11328.

  Editora: Springer

Referências

APPLEBY, J.; BLAKE, D.; NEWMAN, E. Techniques for producing school timetables on a computer and their application to other scheduling problems. **The Computer Journal**, Oxford University Press, v. 3, n. 4, p. 237–245, 1961. Citado na página 8.

BARDADYM, V. A. Computer-aided school and university timetabling: The new wave. In: SPRINGER. **International Conference on the Practice and Theory of Automated Timetabling**. [S.l.], 1995. p. 22–45. Citado na página 1.

BIRATTARI, M.; BALAPRAKASH, P.; DORIGO, M. The aco/f-race algorithm for combinatorial optimization under uncertainty. In: **Metaheuristics**. [S.l.]: Springer, 2007. p. 189–203. Citado na página 35.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, ACM, v. 35, n. 3, p. 268–308, 2003. Citado na página 10.

COOPER, T. B.; KINGSTON, J. H. The complexity of timetable construction problems. In: SPRINGER. **International Conference on the Practice and Theory of Automated Timetabling**. [S.l.], 1995. p. 281–295. Citado na página 8.

CSIMA, J.; GOTLIEB, C. Tests on a computer method for constructing school timetables. **Communications of the ACM**, ACM, v. 7, n. 3, p. 160–163, 1964. Citado na página 8.

DÍAZ, A.; GLOVER, F.; GHAZIRI, H. M.; GONZÁLEZ, J.; LAGUNA, M.; MOSCATO, P.; TSENG, F. T. **Optimización Heurística y Redes Neuronales**. [S.l.]: Paraninfo, 2000. Citado na página 10.

EVEN, S.; ITAI, A.; SHAMIR, A. On the complexity of time table and multi-commodity flow problems. In: IEEE. **16th Annual Symposium on Foundations of Computer Science**. [S.l.], 1975. p. 184–193. Citado 2 vezes nas páginas 5 e 9.

FONSECA, G. H.; SANTOS, H. G. Variable neighborhood search based algorithms for high school timetabling. **Computers & Operations Research**, Elsevier, v. 52, p. 203–208, 2014. Citado na página 17.

FONSECA, G. H. G. da; SANTOS, H. G.; TOFFOLO, T. A. M.; BRITO, S. S.; SOUZA, M. J. F. GOAL solver: a hybrid local search based solver for high school timetabling. **Annals of Operations Research**, v. 239, n. 1, p. 77–97, 2016. Citado na página 9.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers & operations research**, Elsevier, v. 13, n. 5, p. 533–549, 1986. Citado na página 28.

GOTLIEB, C. The construction of class-teacher timetables. **In Proceedings of IFIP Congress**, p. 73–77, 1963. Citado 2 vezes nas páginas 1 e 5.

HAAN, P. D.; LANDMAN, R.; POST, G.; RUIZENAAR, H. A case study for timetabling in a dutch secondary school. In: SPRINGER. **International Conference on the Practice and Theory of Automated Timetabling**. [S.l.], 2006. p. 267–279. Citado na página 15.

HANSEN, P.; MLADENOVIC, N.; PÉREZ, J. A. M. Variable neighborhood search: methods and applications. **4OR: Quarterly journal of the Belgian, French and Italian operations research societies**, v. 6, p. 319–360, 2008. Citado na página 27.

HERTZ, A. Tabu search for large scale timetabling problems. **European Journal of Operational Research**, Elsevier, v. 54, n. 1, p. 39–47, 1991. Citado na página 9.

JOHNSON, D. S.; ARAGON, C. R.; MCGEOCH, L. A.; SCHEVON, C. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. **Operations Research**, INFORMS, v. 39, n. 3, p. 378–406, 1991. Citado na página 17.

KINGSTON, J. H. Hierarchical timetable construction. In: SPRINGER. **International Conference on the Practice and Theory of Automated Timetabling**. [S.l.], 2006. p. 294–307. Citado na página 16.

KINGSTON, J. H. **A software library for school timetabling**. 2012. <http://sydney.edu.au/engineering/it/~jeff/khe/>. Citado na página 9.

LÓPEZ-IBÁNEZ, M.; DUBOIS-LACOSTE, J.; STÜTZLE, T.; BIRATTARI, M. The irace package: Iterated racing for automatic algorithm configuration. **IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2011-004**, 2011. Citado na página 35.

MLADENOVIĆ, N.; DRAŽIĆ, M.; KOVAČEVIC-VUJČIĆ, V.; ČANGALOVIĆ, M. General variable neighborhood search for the continuous optimization. **European Journal of Operational Research**, Elsevier, v. 191, n. 3, p. 753–770, 2008. Citado 4 vezes nas páginas 12, 13, 23 e 27.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, Elsevier, v. 24, n. 11, p. 1097–1100, 1997. Citado 3 vezes nas páginas 12, 25 e 27.

MONTGOMERY, D. C. **Design and analysis of experiments**. [S.l.]: John wiley & sons, 2017. Citado 2 vezes nas páginas 42 e 47.

NURMI, K.; KYNGAS, J. A framework for school timetabling problem. In: **Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications, Paris**. [S.l.: s.n.], 2007. p. 386–393. Citado na página 14.

PILLAY, N. A survey of school timetabling research. **Annals of Operations Research**, Springer, v. 218, n. 1, p. 261–293, 2014. Citado na página 10.

POST, G.; KINGSTON, J. H.; AHMADI, S.; DASKALAKI, S.; GOGOS, C.; KYNGAS, J.; NURMI, C.; MUSLIU, N.; PILLAY, N.; SANTOS, H. et al. Xhstt: an xml archive for high school timetabling problems in different countries. **Annals of Operations Research**, Springer, v. 218, n. 1, p. 295–301, 2014. Citado na página 14.

SANTOS, H. G.; OCHI, L. S.; SOUZA, M. J. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. **Journal of Experimental Algorithmics (JEA)**, ACM, v. 10, p. 2–9, 2005. Citado 2 vezes nas páginas 6 e 9.

SAVINIEC, L.; CONSTANTINO, A. A. Effective local search algorithms for high school timetabling problems. **Applied Soft Computing**, Elsevier, 2017. Citado na página 50.

SCHAERF, A. A survey of automated timetabling. **Artificial intelligence review**, Springer, v. 13, n. 2, p. 87–127, 1999. Citado 2 vezes nas páginas 4 e 5.

SOUZA, M. J. **Programação de horários em escolas: uma aproximação por metaheurísticas**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2000. Citado 4 vezes nas páginas 4, 5, 9 e 15.

SOUZA, M. J.; COELHO, I. M.; RIBAS, S.; SANTOS, H. G.; MERSCHMANN, L. H. d. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. **European Journal of Operational Research**, Elsevier, v. 207, n. 2, p. 1041–1051, 2010. Citado na página 25.

TEAM, R. C. et al. R: A language and environment for statistical computing. Vienna, Austria, 2014. Citado na página 35.

THOMPSON, J. M.; DOWSLAND, K. A. A robust simulated annealing based examination timetabling system. **Computers & Operations Research**, Elsevier, v. 25, n. 7-8, p. 637–648, 1998. Citado na página 9.

TRIPATHY, A. School timetabling—a case in large binary integer linear programming. **Management Science**, INFORMS, v. 30, n. 12, p. 1473–1489, 1984. Citado na página 8.

VALOUXIS, C.; HOUSOS, E. Constraint programming approach for school timetabling. **Computers & Operations Research**, Elsevier, v. 30, n. 10, p. 1555–1572, 2003. Citado na página 14.

WERRA, D. de. An introduction to timetabling. **European Journal of Operational Research**, Elsevier, v. 19, n. 2, p. 151–162, 1985. Citado na página 8.

WERRA, D. de. Restricted coloring models for timetabling. **Discrete Mathematics**, Elsevier, v. 165, p. 161–170, 1997. Citado na página 8.

WOOD, D. A technique for colouring a graph applicable to large scale timetabling problems. **The Computer Journal**, Oxford University Press, v. 12, n. 4, p. 317–319, 1969. Citado na página 8.

WRIGHT, M. School timetabling using heuristic search. **Journal of the Operational Research Society**, Springer, v. 47, n. 3, p. 347–357, 1996. Citado na página 14.

ZHANG, D.; LIU, Y.; M'HALLAH, R.; LEUNG, S. C. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. **European Journal of Operational Research**, Elsevier, v. 203, n. 3, p. 550–558, 2010. Citado na página 9.