Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Programa de Pós-Graduação em Engenharia de Produção

Problema de Programação de Horários de Cursos Universitários da ITC2019: Modelos e Algoritmos

Paulo Sérvulo dos Santos

João Monlevade – MG, Janeiro de 2022

Paulo Sérvulo dos Santos

Problema de Programação de Horários de Cursos Universitários da ITC2019: Modelos e Algoritmos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Ouro Preto (Linha de Pesquisa: Modelagem de Sistemas Produtivos e Logísticos), como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia de Produção.

Universidade Federal de Ouro Preto

Instituto de Ciências Exatas e Aplicadas

Programa de Pós-Graduação em Engenharia de Produção

Orientador: George Henrique Godim da Fonseca

Coorientador: Paganini Barcellos de Oliveira

João Monlevade – MG

Janeiro de 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S237p Santos, Paulo Sérvulo.
Problema de programação de horários de cursos universitários da ITC2019 [manuscrito]: Modelos e algoritmos. / Paulo Sérvulo Santos. - 2022.
75 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. George Henrique Godim Fonseca.
Coorientador: Prof. Dr. Paganini Barcellos de Oliveira.
Dissertação (Mestrado Acadêmico). Universidade Federal de Ouro Preto. Departamento de Engenharia de Produção. Programa de Pós-Graduação em Engenharia de Produção.

1. Programação de horários. 2. Programação linear inteira mista. 3. Heurística. I. de Oliveira, Paganini Barcellos. II. Fonseca, George Henrique Godim. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 658.51

Bibliotecário(a) Responsável: Angela Maria Raimundo - SIAPE: 1.644.803

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Paulo Sérvulo dos Santos

Problema de Programação de Horários de Cursos Universitários da ITC2019: Modelos e Algoritmos

Dissertação apresentada ao Programa de Pós Graduação em Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Mestre em Engenharia de Produção

Aprovada em 12 de Janeiro de 2022

Membros da banca

Dr. George Henrique Godim da Fonseca - Orientador(a) (Universidade Federal de Ouro Preto)
Dr. Paganini Barcellos de Oliveira - Coorientador - (Universidade Federal de Ouro Preto)
Dr. Haroldo Gambini Santos - (Amazon)
Dr. Túlio Ângelo Machado Toffolo - (Universidade Federal de Ouro Preto)

George Henrique Godim da Fonseca, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 14/02/2022

Documento assinado eletronicamente por **George Henrique Godim da Fonseca**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 14/02/2022, às 18:51, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0279850** e o código CRC **C37C6017**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: (31)3808-0819 - www.ufop.br

A Deus e a todas as pessoas que de alguma forma me auxiliaram nessa caminhada.

Agradecimentos

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase da minha vida. Portanto, desde já, peço desculpas àquelas que não estão presentes entre essas palavras, mas, elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço primeiramente a Deus por ter me concedido o dom da vida e a oportunidade de realizar mais este grande sonho.

Reverencio o Professor Dr. George Henrique Godim da Fonseca e o Professor Dr. Paganini Barcellos de Oliveira pela dedicação e pela orientação deste trabalho e, por meio deles, me reporto a toda comunidade da Universidade Federal de Ouro Preto.

Agradeço à Universidade Federal de Ouro Preto (UFOP), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) por terem financiado esta pesquisa.

Gostaria de deixar registrado também o meu agradecimento aos meus pais, Jesus e Rosária, meus irmãos José Cláudio, Jussara e Luís, e ao meu grande amigo Glaucus, por terem me apoiado e incentivado durante toda essa caminhada.

Por fim, e não menos importante, agradeço imensamente a minha noiva Iaponyra por todo amor, carinho e compreensão.

"Combati o bom combate,
completei a carreira,
guardei a fé."
– 2Tm 4:7

Resumo

Este trabalho aborda o Problema de Agendamento de Horários de Cursos Universitários apresentado na Competição Internacional de Horários 2019 (ITC2019). O problema é composto por um conjunto de cursos, salas e estudantes, onde cada curso possui uma estrutura hierárquica que define em quais turmas o aluno pode se matricular. O objetivo é alocar uma sala e um horário para cada turma e alocar os alunos às turmas de forma a não violar as restrições de distribuição rígidas e minimizar os custos associados aos tempos, salas, penalidades das restrições fracas e conflitos de alunos. Para solucionar a problemática, uma heurística matemática multi-vizinhança do tipo Fixa-e-Otimiza, que utiliza um modelo já disponível na literatura, foi proposta. Além da heurística matemática, foram propostas diferentes técnicas de pré-processamento para a redução da dimensão das instâncias, o que contribui para compactação dos modelos. Também foi desenvolvida uma heurística construtiva capaz de gerar soluções válidas que são usadas como entrada para o algoritmo Fixa-e-Otimiza. Os resultados computacionais indicam que, para algumas das instâncias, as estratégias de pré-processamento auxiliam na geração de um modelo mais compacto. Obteve-se uma redução média de 22,03% e 7,65% na quantidade de variáveis e restrições, respectivamente, quando comparados com trabalhos da literatura. O algoritmo Fixa-e-Otimiza também se mostrou eficiente na medida em que obteve alguns resultados melhores que o segundo e terceiro colocados da ITC2019. Mesmo com o grande esforço no pré-processamento para reduzir a dimensão das instâncias, algumas delas não puderam ser carregadas em memória para serem resolvidas pelo modelo matemático.

Palavras-chaves: Problema de Agendamento de Horários de Cursos Universitários. Competição Internacional de Horários 2019. Programação Linear Inteira Mista. Heurísticas.

Abstract

This work addresses the Problem University Course Timetabling presented in the International Schedule Competition 2019 (ITC2019). The problem is composed of a set of courses, rooms and students, where each course has a hierarchical structure that define how the student should participate in classes. The objective is to allocate a room and time to each class and to allocate students to classes so as not to violate strict allocation restrictions and to minimize the costs associated with times, classrooms, penalties of soft constraints and student conflicts. To solve the problem, a Fix-and-Optimize multi-neighborhood mathematical heuristic that uses a model already available in the literature was proposed. In addition to the heuristic, different pre-processing techniques were proposed to reduce the instance size, which contributes to the compaction of the models. A constructive heuristic capable of generating valid solutions that are used as input to Fix-and-Optimize algorithm was also developed. The computational results indicate that, for some of the instances, the pre-processing strategies help in the generation of a more compact model, and on average it was possible to obtain a reduction of 22.03% and 7.65% in the number of variables and constraints, respectively, compared with the literature. Fix-and-Optimize algorithm also proved to be efficient as it obtained some better results than the second and third placed in the ITC2019. Even with the great effort in the pre-processing to reduce the dimension of the instances, some of them could not be loaded in memory to be solved by the mathematical model.

Keywords: University Course Timetabling Problem. International Timetabling Competition 2019. Mixed Integer Programing. Heuristics.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CB-CTT	*Curriculum-Based Course Timetabling*
FGO	*Forest Growth Optimization*
IFS	*Iterative Forward Search Algorithm*
ILS	*Iterated Local Search*
ITC	*International Timetabling Competition*
MaxSAT	*Maximum Satisfiability*
PE-CTT	*Post-Enrolment Course Timetabling*
PLIM	Programação Linear Inteira Mista
PM	Programação Matemática
PSO	*Particle Swarm Optimization*
SA	*Simulated Annealling*
TS	*Tabu Search*

Sumário

1 Introdução

A geração de quadros de horários em instituições de ensino é um problema de otimização amplamente abordado na literatura (QU et al., 2009; TAN et al., 2021; CHEN et al., 2021). Este tipo de problema combinatório é classificado como NP-Completo e consiste em determinar a existência de uma grade horária que atenda a um conjunto de pré-requisitos (ZUTERS, 2006). Embora a tecnologia tenha evoluído nos últimos anos, muitas das instituições de ensino, tanto do Brasil como do exterior, ainda realizam a montagem da grade horária de forma manual (GÓES; COSTA; STEINER, 2010). Tal tarefa demanda tempo e mão de obra, sendo, em muitos casos, impossível construir um horário que atenda a todos os requisitos em um horizonte de tempo adequado.

Levando em consideração a complexidade e a importância prática e teórica do problema, a literatura apresenta diversas pesquisas voltadas para o desenvolvimento de técnicas computacionais, como heurísticas, meta-heurísticas e modelos de Programação Matemática (PM) que auxiliam na geração de um quadro de horário de forma eficiente (FERLAND; ROY, 1985; ABRAMSON et al., 1999; ZHANG et al., 2010; BRITO et al., 2012; FONSECA; SANTOS, 2014; KRISTIANSEN; SØRENSEN; STIDSEN, 2015; FENG; LEE; MOON, 2017). Ainda que os esforços dos pesquisadores no desenvolvimento dessas técnicas sejam grandes, a criação de um sistema de agendamento de horários generalizado é uma tarefa complexa, uma vez que cada instituição de ensino possui uma estrutura particular. Por este motivo, os sistemas são geralmente desenvolvidos para atender a uma instituição específica (SOUZA, 2000).

Existem inúmeras variantes do problema de agendamento de horários em instituições de ensino, que se diferem principalmente pelas características das restrições envolvidas. Segundo Schaerf (1999), o problema pode ser classificado em três tipos:

- *Problema de Programação de Horários em Escolas*: exitem um conjunto de horários e um conjunto de professores que devem ser atribuídos a cada turma, levando em conta que um professor não deve estar em duas ou mais turmas no mesmo horário;
- *Problema de Programação de Horários de Cursos em Universidades*: existe um conjunto de cursos aos quais os estudantes solicitam matrícula. As aulas de cada curso devem ser alocadas a uma sala e a um horário, de forma que não haja duas aulas ocorrendo na mesma sala em horários conflitantes;

- *Problema de Programação de Horários de Exames*: existem um conjunto de estudantes, um conjunto de exames que cada estudante deve fazer e um conjunto de horários possíveis para os exames. Cada exame deve ser atribuído a um horário de forma que nenhum estudante tenha que fazer dois ou mais exames simultaneamente.

Levando em consideração a importância da problemática, algumas competições foram organizadas ao longo dos anos para incentivar o estudo sobre o agendamento de horários. A Competição Internacional de Horários, do inglês *International Timetabling Competition* (ITC), tem como objetivo motivar pesquisas sobre problemas complexos de horários decorrentes da prática. Nos últimos 20 anos ocorreram 5 edições da competição: ITC2002 (PAECHTER; GAMBARDELLA; ROSSI-DORIA, 2002); ITC2007 (GASPERO; MCCOLLUM; SCHAERF, 2007; LEWIS; PAECHTER; MCCOLLUM, 2007; MCCOLLUM et al., 2007); ITC2011 (POST et al., 2016); ITC2019 (MÜLLER; RUDOVÁ; MÜLLEROVÁ, 2018) e ITC2021 (BULCK et al., 2021). Neste contexto, este trabalho aborda a problemática da competição de 2019, focado no Problema de Agendamento de Horários de Cursos Universitários.

O Problema de Agendamento de Cursos Universitários possui inúmeras variantes que surgem devido às características da instituição de estudo. De forma geral, pode-se classificar este problema em duas classes: Problema de Agendamento de Horários de Cursos Universitários Baseados no Currículo, *Curriculum-Based Course Timetabling* (CB-CTT), e Problema de Agendamento de Horários de Cursos Universitários Pós-Matrícula, *Post-Enrolment Course Timetabling* (PE-CTT). O CB-CTT consiste em alocar uma sala e um período para vários cursos, onde os conflitos entre os cursos são definidos de acordo com o currículo publicado pela instituição de ensino (CARVALHO, 2012). Este currículo pode ser visto como um conjunto de disciplinas que constituem a carga horária completa de um conjunto de alunos (CESCHIA; GASPERO; SCHAERF, 2012). Já o PE-CTT parte do pressuposto de que os alunos já se inscreveram nos cursos que pretendem frequentar e os horários devem ser construídos de forma a não haver conflito para nenhum aluno (FONSECA et al., 2016a). Tanto no CB-CTT quanto no PE-CTT, outros objetivos secundários podem ser considerados, como, por exemplo, a disponibilidade e preferência dos professores em cada semestre letivo.

O problema da ITC2019 consiste em um Problema de Agendamento de Horários de Cursos Universitários Pós-Matrícula, onde o objetivo é alocar uma sala e um horário para uma turma de forma a minimizar os custos associados às salas, tempos, restrições de distribuição e penalidade de conflitos de estudantes. Neste problema, embora os conflitos de estudantes sejam aceitos, os mesmos são penalizados na função objetivo do problema de otimização.

1.1 Objetivos

O objetivo geral deste trabalho é o desenvolvimento de técnicas de pré-processamento e heurísticas matemáticas para o Problema de Agendamento de Horários da ITC2019. Como objetivos específicos, tem-se:

- Reproduzir o modelo apresentado por Holm et al. (2020b);
- Propor estratégias para redução do tamanho das instâncias;
- Criar técnicas de geração de solução inicial;
- Desenvolver uma heurística matemática do tipo Fixa-e-Otimiza;
- Desenvolver estruturas de vizinhança para a heurística;
- Implementar os modelos e algoritmos em ambiente computacional via *Gurobi* e o *Python-MIP* utilizando a linguagem *Python*;
- Comparar os resultados disponíveis na literatura com os resultados encontrados ao longo do trabalho.

1.2 Contribuições

Além de contribuir para o desenvolvimento das pesquisas nas áreas de pesquisa operacional e otimização, este trabalho apresenta algoritmos que podem ser adaptados para serem utilizados em diversas universidades para a geração de quadro de horários. Dentre as contribuições mais relevantes, podem-se citar:

- A proposição de um conjunto de estratégias de pré-processamento para redução das dimensões das instâncias do problema de programação de horários de cursos universitários;
- O desenvolvimento de uma heurística construtiva para a geração da solução inicial dos problemas do ITC2019;
- A implementação de uma heurística matemática de Fixa-e-Otimiza multi-vizinhança para o problema estudado.

1.3 Organização do Trabalho

O presente trabalho está organizado em sete capítulos. Neste primeiro, uma sucinta introdução ao tema é realizada para contextualização, enumeração das contribuições e definição dos objetivos pretendidos. O Capítulo 2 apresenta uma revisão de literatura sobre o agendamento de horários em instituições de ensino, com foco no agendamento em universidades. O Capítulo 3 apresenta a descrição e as principais características do problema da ITC2019. O Capítulo 4 mostra o modelo matemático proposto por Holm et al. (2020b) e reproduzido neste trabalho. O Capítulo 5 apresenta as técnicas usadas no pré-processamento, desenvolvimento e execução da heurística matemática proposta. O Capítulo 6 mostra os resultados encontrados ao longo do trabalho juntamente com as discussões acerca dos mesmos. Por fim, o Capítulo 7 apresenta as considerações finais e propostas de trabalhos futuros.

2 Revisão da Literatura

Os primeiros trabalhos relacionados ao problema de programação de horários foram apresentados por Gotlieb (1962) e Csima e Gotlieb (1964). Posteriormente, diversas pesquisas sobre o assunto foram realizadas, utilizando-se de diversas técnicas. Os primeiros esforços na área basearam-se em heurísticas construtivas, que simulavam o modo como os humanos construíam soluções para o problema. Tais métodos podem ser vistos nos trabalhos de Csima e Gotlieb (1964), Papoulias (1980) e Junginger (1986).

2.1 Problemas de Programação de Horários Escolares

Os métodos heurísticos se mostraram uma boa estratégia para solucionar esta classe de problemas. Costa (1994) apresenta um algoritmo baseado em *Tabu Search* (TS) para resolver problemas reais de cursos, obtendo resultados satisfatórios. Abramson et al. (1999) utilizam quatro variações da meta-heurística *Simulated Annealling* (SA) para a construção de quadro de horários escolares. Os testes foram realizados em instâncias geradas pelos próprios autores, onde os resultados apontaram que o esquema que usa a temperatura de transição de fase se saiu melhor. Fonseca et al. (2016b) apresentam uma metaheurística híbrida baseada na SA e no *Iterated Local Search* (ILS) utilizando diversas estruturas de vizinhança. O algoritmo proposto foi o primeiro colocado no ITC2011. O trabalho de Islam et al. (2016) é baseado no *University Timetable Generator* e utiliza o algoritmo TS para gerar um cronograma de cursos e de exames para uma universidade. No artigo, são apresentados o funcionamento do algoritmo e como obter uma solução viável. A pesquisa de Bellio et al. (2016) é voltada para o desenvolvimento de um método de busca baseado na metaheurística SA. Um dos principais pontos abordados foi o desenvolvimento de uma técnica de regressão linear capaz de retornar bons parâmetros para instâncias que ainda não foram vistas. Apesar da simplicidade, o algoritmo proposto foi capaz de retornar bons resultados.

Métodos evolutivos também são utilizados na geração de quadro de horários em instituições de ensino. Burke e Newall (1999) propõem uma técnica de decomposição para resolução de problemas menores via algoritmos evolutivos. Além de reduzir o tempo, o algoritmo é capaz de encontrar soluções de qualidade. Abdullah e Turabieh (2012) apresentam uma hibridização entre o Algoritmo Memético e a heurística TS para resolver os Problemas de Agendamento de Horário de Exames e de Cursos Universitários da ITC2007. A principal contribuição deste trabalho está relacionada com a forma e a ordem em que as estruturas de vizinhança foram utilizadas. A técnica abordada foi capaz de produzir alguns dos melhores resultados da literatura. Fong, Asmuni e McCollum (2015) apresentam o estudo de um Algoritmo de Colônia de Abelhas Artificiais combinado com a Otimização por Enxame de Partículas, do inglês *Particle swarm optimization* (PSO), e o Grande Dilúvio, onde estes dois últimos foram utilizados para melhorar as capacidades de exploração global e local das soluções. Para os experimentos foram utilizados dois conjuntos de dados, um voltado para o agendamento de exames e outro para o agendamento de cursos, ambos em universidades. Os resultados mostram que a abordagem proposta é capaz de produzir soluções de boa qualidade. Um Algoritmo Genético Híbrido foi utilizado por Feng, Lee e Moon (2017) para realizar o agendamento de horários universitários. Antes de ser solucionado, o problema foi convertido no problema de empacotamento de contêineres tridimensional. Tal solução obteve resultados melhores que o método TS, tendo como referência um tempo limite.

Em se tratando de técnicas exatas para a resolução das variantes dos problemas de horários escolares, os modelos de Programação Linear Inteira Mista (PLIM) são comumente utilizados na solução do problema. Tripathy (1984) apresenta uma formulação para o Problema de Programação de Horários Escolares em um grande Problema de Programação Linear Binária, onde é apresentado um método de solução baseado na relaxação lagrangeana acoplada à otimização de subgradientes. O método foi capaz de resolver um problema envolvendo 900 disciplinas em um programa de pós-graduação. O trabalho de Daskalaki e Birbas (2005) apresenta um modelo de programação inteira para um problema de agendamento de horários de cursos universitários. O modelo é composto por restrições fortes e fracas. Testes foram realizados em três problemas de diferentes tamanhos, além de um estudo de caso efetuado no Departamento de Engenharia Elétrica e Computação da Universidade de Patras, na Grécia. Kristiansen, Sørensen e Stidsen (2015) apresentam um modelo de PLIM capaz de lidar com problemas no formato XHSTT. A abordagem apresentada se mostrou eficaz, encontrando a solução ótima de duas instâncias que até então eram desconhecidas. Além do Algoritmo Genético Híbrido, Feng, Lee e Moon (2017) apresentam também um modelo de PM para o Problema de Agendamento de Horários em Universidades.

A Tabela 1 apresenta um conjunto de trabalhos que foram mapeados na literatura, separados pela categoria do problema (programação de horários em escolas, programação de horários de cursos universitários e programação de horários de exames) e pelo método utilizado na solução.

2.2 Problema de Programação de Horários de Cursos Universitários

Voltados especificamente para o Problema de Agendamento de Cursos em Universidades, os trabalhos de Lü e Hao (2010) e Bellio et al. (2016) são destinados a resolver problemas do tipo CB-CTT. Já Boland et al. (2008) e Ceschia, Gaspero e Schaerf (2012) voltam seus trabalhos para solução de problemas do tipo CE-CTT. Tem-se ainda problemas que envolvem professores, como pode ser visto em Hertz (1992) e Gunawan, Ng e Poh (2012). A Tabela 2 apresenta alguns trabalhos que tratam do Agendamento de Horários de Cursos Universitários destacando suas particularidades.

Tabela 1 – Trabalhos relacionados à alocação de horários em instituições de ensino.

	Categoria		
Método	**Escolas**	**Cursos**	**Exames**
PLIM	(TRIPATHY, 1984) (KRISTIANSEN; SØRENSEN; STIDSEN, 2015)	(FERLAND; ROY, 1985) (BOLAND et al., 2008) (PHILLIPS et al., 2017)	(AL-YAKOOB; SHERALI; AL-JAZZAF, 2010) (ARBAOUI; BOUFFLET; MOUKRIM, 2015) (CATALDO et al., 2017)
SA	(ABRAMSON, 1991) (ABRAMSON et al., 1999) (ZHANG et al., 2010) (FONSECA et al., 2016b)	(CESCHIA; GASPERO; SCHAERF, 2012) (GUNAWAN; NG; POH, 2012) (BELLIO et al., 2016)	(BATTISTUTTA; SCHAERF; URLI, 2017) (LEITE; MELÍCIO; ROSA, 2019)
TS	(SCHAERF, 1996) (SOUZA; MACULAN; OCHI, 2003) (HOOSHMAND; BEHSHAMEH; HAMIDI, 2013)	(HERTZ, 1992) (LÜ; HAO, 2010) (ISLAM et al., 2016)	(KHADER; SEE, 2005) (AMARAL; PAIS, 2016)
VNS	(BRITO et al., 2012) (FONSECA; SANTOS, 2014) (SAVINIEC; CONSTANTINO, 2017)	(NGUYEN et al., 2011) (BORCHANI; ELLOUMI; MASMOUDI, 2017) (MUKLASON; IRIANTI; MAROM, 2019)	(BURKE et al., 2010) (ELLOUMI et al., 2014)
ILS	(SAVINIEC; CONSTANTINO, 2017) (ANDRADE; STEINER; GÓES, 2019)	(GASPERO; SCHAERF, 2002) (BADONI; GUPTA; MISHRA, 2014) (SORIA-ALCARAZ et al., 2016)	(HADDADI; CHERAITIA, 2018)
Evolutivo	(COLORNI; DORIGO; MANIEZZO, 1998) (MOSCATO; SCHAERF, 1998) (TASSOPOULOS; BELIGIANNIS, 2012) (RAGHAVJEE; PILLAY, 2015)	(BURKE; NEWALL, 1999) (JAT; YANG, 2011) (EL-SHERBINY; ZEINELDIN; EL-DHSHAN, 2015) (FONG; ASMUNI; MCCOLLUM, 2015) (FENG; LEE; MOON, 2017)	(MOREIRA, 2008) (FONG; ASMUNI; MCCOLLUM, 2015) (LEI et al., 2015) (LEITE et al., 2018)

Tabela 2 – Trabalhos abordando o Agendamento de Horários de Cursos Universitários e suas características.

Trabalho	**CB-CTT**	**PE-CTT**	**Alunos**	**Professores**	**Capacidade das Salas**
(FERLAND; ROY, 1985)		✓	✓	✓	*Soft*
(HERTZ, 1992)	✓			✓	-
(BOLAND et al., 2008)	✓		✓	✓	*Hard*
(CESCHIA; GASPERO; SCHAERF, 2012)		✓	✓		*Hard*
(DEMIR; TUNALI; ELIIYI, 2012)	✓		✓		*Soft*
(BADONI; GUPTA; MISHRA, 2014)		✓	✓		*Hard*
(BELLIO et al., 2016)	✓		✓		*Soft*
(ISLAM et al., 2016)	✓		✓	✓	*Hard*
(MUKLASON; IRIANTI; MAROM, 2019)		✓	✓		*Hard*

2.3 Competição Internacional de Horários 2019

A ITC2019 contou com a participação de 20 competidores, sendo que cinco foram finalistas. A seguir, são apresentados os melhores colocados e os métodos utilizados:

1. Holm et al. (2020a) utilizaram uma heurística matemática do tipo Fixa-e-Otimiza. Além da heurística, destacam-se os pré-processamentos realizados pelos competidores, que reduziram consideravelmente o tamanho das instâncias e o modelo matemático utilizado. Tais estratégias garantiram um bom desempenho e o primeiro lugar da competição;

2. Rappos et al. (2020) alcançaram seus resultados por meio de um modelo de PLIM utilizando técnicas tradicionais de ramificação e corte. De início, uma técnica para geração de solução inicial foi utilizada e, em seguida, o modelo foi responsável por realizar uma busca local na solução;

3. Gashi e Sylejmani (2020) utilizaram a metaheurística SA com penalização. O algoritmo explora regiões viáveis e inviáveis do espaço de soluções, onde o último caso é tratado utilizando uma combinação de penalização e pesquisa restrita em restrições rígidas específicas;

4. Er-rhaimini (2020), inicialmente, gerou uma solução inicial válida que, em alguns casos, só foi possível gerar após um pré-tratamento dos dados. Em seguida, uma heurística foi utilizada para realizar a busca local. De modo geral, a heurística proposta escolhe uma classe ou um aluno e troca os seus recursos por um recurso melhor. Por fim, uma meta-heurística do tipo *Forest Growth Optimization* (FGO) é usada para melhorar ainda mais a solução;

5. Lemos, Monteiro e Lynce (2021) propuseram uma estratégia *Maximum Satisfiability* (MaxSAT) para solucionar o problema da competição. Além de uma etapa de pré-processamento, os autores dividiram a problemática em duas etapas: a alocação de turmas e a alocação de alunos. Neste contexto, a *MaxSAT* foi utilizado juntamente com técnicas de busca local para encontrar as soluções.

3 Descrição do Problema

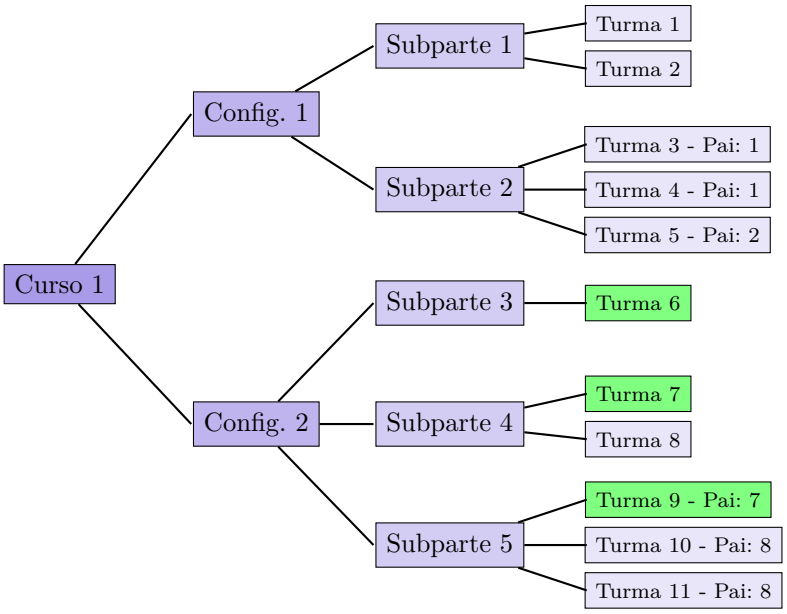
O Problema de Agendamento de Horários Universitários abordado pela ITC2019[1] é composto por um conjunto de salas, cursos e estudantes. Cada estudante solicita matrícula em um ou mais cursos, onde cada um desses cursos possui um conjunto de turmas que são organizadas em uma estrutura hierárquica predefinida. Essa estrutura define como cada aluno irá cursar cada turma dos cursos solicitados. As turmas, por sua vez, necessitam de uma sala e um horário para serem alocadas. O objetivo é alocar cada uma das turmas a uma sala e um horário, e alocar os alunos às turmas de acordo com os cursos solicitados por cada um, minimizando os conflitos de alunos e o não atendimento das restrições de distribuição fracas.

Cada curso possui uma estrutura hierárquica que deve ser respeitada por cada aluno que requisita matrícula. Um curso é composto por um conjunto de configurações, onde cada configuração possui um conjunto de subpartes, que por sua vez possui um conjunto de turmas. Cada aluno deve ser matriculado em uma turma de cada subparte de uma única configuração. Cada turma pode ter ainda uma turma pai. Caso um aluno esteja matriculado em uma turma que possui uma turma pai, o mesmo deve ser matriculado também na turma pai, ou seja, para cursar a turma filho, é um pré-requisito o aluno estar matriculado na turma pai.

Supondo um curso denominado Curso 1, que possui a estrutura apresentada na Figura 1, e um aluno que solicita matrícula nesse curso. Como o Curso 1 possui duas configurações, qualquer uma pode ser escolhida, por exemplo, a Configuração 2. Nesta configuração, o aluno deve participar de uma única turma em cada subparte. Pode-se escolher então a Turma 6 da Subparte 3, a Turma 8 da Subparte 4 e a Turma 9 da Subparte 5. Porém, é necessário respeitar a relação pai-filho. Assim sendo, como a Turma 9 possui como pai a Turma 7, obrigatoriamente, o aluno deve cursar a Turma 7 da Subparte 4, ao invés da Turma 8.

[1] Página da ITC: <https://www.itc2019.org/home>.

Figura 1 – Exemplo de estrutura de cursos do problema.



Cada uma dessas turmas deve ser alocadas a uma sala e um horário. Uma turma possui um conjunto de salas e um conjunto de horários em que pode ser alocada. Cada uma das salas possui um identificador, a quantidade máxima de alunos e um peso. Os horários também possuem um peso e são especificados da seguinte maneira: cada horário possui um vetor de binários onde cada posição representa uma semana. Caso haja aula em uma dada semana, a posição do vetor que representa essa posição recebe o valor 1, e 0 caso contrário. Por exemplo: se um semestre possui 13 semanas, tem-se o vetor [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1] que informa que há aula semana sim, semana não. Além disso, um horário também possui um vetor de dias da semana, que indica os dias em que haverá aula na semana. Por exemplo, o vetor [1, 0, 1, 0, 1, 0, 0] informa que há aula na segunda, quarta e sexta-feira.

Além dos vetores supracitados, cada horário possui ainda um *slot* de início e uma duração dada pela quantidade de *slots* de tempo de duração, sendo que, cada *slot* representa um intervalo de tempo. Por exemplo: se um dia é dividido em 288 *slots*, cada *slot* corresponde a 5 minutos, então, se uma aula se inicia no *slot* 90 e tem 10 *slots* de duração, os horários efetivos de início e término da aula são 7:30 e 8:20, respectivamente.

O problema possui algumas restrições típicas, como: duas turmas não podem se reunir em uma mesma sala em horários que se sobrepõem; as turmas que estão em uma relação pai-filho não podem se sobrepor no tempo; os conflitos de alunos ocorrem quando um aluno é matriculado em duas turmas que acontecem em horários conflitantes, mas este tipo de conflito é permitido e penalizado.

Além dessas, há um conjunto de outras restrições, conhecidas como Restrições de Distribuição, que são definidas em cada instância do problema. Cada uma dessas restrições podem ser forte ou fraca, sendo que as fracas possuem uma penalidade associada. Ademais, cada restrição possui um conjunto de turmas. Neste sentido, as seguintes restrições para o problema são enunciadas:

- *SameStart*: as aulas de todas as turmas desta restrição devem começar no mesmo horário do dia;
- *SameTime*: as aulas de todas as turmas nesta restrição devem ser ministradas no mesmo horário do dia, independentemente dos dias da semana ou semanas. Para as classes do mesmo comprimento, esta é a mesma restrição que *SameStart* (as classes devem começar no mesmo intervalo de tempo). Para as aulas de diferentes duração, a aula mais curta pode começar depois da aula mais longa, mas deve terminar antes ou ao mesmo tempo que a aula mais longa;
- *DifferentTime*: as aulas de todas as turmas nesta restrição devem ser ministradas em diferentes horários do dia, independentemente de seus dias da semana ou semanas;
- *SameDays*: as aulas de todas as turmas nesta restrição devem ser ministradas nos mesmos dias, independente do horário de início e das semanas;
- *DifferentDays*: as aulas de todas as turmas nesta restrição devem ser ministradas em diferentes dias da semana, independente de seus horários e semanas;
- *SameWeeks*: as aulas de todas as turmas nesta restrição devem ser ministradas nas mesmas semanas, independente de seus horários ou dia da semana;
- *DifferentWeeks*: as aulas de todas as turmas nesta restrição devem ser ministradas em semanas diferentes, independente do horário ou dias da semana;
- *Overlap*: as aulas de todas as turmas nesta restrição devem se sobrepor no tempo;
- *NotOverlap*: as aulas de todas as turmas nesta restrição não podem se sobrepor no tempo;
- *SameRoom*: as aulas de todas as turmas nesta restrição devem acontecer na mesma sala;
- *DifferentRoom*: as aulas de todas as turmas nesta restrição devem acontecer em salas diferentes;
- *SameAttendees*: as aulas de todas as turmas nesta restrição não podem se sobrepor no tempo, e caso sejam alocadas nos mesmos dias da semana e nas mesmas semanas, o tempo de translado entre uma sala e outra deve ser respeitado;

- *Precedence*: as aulas de todas as turmas nesta restrição devem estar uma após a outra, obedecendo a ordem predefinida;
- *WorkDay(S)*: não deve haver mais que S horários entre o início da primeira aula e o final da última aula em um determinado dia;
- *MinGap(G)*: quaisquer duas aulas ministradas no mesmo dia devem ter G espaços de tempo entre o fim da primeira e o início da segunda;
- *MaxDays(D)*: as aulas de todas as turmas nesta restrição não podem se espalhar por mais de D dias da semana, independente de estarem na mesma semana do semestre ou não;
- *MaxDayLoad(S)*: as aulas de todas as turmas nesta restrição devem ser distribuídas ao longo dos dias da semana e das semanas de forma que não haja mais do que um determinado número de horários S em todos os dias;
- *MaxBreaks(R,S)*: não deve haver mais que R quebras de horário dado que para cada dia da semana e para cada semana há uma pausa entre as aulas se houver mais de S intervalos de tempos vazios no meio;
- *MaxBlock(M,S)*: não deve haver mais que M espaços de tempo de tamanho S em um bloco, sendo que duas turmas são consideradas no mesmo bloco caso o intervalo de tempo entre elas não seja maior que S.

3.1 Descrição das Instâncias do Problema

As instâncias utilizadas na competição estão disponíveis em: <https://www.itc2019.org/instances/all>. Cada instância é um arquivo XML, como mostra a Figura 2. De início, se tem a *tag* `problem` que possui o nome e algumas características como o número de dias, número de semanas e a quantidade de *slots* por dia da instância. Dentro desta *tag*, tem-se ainda as *tags* `optimization`, `rooms`, `courses`, `distributions` e `students`. A *tag* `optimization` apresenta os pesos do objetivo de tempo, sala, restrições de distribuição e estudantes. A *tag* `rooms` contém uma série de *tags* que representam as salas do problema. Cada uma das salas possui um *ID*, uma capacidade, a distância para as outras salas, em termos do número mínimo de slots de tempo para deslocamento, além dos instantes de tempo em que está indisponível. A *tag* `courses` é responsável por armazenar todos os cursos e suas respectivas estruturas hierárquicas. A *tag* `distribution` apresenta os dados de cada uma das restrições de distribuição presentes na instância, e, além disso, contém uma série de outras *tags* que representam as restrições de distribuição, que, por sua vez, podem ser fortes ou fracas. As restrições fortes são marcadas como `required`. Já as restrições fracas possuem uma penalidade (`penalty`). Por fim, a *tag* `students` apresenta os dados dos estudantes, onde cada um é representado por uma *tag* contendo um *ID*. Cada uma das *tags* de estudantes possui um conjunto de outras *tags* responsáveis por identificar os cursos em que um aluno solicita matrícula.

Figura 2 – Especificação XML do problema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE solution PUBLIC
        "-//ITC 2019//DTD Problem Format/EN"
        "http://www.itc2019.org/competition-format.dtd">
<problem name="unique-instance-name" nrDays="7" nrWeeks="13" slotsPerDay="288">
        <optimization ... />
        <rooms> ... </rooms>
        <courses> ... </courses>
        <distributions> ... </distributions>
        <students> ... </students>
</problem>
```

Fonte: Müller, Rudová e Müllerová (2018).

Durante a competição de 2019 foram disponibilizados três grupos de instâncias: *Early*, *Middle* e *Late*. Cada um desses grupos possui um total de 10 instâncias com características distintas. A Tabela 3 apresenta algumas das principais características das instâncias dos grupos.

Tabela 3 – Características das instâncias.

	Instância	$\lvert \mathcal{K} \rvert$	$\lvert \mathcal{C} \rvert$	$\lvert \mathcal{C}_{fix} \rvert$	$\lvert \mathcal{R} \rvert$	$\lvert \mathcal{S} \rvert$	$\lvert \Delta_{\text{forte}} \rvert$	$\lvert \Delta_{\text{fraca}} \rvert$	Tam.
Early	*agh-fis-spr17*	340	1.239	543	80	1.641	820	400	14,55
	agh-ggis-spr17	272	1.852	332	44	2.116	2.202	488	5,82
	bet-fal17	353	983	79	62	3.018	861	390	3,88
	iku-fal17	1.206	2.641	530	214	0	2.237	665	12,60
	mary-spr17	544	882	63	90	3.666	3.151	796	2,94
	muni-fi-spr16	228	575	128	35	1.543	645	95	1,41
	muni-fsps-spr17	226	561	191	44	865	331	69	1,48
	muni-pdf-spr16c	1.089	2.526	1.132	70	2.938	1.456	570	15,92
	pu-llr-spr17	687	1.001	318	75	27.018	416	218	4,69
	tg-fal17	36	711	74	15	0	459	42	1,94
Middle	*agh-ggos-spr17*	406	1.144	97	84	2.254	1.181	507	11,18
	agh-h-spr17	406	460	2	39	1.988	288	111	10,74
	lums-spr18	313	487	3	73	0	449	69	3,03
	muni-fi-spr17	186	516	63	35	1.469	639	60	1,39
	muni-fsps-spr17c	116	650	32	29	395	562	147	7,70
	muni-pdf-spr16	881	1.515	159	83	3.443	579	433	6,71
	nbi-spr18	404	782	41	67	2.293	585	11	3,53
	pu-d5-spr17	212	1.061	115	84	13.497	1.262	273	2,82
	pu-proj-fal19	2.839	8.813	2.809	768	38.437	6.399	1.398	32,11
	yach-fal17	91	417	14	28	821	529	116	2,05
Late	*agh-fal17*	1.363	5.081	341	327	6.925	4.836	2.318	42,84
	bet-spr18	357	1.083	97	63	2.921	1.004	414	4,16
	iku-spr18	1.290	2.782	838	208	0	2.833	655	12,31
	lums-fal17	328	502	8	73	0	521	76	3,10
	mary-fal18	540	951	186	93	5.051	349	164	2,49
	muni-fi-fal17	188	535	60	36	1.685	635	152	1,36
	muni-fspsx-fal17	515	1.623	443	33	1.152	1.070	289	11,52
	muni-pdfx-fal17	1.635	3.717	312	86	5.651	2.433	1.068	27,25
	pu-d9-fal19	1.154	2.798	449	224	35.213	2.039	707	10,83
	tg-spr18	44	676	47	18	0	376	50	1,74

$\lvert \mathcal{K} \rvert$ número de cursos;
$\lvert \mathcal{C} \rvert$ número de turmas;
$\lvert \mathcal{C}_{\text{fix}} \rvert$ número de turmas pré-fixadas;
$\lvert \mathcal{R} \rvert$ número de salas;
$\lvert \mathcal{S} \rvert$ número de alunos;
$\lvert \Delta_{\text{forte}} \rvert$ número de restrições fortes;
$\lvert \Delta_{\text{fraca}} \rvert$ número de restrições fracas;
Tam. tamanho da instância em MB.

Fonte: Adaptada de ITC2019 (2019).

3.2 Formato do Arquivo de Saída do Problema

A solução do problema também deve ser escrita no formato XML, para que o validador, disponível em <https://www.itc2019.org/validator>, consiga verificar a solução. A Figura 3 apresenta um exemplo de uma solução no formato aceito pelo validador. A *tag* `solution` do arquivo de saída deve conter algumas informações como o nome da instância, tempo de execução e a técnica empregada na solução. A solução é composta por uma série de *tags* responsáveis por representar as turmas, onde cada uma possui as informações do dia e horário em que a aula acontece e também os identificadores dos alunos que frequentam aquela turma.

Figura 3 – Solução do problema no formato XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE solution PUBLIC
        "-//ITC 2019//DTD Problem Format/EN"
        "http://www.itc2019.org/competition-format.dtd">
<!--
        Solution: A solution contains a list of classes with their assignments.
        There are also a few solution attributes that can be used to identify the
        solution. These are:
                - problem name (only needed when the XML does not contain the problem,
                                i.e., solution is the root element)
                - solver runtime in seconds,
                - number of CPU cores that the solver employs (optional, defaults to 1),
                - name of the solver technique/algorithm,
                - name of the competitor or his/her team,
                - and the name and the country of the institution of the competitor
-->
<solution
        name="unique-instance-name"
        runtime="12.3" cores="4" technique="Local Search"
        author="Pavel Novak" institution="Masaryk University" country="Czech Republic">
        <!--
                Each class has an assigned time and (when there are rooms) an assigned
                room. Both must be from the domain of the class. There is also a list of
                students enrolled in the class.
        -->
        <class id="1" days="1010100" start="90" weeks="1111111111111" room="1">
                <student id="1"/>
                <student id="3"/>
        </class>
        <class id="2" days="0100000" start="86" weeks="0101010101011" room="4">
                <student id="2"/>
                <student id="4"/>
        </class>
        <class id="3" days="0010000" start="108" weeks="0100000000000">
                <student id="1"/>
        </class>
                <!-- ... -->
</solution>
```

Fonte: Müller, Rudová e Müllerová (2018).

4 Formulação Matemática

Este capítulo apresenta o modelo de Programação Matemática para o Problema de Agendamento de Horários de Cursos Universitários da ITC2019 proposto por Holm et al. (2020b), o qual é reproduzido neste trabalho. É válido ressaltar que esta é a única formulação matemática publicada para o problema até o presente momento.

4.1 Notações e Definições

A seguir, são apresentadas algumas notações e definições que serão importantes ao longo da formulação do problema. A Tabela 4 apresenta os conjuntos utilizados, a Tabela 5 os parâmetros e a Tabela 6 as demais notações.

Tabela 4 – Conjuntos do modelo matemático.

Símbolo	**Descrição**
Δ	Conjunto de restrições de distribuição
$\mathcal{P}$	Conjunto de variáveis de penalidades
$\mathcal{T}$	Conjunto de *slots* de tempo
$\mathcal{D}$	Conjunto de dias
$\mathcal{W}$	Conjunto de semanas
$\mathcal{K}$	Conjunto de cursos
$\mathcal{C}$	Conjunto de turmas
$\mathcal{T}$	Conjunto de tempos
$\mathcal{R}$	Conjunto de salas
$\mathcal{S}$	Conjunto de estudantes
$\mathcal{C}_\delta$	Conjunto de turmas para cada restrição de distribuição $\delta \in \Delta$
$\mathcal{R}_c$	Conjunto de salas disponíveis para a classe $c \in \mathcal{C}$
$\mathcal{T}_c$	Conjunto de tempos disponíveis para a classe $c \in \mathcal{C}$
$\mathcal{C}_s$	Conjunto de turmas que um estudante $s \in \mathcal{S}$ pode fazer parte
$\mathcal{K}_s$	Conjunto de cursos que um estudante solicita matrícula
$\mathcal{S}_c$	Conjunto de estudantes que podem fazer parte da turma $c \in \mathcal{C}$
$\mathcal{S}_k$	Conjunto de estudantes que solicitam matrícula no curso $k \in \mathcal{K}$
Ω_k	Conjunto de configurações do curso $k \in \mathcal{K}$
Z_ω	Conjunto de subpartes de uma configuração $\omega \in \Omega_k$
$\mathcal{C}_\zeta$	Conjunto de classes de uma subparte $\zeta \in Z_\omega$
$\mathcal{T}_r^u$	Conjunto de tempos em que a sala $r \in \mathcal{R}$ está indisponível

Fonte: Adaptada de Holm et al. (2020b).

Tabela 5 – Parâmetros do modelo matemático.

Símbolo	Descrição
c_δ	Custo de uma restrição de distribuição $\delta \in \Delta$
$p_{c,t}$	Penalidade por atribuir o tempo $t \in \mathcal{T}$ para a turma $c \in \mathcal{C}$
$p_{c,r}$	Penalidade por atribuir a sala $r \in \mathcal{R}$ para a turma $c \in \mathcal{C}$
ψ_t	Peso das penalidades de tempo
ψ_r	Peso das penalidades de sala
ψ_δ	Peso das penalidades de restrições de distribuição
ψ_s	Peso das penalidades de conflito de estudantes
D	Parâmetros das restrições de distribuição *MaxDays*
G	Parâmetros das restrições de distribuição *MinGap*
M	Parâmetros das restrições de distribuição *MaxBlock*
R	Parâmetros das restrições de distribuição *MaxBreaks*
S	Parâmetros das restrições de distribuição *WorkDay*, *MaxDayLoad*, *MaxBreaks* e *MaxBlock*
t^{start}	*Slot* de início do tempo $t \in \mathcal{T}$, $t^{\text{start}} \in T$
t^{length}	Duração em *slots* do tempo $t \in \mathcal{T}$
t^{end}	*Slot* de término do tempo $t \in \mathcal{T}$, $t^{\text{start}} + t^{\text{length}} = t^{\text{end}} \in T$
t^{days}	Conjunto dos dias do tempo $t \in \mathcal{T}$, $t^{\text{days}} \subseteq \mathcal{D}$
$t^{\text{days.first}}$	Primeiro dia do tempo $t \in \mathcal{T}$, $t^{\text{days.first}} \in \mathcal{D}$
t^{weeks}	Conjunto das semanas do tempo $t \in \mathcal{T}$, $t^{\text{weeks}} \subseteq \mathcal{W}$
$t^{\text{weeks.first}}$	Primeira semana do tempo $t \in \mathcal{T}$, $t^{\text{weeks.first}} \in \mathcal{W}$

Fonte: Adaptada de Holm et al. (2020b).

Tabela 6 – Demais notações do modelo matemático.

Símbolo	Descrição
c_i	Uma turma específica com ID $i \in \mathbb{Z}^+$
c_i^{parent}	Turma pai da turma c_i, $c_i^{parent} \in \mathcal{C}$
c_i^{limit}	Quantidade máxima de estudantes na turma c
$\tilde{r}$	Uma sala "fictícia" que só existe no modelo e não segue as regras de uma sala normal
r_i	Uma sala de uma turma c_i, $r_i \in \mathcal{R}_{c_i}$
t_i	Um tempo de uma turma c_i, $t_i \in \mathcal{T}_{c_i}$
$\bar{t}$	Outro tempo diferente de $t \in \mathcal{T}$, $\bar{t} \in \mathcal{T}$
τ	Um *slot* de tempo $\tau \in T$
$\bar{\tau}$	Outro *slot* de tempo diferente de $\tau \in T$, $\bar{\tau} \in T$
c_p	Custo da penalidade $p \in \mathcal{P}$
T'	Conjunto de *slots* de início, $T' \subseteq T$
T''	Conjunto de *slots* de fim, $T'' \subseteq T$
M	Big-M

Fonte: Adaptada de Holm et al. (2020b).

4.2 Variáveis de Decisão

O modelo possui duas variáveis de decisão principais, $x_{c,t,r}$ e $e_{s,c}$. A variável $x_{c,t,r}$ é responsável por indicar se uma sala foi alocada a uma turma em um determinado tempo, e pode ser formalmente definida como

$$x_{c,t,r} = \begin{cases} 1 & \text{se a turma } c \in \mathcal{C} \text{ é atribuída ao tempo } t \in \mathcal{T}_c \text{ e sala } r \in \mathcal{R}_c \\ 0 & \text{caso contrário.} \end{cases} \tag{4.1}$$

Esta variável é definida para todo $c \in \mathcal{C}$, $t \in \mathcal{T}_c$ e $r \in \mathcal{R}_c$. Caso a turma não necessite de uma sala para se reunir, o conjunto de salas é dado pela sala "fictícia", ou seja, $R_c = \{\tilde{r}\}$, sendo $\tilde{r}$ a sala "fictícia" e $\tilde{r} \notin \mathcal{R}$.

Tem-se ainda as variáveis auxiliares

$$y_{c,t} = \begin{cases} 1 & \text{se a turma } c \in \mathcal{C} \text{ é atribuída ao tempo } t \in \mathcal{T}_c \\ 0 & \text{caso contrário} \end{cases} \tag{4.2}$$

$$z_{c,d} = \begin{cases} 1 & \text{se a turma } c \in \mathcal{C} \text{ é atribuída no dia } d \in \mathcal{D} \\ 0 & \text{caso contrário} \end{cases} \tag{4.3}$$

$$w_{c,r} = \begin{cases} 1 & \text{se a turma } c \in \mathcal{C} \text{ é atribuída à sala } r \in \mathcal{R}_c \\ 0 & \text{caso contrário.} \end{cases} \tag{4.4}$$

A variável $e_{s,c}$ indica se um estudante faz parte de uma turma. Formalmente, tem-se

$$e_{s,c} = \begin{cases} 1 & \text{se o estudante } s \in \mathcal{S} \text{ participa da turma } c \in \mathcal{C}_s \\ 0 & \text{caso contrário.} \end{cases} \tag{4.5}$$

4.3 Função Objetivo

A função objetivo pode ser dividida em quatro partes: tempo, sala, restrições de distribuição e conflito de estudantes. Estas categorias tem como pesos ψ_t, ψ_r, ψ_δ e ψ_s, respectivamente. O objetivo é minimizar a soma das quatro etapas multiplicadas pelos seus respectivos pesos. A seguir, as partes da função objetivo refente a cada uma dessa etapas objetivos é apresentada.

Tempo

A parte da função objetivo referente ao tempo está relacionada a uma penalidade associada ao tempo t atribuído à cada turma, como define o termo

$$\psi_t \sum_{\substack{c \in \mathcal{C}, \\ t \in \mathcal{T}_c}} p_{c,t} y_{c,t}. \tag{4.6}$$

Sala

A etapa da função objetivo referente à sala está relacionada a uma penalidade associada à sala r atribuída a cada turma, ou seja,

$$\psi_r \sum_{\substack{c \in \mathcal{C}, \\ r \in \mathcal{R}_c}} p_{c,r} w_{c,r}. \tag{4.7}$$

Restrições de Distribuição

É importante lembrar que cada restrição de distribuição fraca define uma variável de penalidade p, que é escrita sem dimensões e um custo de penalidade c_p. Tem-se, então o conjunto de todas as variáveis de penalidade $\mathcal{P}$. Sendo assim,

$$\psi_\delta \sum_{p \in \mathcal{P}} c_p p. \tag{4.8}$$

Conflitos de Estudantes

Cada conflito de aluno tem uma penalidade unitária. Sendo assim, há uma penalização em função do número total de conflitos de alunos dada por

$$\psi_s \sum_{\substack{s \in \mathcal{S}, \\ (c_i, c_j) \in \mathcal{C}}} \chi_{s,c_i,c_j}. \tag{4.9}$$

4.4 Restrições

As restrições do modelo são divididas em três grupos: restrições primárias, restrições de distribuição e restrições de estudantes. O conjunto de restrições primárias contém as restrições gerais para a viabilidade do problema. O conjunto de restrições de distribuição definem a viabilidade do agendamento das aulas definida em cada instância. Por fim, as restrições de estudantes são responsáveis por definir quais turmas o estudante deve frequentar de acordo com a estrutura hierárquica dos cursos e por computar os conflitos de alunos.

Antes de iniciar a descrição das restrições, algumas funções que serão usadas ao longo do modelo serão definidas. A primeira é a $t.\texttt{Overlap}(\{\tau_{\min}, \ldots, \tau_{max}\})$. Esta função retorna *true* se os intervalos de tempo de t se sobrepõem aos intervalos de tempo de $\tau_{\min}$ a $\tau_{\max}$, ou seja,

$$\tau_{\min} < t^{\text{end}} \wedge t^{\text{start}} < \tau_{\max}.$$

A função $t.\texttt{Overlap}(\bar{t})$ retorna *true* se os tempos t e $\bar{t}$ tiverem pelo menos uma semana e um dia em comum e os intervalos de tempo se sobrepõem, ou seja,

$$\begin{aligned} t^{\text{weeks}} \cap \bar{t}^{\text{weeks}} \neq \emptyset \wedge \\ t^{\text{days}} \cap \bar{t}^{\text{days}} \neq \emptyset \wedge \\ \bar{t}^{\text{start}} < t^{\text{end}} \wedge \\ t^{\text{start}} < \bar{t}^{\text{end}}. \end{aligned} \tag{4.10}$$

Por último, a função $t.\texttt{Overlap}(\bar{t}, \bar{r}, r)$ retorna *true* se os horários t e $\bar{t}$ têm pelo menos uma semana e um dia em comum e os horários se sobrepõem ou, caso contrário,

não seja possível se deslocar de uma sala a outra na diferença de tempo. Sendo assim,

$$\begin{aligned} t^{\text{weeks}} \cap \bar{t}^{\text{weeks}} \neq \emptyset \wedge \\ t^{\text{days}} \cap \bar{t}^{\text{days}} \neq \emptyset \wedge \\ (\bar{t}^{\text{start}} < t^{\text{end}} \wedge t^{\text{start}} < \bar{t}^{\text{end}} \vee \\ \max\{t^{\text{start}}, \bar{t}^{\text{start}}\} - \min\{t^{\text{end}}, \bar{t}^{\text{end}}\} < \text{distance}(\bar{r}, r)). \end{aligned} \tag{4.11}$$

4.4.1 Restrições Primárias

Esta subseção apresenta a formulação das restrições principais do problema. O primeiro conjunto de restrições define que cada turma deve ter atribuída a si um tempo e uma sala. Essas restrições podem ser escritas como

$$\sum_{t \in \mathcal{T}_c} \sum_{r \in \mathcal{R}_c} x_{c,t,r} = 1 \qquad \forall c \in \mathcal{C}. \tag{4.12}$$

O próximo conjunto de restrições não permite que uma sala possua duas turmas no mesmo instante de tempo. Tais restrições podem ser escritas como

$$\sum_{\substack{c \in \mathcal{C} \\ \bar{t} \in \mathcal{T}_c: \\ t \neq \bar{t} \wedge \\ r \in \mathcal{R}_c \wedge \\ t.\texttt{Overlap}(\bar{t})}} x_{c,\bar{t},r} + M \sum_{\substack{c \in \mathcal{C}: \\ t \in \mathcal{T}_c \wedge \\ r \in \mathcal{R}_c}} x_{c,t,r} \leq M \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.13}$$

onde o Big-M é igual ao número de turmas do problema, ou seja, $M = |C|$.

As restrições 4.14 indicam os horários de indisponibilidade de uma sala, enquanto as restrições 4.15 - 4.17 são responsáveis por realizar a ligação entre a variável $x_{c,t,r}$ e as variáveis auxiliares $y_{c,t}$, $z_{c,d}$ e $w_{c,r}$.

$$\sum_{\substack{c \in \mathcal{C}: \\ r \in \mathcal{R}_c \wedge \\ t \in \mathcal{T}_c}} x_{c,t,r} = 0 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}_r^u. \tag{4.14}$$

$$\sum_{r \in \mathcal{R}_c} x_{c,t,r} = y_{c,t} \qquad \forall c \in \mathcal{C}, t \in \mathcal{T}_c. \tag{4.15}$$

$$\sum_{\substack{t \in \mathcal{T}_c: \\ d \in t^{\text{days}}, \\ r \in \mathcal{R}_c}} x_{c,t,r} = z_{c,d} \qquad \forall c \in \mathcal{C}, d \in \mathcal{D}. \tag{4.16}$$

$$\sum_{t \in \mathcal{T}_c} x_{c,t,r} = w_{c,r} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R}_c. \tag{4.17}$$

4.4.2 Restrições de Distribuição

Nesta seção serão apresentadas as formulações das restrições de distribuição. Cada tópico aborda uma única restrição (ou conjunto de restrições) do tipo em questão. Para cada restrição presente na instância, uma ou mais restrições são geradas no modelo. As variáveis auxiliares criadas ao longo da seção possuem uma dimensão δ a mais, que não é especificada. Para as restrições fracas será incluída uma variável p, cujas dimensões não serão declaradas explicitamente. O custo da variável de penalidade é denotado por c_p. Cada restrição de distribuição possui um conjunto de turmas $\mathcal{C}_\delta$. As restrições de distribuição fracas possuem uma penalidade denotada por c_δ.

SameStart

$$\text{Forte}: \quad \sum_{\substack{t_i \in \mathcal{T}_{c_i}: \\ t_i^{\text{start}} = \tau}} y_{c_i,t_i} + \sum_{\substack{t_i \in \mathcal{T}_{c_i}: \\ t_i^{\text{start}} \neq \tau}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, \tau \in \bigcup_{t \in \mathcal{T}_{c_i}} t^{\text{start}} \tag{4.18}$$

$$\text{Fraca}: \quad \sum_{\substack{t_i \in \mathcal{T}_{c_i}: \\ t_i^{\text{start}} = \tau}} y_{c_i,t_i} + \sum_{\substack{t_i \in \mathcal{T}_{c_i}: \\ t_i^{\text{start}} \neq \tau}} y_{c_i,t_i} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, \tau \in \bigcup_{t \in \mathcal{T}_{c_i}} t^{\text{start}} \tag{4.19}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

SameTime

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_i^{\text{start}} \leq t_j^{\text{start}} \wedge t_j^{\text{end}} \leq t_i^{\text{end}}) \\ \vee (t_j^{\text{start}} \leq t_i^{\text{start}} \wedge t_i^{\text{end}} \leq t_j^{\text{end}}))}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.20}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_i^{\text{start}} \leq t_j^{\text{start}} \wedge t_j^{\text{end}} \leq t_i^{\text{end}}) \\ \vee (t_j^{\text{start}} \leq t_i^{\text{start}} \wedge t_i^{\text{end}} \leq t_j^{\text{end}}))}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.21}$$

sendo $p \in \{0, 1\}$ e $c_p = c_\delta$.

DifferentTime

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_i^{\text{end}} \leq t_j^{\text{start}}) \\ \vee (t_j^{\text{end}} \leq t_i^{\text{start}}))}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.22}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_i^{\text{end}} \leq t_j^{\text{start}}) \\ \vee (t_j^{\text{end}} \leq t_i^{\text{start}}))}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.23}$$

onde $p \in \{0,1\}$ e $c_p = c_\delta$.

SameDays

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{days}} \not\subseteq t_j^{\text{days}} \wedge \\ t_j^{\text{days}} \not\subseteq t_i^{\text{days}}}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.24}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{days}} \not\subseteq t_j^{\text{days}} \wedge \\ t_j^{\text{days}} \not\subseteq t_i^{\text{days}}}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.25}$$

sendo $p \in \{0,1\}$ e $c_p = c_\delta$.

DifferentDays

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.26}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.27}$$

onde $p \in \{0,1\}$ e $c_p = c_\delta$.

SameWeeks

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \not\subseteq t_j^{\text{weeks}} \wedge \\ t_j^{\text{weeks}} \not\subseteq t_i^{\text{weeks}}}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.28}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \not\subseteq t_j^{\text{weeks}} \wedge \\ t_j^{\text{weeks}} \not\subseteq t_i^{\text{weeks}}}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.29}$$

sendo $p \in \{0,1\}$ e $c_p = c_\delta$.

DifferentWeeks

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.30}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.31}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

Overlap

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_j^{\text{start}} < t_i^{\text{end}}) \wedge \\ (t_i^{\text{start}} < t_j^{\text{end}}) \wedge \\ (t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset) \wedge \\ (t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset))}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.32}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg((t_j^{\text{start}} < t_i^{\text{end}}) \wedge \\ (t_i^{\text{start}} < t_j^{\text{end}}) \wedge \\ (t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset) \wedge \\ (t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset))}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.33}$$

sendo $p \in \{0, 1\}$ e $c_p = c_\delta$.

NotOverlap

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ (t_j^{\text{start}} < t_i^{\text{end}}) \wedge \\ (t_i^{\text{start}} < t_j^{\text{end}}) \wedge \\ (t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset) \wedge \\ (t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset)}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.34}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ (t_j^{\text{start}} < t_i^{\text{end}}) \wedge \\ (t_i^{\text{start}} < t_j^{\text{end}}) \wedge \\ (t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset) \wedge \\ (t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset)}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.35}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

SameRoom

$$\text{Forte}: \quad w_{c_i,r_i} + \sum_{r_j \in \mathcal{R}_{c_j} \setminus \{r_i\}} w_{c_j,r_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, r_i \in \mathcal{R}_{c_i} \tag{4.36}$$

$$\text{Fraca}: \quad w_{c_i,r_i} + \sum_{r_j \in \mathcal{R}_{c_j} \setminus \{r_i\}} w_{c_j,r_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, r_i \in \mathcal{R}_{c_i} \tag{4.37}$$

sendo $p \in \{0, 1\}$ e $c_p = c_\delta$.

DifferentRoom

$$\text{Forte}: \quad w_{c_i,r} + w_{c_j,r} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, r \in \mathcal{R}_{c_i} \cup \mathcal{R}_{c_j} \tag{4.38}$$

$$\text{Fraca}: \quad w_{c_i,r} + w_{c_j,r} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, r \in \mathcal{R}_{c_i} \cup \mathcal{R}_{c_j} \tag{4.39}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

SameAttendees

$$\text{Forte}: \quad x_{c_i,t_i,r_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i.\texttt{Overlap}(t_j)}} y_{c_j,t_j} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg t_i.\texttt{Overlap}(t_j), \\ r_j \in \mathcal{R}_{c_j}: \\ t_i.\texttt{Overlap}(t_j,r_j,r)}} x_{c_j,t_j,r_j} \leq 1$$
$$\forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i}, r_i \in \mathcal{R}_{c_j} \tag{4.40}$$

$$\text{Fraca}: \quad x_{c_i,t_i,r_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i.\texttt{Overlap}(t_j)}} y_{c_j,t_j} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg t_i.\texttt{Overlap}(t_j), \\ r_j \in \mathcal{R}_{c_j}: \\ t_i.\texttt{Overlap}(t_j,r_j,r)}} x_{c_j,t_j,r_j} - 1 \leq p$$
$$\forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i}, r_i \in \mathcal{R}_{c_j} \tag{4.41}$$

sendo $p \in \{0, 1\}$ e $c_p = c_\delta$.

Precedence

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_j^{\text{weeks.first}} < t_i^{\text{weeks.first}} \vee \\ (t_j^{\text{weeks.first}} = t_i^{\text{weeks.first}} \wedge \\ (t_j^{\text{days.first}} < t_i^{\text{days.first}} \vee \\ (t_j^{\text{days.first}} = t_i^{\text{days.first}} \wedge t_j^{\text{start}} <_j^{\text{start}})))}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.42}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_j^{\text{weeks.first}} < t_i^{\text{weeks.first}} \vee \\ (t_j^{\text{weeks.first}} = t_i^{\text{weeks.first}} \wedge \\ (t_j^{\text{days.first}} < t_i^{\text{days.first}} \vee \\ (t_j^{\text{days.first}} = t_i^{\text{days.first}} \wedge t_j^{\text{start}} <_j^{\text{start}})))}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.43}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

WorkDay(S)

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset \wedge \\ t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset \wedge \\ \max(t_i^{\text{end}}, t_j^{\text{end}}) - \min(t_i^{\text{start}}, t_j^{\text{start}}) > \mathbf{S}}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.44}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t_i^{\text{weeks}} \cap t_j^{\text{weeks}} \neq \emptyset \wedge \\ t_i^{\text{days}} \cap t_j^{\text{days}} \neq \emptyset \wedge \\ \max(t_i^{\text{end}}, t_j^{\text{end}}) - \min(t_i^{\text{start}}, t_j^{\text{start}}) > \mathbf{S}}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.45}$$

sendo $p \in \{0, 1\}$ e $c_p = c_\delta$.

MinGap(G)

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg(t_i^{\text{weeks}} \cap t_j^{\text{weeks}} = \emptyset \vee \\ t_i^{\text{days}} \cap t_j^{\text{days}} = \emptyset \vee \\ t_i^{\text{end}} + \mathbf{G} \leq t_j^{\text{start}} \vee \\ t_i^{\text{end}} + \mathbf{G} \leq t_j^{\text{start}})}} y_{c_j,t_j} \leq 1 \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.46}$$

$$\text{Fraca}: \quad y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg(t_i^{\text{weeks}} \cap t_j^{\text{weeks}} = \emptyset \vee \\ t_i^{\text{days}} \cap t_j^{\text{days}} = \emptyset \vee \\ t_i^{\text{end}} + \mathbf{G} \leq t_j^{\text{start}} \vee \\ t_i^{\text{end}} + \mathbf{G} \leq t_j^{\text{start}})}} y_{c_j,t_j} - 1 \leq p \qquad \forall c_i, c_j \in \mathcal{C}_\delta : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.47}$$

onde $p \in \{0, 1\}$ e $c_p = c_\delta$.

MaxDays(D)

Para esta restrição, define-se a variável auxiliar

$$\gamma_d = \begin{cases} 1 & \text{se qualquer classe } c \in \mathcal{C}_\delta \text{ é atribuída no dia } d \in \mathcal{D} \\ 0 & \text{caso contrário.} \end{cases} \tag{4.48}$$

Esta variável é definida para todo dia $d \in \mathcal{D}$ e limitada pela restrição

$$\sum_{c \in \mathcal{C}_\delta} z_{c,d} \leq M \gamma_d \qquad \forall d \in \mathcal{D} \tag{4.49}$$

sendo $M = |\mathcal{C}_\delta|$. A restrição *MaxDays(D)* pode então ser escrita como

$$\text{Forte}: \quad \sum_{d \in \mathcal{D}} \leq \gamma_d \tag{4.50}$$

$$\text{Fraca}: \quad \sum_{d \in \mathcal{D}} -\gamma_d \leq p \tag{4.51}$$

onde $p \in \mathbb{Z}^+$ e $c_p = c_\delta$.

MaxDayLoad(S)

Pode-se definir a carga diária $\phi_{w,d} \in \mathbb{Z}^+$ em um dia d em uma semana w como:

$$\phi_{w,d} = \sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c : t^{\text{week}} = w \wedge t^{\text{day}} = d}} t^{\text{length}} y_{c,t} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}. \tag{4.52}$$

As restrições *MaxDayLoad(S)* podem então ser rescritas como:

$$\text{Forte}: \quad \phi_{w,d} \leq \mathbf{S} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D} \tag{4.53}$$

$$\text{Fraca}: \quad \phi_{w,d} - \mathbf{S} \leq \iota_{w,d} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}. \tag{4.54}$$

A variável $\iota_{w,d} \in \mathbb{Z}^+$ conta a quantidade de *slots* de tempo excedentes no dia d da semana w. Sendo assim, a penalidade pode ser expressa como

$$\frac{c_\delta}{|\mathcal{W}|} \sum_{\substack{w \in \mathcal{W}, \\ d \in \mathcal{D}}} \iota_{w,d} - \epsilon \leq p. \tag{4.55}$$

A penalidade deve ser calculada usando divisão inteira. Como a divisão por $|\mathcal{W}|$ pode resultar em um valor não inteiro, deve-se subtrair $\epsilon = 9,99 \times 10^{-1}$ para vincular p corretamente. Como o custo das restrições de distribuição c_δ está incluído na restrição 4.55, $p \in \mathbb{Z}^+$ e $c_p = 1$.

MaxBreaks(R,S)

Em qualquer dia d e semana w, o número de blocos $\beta_{w,d} \in \mathbb{Z}^+$ deve ser menor que $R + 1$. Então, tem-se as restrições

$$\text{Forte}: \quad \beta_{w,d} - 1 \leq \mathbf{R} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D} \tag{4.56}$$

$$\text{Fraca}: \quad \beta_{w,d} - 1 - \mathbf{R} \leq \eta_{w,d} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D} \tag{4.57}$$

onde a variável $\eta_{w,d} \in$ conta a quantidade de *slots* de tempo excedentes no dia d e semana w. Logo, a penalidade para esta restrição é dada por

$$\frac{c_\delta}{|\mathcal{W}|} \sum_{\substack{w \in \mathcal{W}, \\ d \in \mathcal{D}}} \eta_{w,d} - \epsilon \leq p. \tag{4.58}$$

Como o custo da restrição de distribuição c_δ está incluído na restrição acima, $p \in \mathbb{Z}^+$ e $c_p = 1$.

Restrições para controlar $\beta_{w,d}$

As restrições subsequentes são usadas para realizar o controle de $\beta_{w,d}$. O número de blocos em um dia d de uma semana w é dado pela variável $\alpha_{w,d,\tau} \in \{0, 1\}$, onde $T' \subseteq T$ é o conjunto de *slots* de tempo em que qualquer $c \in \mathcal{C}_\delta$ pode começar, ou seja,

$$\alpha_{w,d,\tau} = \begin{cases} & \text{se um bloco começa na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ no intervalo de tempo } \tau \in T' \\ 0 & \text{caso contrário.} \end{cases} \tag{4.59}$$

sendo que

$$\beta_{w,d} = \sum_{\tau \in T'} \alpha_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}. \tag{4.60}$$

Para definir $\alpha_{w,d,\tau}$ corretamente, é necessária a utilização da uma variável auxiliar

$$\sigma_{w,d,\tau} = \begin{cases} & \text{se qualquer turma } c \in \mathcal{C}_\delta \text{ começa na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ no intervalo de tempo } \tau \in T' \\ 0 & \text{caso contrário} \end{cases} \tag{4.61}$$

sendo que $\sigma_{w,d,\tau}$ é controlada por

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c : t^{\text{start}} = \tau}} y_{c,t} \geq \sigma_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T' \tag{4.62}$$

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c : t^{\text{start}} = \tau}} y_{c,t} \leq M\sigma_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T' \tag{4.63}$$

onde $M = |\mathcal{C}_\delta|$.

Uma segunda variável auxiliar é dada por

$$\varepsilon_{w,d,\tau} = \begin{cases} & \text{se alguma turma } c \in \mathcal{C}_\delta \text{ está agendada para a semana} \\ & w \in \mathcal{W} \text{ no dia } d \in \mathcal{D} \text{ e se sobrepõe a qualquer } \textit{slot} \text{ de} \\ 1 & \text{tempo } \{\tau - 1 - \mathbf{S}, \ldots, \tau - 1\} \\ 0 & \text{caso contrário} \end{cases} \tag{4.64}$$

que é controlada pelas restrições

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c : \\ t.\texttt{Overlap}(\{\tau-1-\mathbf{S},\ldots,\tau-1\})}} y_{c,t} \geq \varepsilon_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T \tag{4.65}$$

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c : \\ t.\texttt{Overlap}(\{\tau-1-\mathbf{S},\ldots,\tau-1\})}} y_{c,t} \leq M\varepsilon_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T \tag{4.66}$$

sendo $M = |\mathcal{C}_\delta|$.

As variáveis auxiliares $\sigma_{w,d,\tau}$ e $\varepsilon_{w,d,\tau}$ definem $\alpha_{w,d,\tau}$ usando as restrições

$$\sigma_{w,d,\tau} - \varepsilon_{w,d,\tau} \leq \alpha_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T' \tag{4.67}$$

$$\sigma_{w,d,\tau} \geq \alpha_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T' \tag{4.68}$$

$$\varepsilon_{w,d,\tau} + \alpha_{w,d,\tau} \leq 1 \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'. \tag{4.69}$$

Note que, para um *slot* de tempo $\bar{\tau}$ onde $\varepsilon_{w,d,\bar{\tau}} = 0$, assumindo que

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c: \\ t.Overlap(\{\tau-1-\mathbf{S},\ldots,\tau-1\})}} y_{c,t} \tag{4.70}$$

é fixado em 0 (não existe $y_{c,t}$ que satisfaz as condições de soma), então

$$\sigma_{w,d,\bar{\tau}} = \alpha_{w,d,\bar{\tau}} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \bar{\tau} \in T : \left| \bigcap_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c: \\ t.\texttt{Overlap}(\{\bar{\tau}-1-\mathbf{S},\ldots,\bar{\tau}-1\})}} \{y_{c,t}\} \right| = 0. \tag{4.71}$$

A função `Overlap` informa se um tempo t se sobrepõe a um período de tempo, de forma semelhante à restrição de distribuição de sobreposição.

MaxBlock(M,S)

Inicialmente, considera-se apenas as classes que possuem um tamanho menor ou igual a **M**. As classes com tamanho maior que **M** serão abordadas posteriormente no texto. Inicialmente, a variável $\rho_{w,d,\tau} \in \{0, 1\}$, é definida como

$$\rho_{w,d,\tau} = \begin{cases} & \text{se um bloco maior que } \mathbf{M} \text{ começa na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ no } \textit{slot} \text{ de tempo } \tau \\ 0 & \text{caso contrário.} \end{cases} \tag{4.72}$$

Então, pode-se definir as restrições

$$\text{Forte}: \quad \sum_{\substack{w \in \mathcal{W}, \\ d \in \mathcal{D}, \\ \tau \in T}} \rho_{w,d,\tau} = 0 \tag{4.73}$$

$$\text{Fraca}: \quad \frac{c_\delta}{|\mathcal{W}|} \sum_{\substack{w \in \mathcal{W}, \\ d \in \mathcal{D}, \\ \tau \in T}} \rho_{w,d,\tau} \leq p. \tag{4.74}$$

Uma vez que o custo das restrições de distribuição c_δ está incluído na restrição 4.74, $p \in \mathbb{Z}^+$ e $c_p = 1$.

Restrições para controlar $\rho_{w,d,\tau}$

As restrições a seguir utilizadas são para controlar a variável $\rho_{w,d,\tau}$. Inicialmente, é preciso definir a variável

$$\gamma_{w,d,\tau} = \begin{cases} & \text{se um bloco terminar na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ no } \textit{slot} \text{ de tempo } \tau \in T'' \\ 0 & \text{caso contrário.} \end{cases} \tag{4.75}$$

Desse modo, tem-se

$$\rho_{w,d,\tau} \leq \alpha_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T \tag{4.76}$$

$$\mathbf{M}\rho_{w,d,\tau} + \mathbf{S} \sum_{\substack{\bar{\tau} \in T: \\ \tau < \bar{\tau} \leq \tau + \mathbf{M}}} \gamma_{w,d,\bar{\tau}} \leq \mathbf{M} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T \tag{4.77}$$

$$\alpha_{w,d,\tau} - \sum_{\substack{\bar{\tau} \in T: \\ \tau < \bar{\tau} \leq \bar{\tau} + \mathbf{M}}} \gamma_{w,d,\bar{\tau}} \leq \rho_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T. \tag{4.78}$$

Para controlar $\gamma_{w,d,\tau}$ é necessária a utilização de duas variáveis auxiliares, onde $T'' \subseteq T$ é o conjunto de intervalos de tempo em que qualquer $c \in \mathcal{C}_\delta$ pode terminar. Em outras palavras, tem-se

$$\varphi_{w,d,\tau} = \begin{cases} & \text{se uma turma } c \in \mathcal{C}_\delta \text{ termina na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ no } \textit{slot} \text{ de tempo } \tau \in T'' \\ 0 & \text{caso contrário} \end{cases} \tag{4.79}$$

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c: \\ t^{\text{end}} = \tau}} y_{c,t} \geq \varphi_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.80}$$

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c: \\ t^{\text{end}} = \tau}} y_{c,t} \leq M\varphi_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.81}$$

onde $M = |\mathcal{C}_\delta|$.

Na sequência, é possível definir a variável $\theta_{w,d,\tau}$ como

$$\theta_{w,d,\tau} = \begin{cases} & \text{se uma turma } c \in \mathcal{C}_\delta \text{ é atribuída na semana } w \in \mathcal{W} \\ 1 & \text{no dia } d \in \mathcal{D} \text{ e se sobrepõe à } \{\tau + 1, \ldots, \tau + 1 + \mathbf{S}\} \\ 0 & \text{caso contrário} \end{cases} \tag{4.82}$$

e as seguintes restrições

$$\sum_{\substack{c \in \mathcal{C}_\delta, \\ t \in \mathcal{T}_c: \\ t.\texttt{Overlaps}(\tau+1,\ldots,\tau+1+\mathbf{S})}} y_{c,t} \geq \theta_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.83}$$

$$\sum_{\substack{c\in\mathcal{C}_\delta,\\ t\in\mathcal{T}_c:\\ t.\texttt{Overlaps}(\tau+1,\ldots,\tau+1+\mathbf{S})}} y_{c,t} \leq M\theta_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.84}$$

sendo $M = |\mathcal{C}_\delta|$.

Agora, é possível definir as restrições em γ_w, d, τ como

$$\varphi_{w,d,\tau} - \theta_{w,d,\tau} \leq \gamma_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.85}$$

$$\varphi_{w,d,\tau} \geq \gamma_{w,d,\tau} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T'' \tag{4.86}$$

$$\theta_{w,d,\tau} + \gamma_{w,d,\tau} \leq 1 \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \tau \in T''. \tag{4.87}$$

Note que, para um *slot* de tempo $\bar{\tau}$ onde $\theta_{w,d,\bar{\tau}} = 0$, assumindo que

$$\sum_{\substack{c\in\mathcal{C}_\delta,\\ t\in\mathcal{T}_c:\\ t.\texttt{Overlap}(\{\tau-1-\mathbf{S},\ldots,\tau-1\})}} y_{c,t} \tag{4.88}$$

é fixado em 0 (não existe $y_{c,t}$ que satisfaça as condições de soma), então

$$\varphi_{w,d,\bar{\tau}} = \gamma_{w,d,\bar{\tau}} \qquad \forall w \in \mathcal{W}, d \in \mathcal{D}, \bar{\tau} \in T : \left| \bigcap_{\substack{c\in\mathcal{C}_\delta,\\ t\in\mathcal{T}_c:\\ t.\texttt{Overlap}(\{\bar{\tau}-1-\mathbf{S},\ldots,\bar{\tau}-1\})}} \{y_{c,t}\} \right| = 0. \tag{4.89}$$

A função `Overlap` informa se um tempo t se sobrepõe a um período de tempo, de forma semelhante à restrição de distribuição de sobreposição.

Caso especial

Um caso especial ocorre quando a turma tem um tamanho maior que $\mathbf{M}$. Definimos o conjunto de tais classes como $\mathcal{C}_\delta^{>\mathbf{M}}$. Para a restrição rígida, sabe-se que se uma classe $c \in \mathcal{C}_\delta^{>\mathbf{M}}$ é escalonada em um tempo específico, então qualquer outra classe de $\mathcal{C}_\delta$ não pode usar um tempo que coloque as duas no mesmo bloco. Isso implica que, se os horários compartilham pelo menos uma semana e pelo menos um dia, não pode haver menos de $\mathbf{S}$ intervalos de tempo entre eles. Por conseguinte, temos que a restrição pode ser escrita como

$$\text{Forte}: \quad y_{c_i,t_i} + \sum_{\substack{c_j\in\mathcal{C}_\delta\setminus\{c_i\},\\ t_j\in\mathcal{T}_{c_j}:\\ t_i^{\text{weeks}}\cap t_j^{\text{weeks}}\neq\emptyset\wedge\\ t_i^{\text{days}}\cap t_j^{\text{days}}\neq\emptyset\wedge\\ (t_i^{\text{start}}-t_j^{\text{end}}<\mathbf{S}\vee\\ t_j^{\text{start}}-t_i^{\text{end}}<\mathbf{S})}} y_{c_j,t_j} \leq 1 \qquad \forall c_i \in \mathcal{C}_\delta^{>\mathbf{M}}, t_i \in \mathcal{T}_{c_i}. \tag{4.90}$$

Para a restrição fraca, é necessário definir a variável $\rho_{w,d,\tau}$ se o tempo entre a turma $c_i \in \mathcal{C}_\delta^{\geq \mathbf{M}}$ e outra turma $c_j \in \mathcal{C}_\delta \backslash \{c_i\}$ for menor que $\mathbf{S}$. Isso significa que é necessária a inclusão de uma restrição para cada semana, dia e qualquer hora da turma c_i, ou seja,

$$\text{Fraca}: \quad y_{c_i,t_i} + M \sum_{\substack{c_j \in \mathcal{C}_\delta \backslash \{c_i\}, \\ t_j \in \mathcal{T}_{c_j}: \\ w \in t_j^{\text{weeks}} \wedge \\ d \in t_j^{\text{days}} \wedge \\ (t_i^{\text{start}} - t_j^{\text{end}} < \mathbf{S} \vee \\ (t_j^{\text{start}} - t_i^{\text{end}} < \mathbf{S})}} \leq \rho_{w,d,t_i^{\text{start}}} \qquad \forall c_i \in \mathcal{C}_\delta^{\geq \mathbf{M}}, t_i \in \mathcal{T}_{c_i}, w \in t_i^{\text{weeks}}, d \in t_i^{\text{days}}. \tag{4.91}$$

4.4.3 Restrições de Estudantes

Esta seção aborda as restrições de estudantes. Lembrando que a variável $e_{s,c}$ indica que um estudante s frequenta a turma c.

Primeiramente, é necessário definir a restrição que impede que uma turma tenha mais estudantes matriculados que o permitido. Então,

$$\sum_{s \in \mathcal{S}_c} e_{s,c} \leq c^{\text{limit}} \qquad \forall c \in \mathcal{C}. \tag{4.92}$$

Caso um estudante esteja matriculado em uma turma que possui uma turma pai, este deve ser matriculado também na turma pai, neste caso,

$$e_{s,c_i} \leq e_{s,c_j} \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}_s : c_i^{\text{parent}} = c_j. \tag{4.93}$$

Para cursos com apenas uma configuração os estudantes devem frequentar uma aula de cada subparte da configuração. Sendo assim,

$$\sum_{c \in \mathcal{C}_\zeta} e_{s,c} = 1 \qquad \forall k \in \mathcal{K}_s : |\Omega_k| = 1, e \in \Omega_k, \zeta \in Z_w, s \in \mathcal{S}_k. \tag{4.94}$$

Estudantes que Frequentam os Cursos

Para cursos que possuem mais de uma configuração, uma variável auxiliar $b_{s,c} \in \{0,1\}$ é utilizada. Formalmente, esta variável pode ser definida como

$$b_{s,w} = \begin{cases} & \text{se o estudante } s \in \mathcal{S} \text{ está matriculado em alguma} \\ 1 & \text{turma da configuração } w \in \Omega_k \\ 0 & \text{caso contrário.} \end{cases} \tag{4.95}$$

Os alunos devem frequentar os cursos fazendo parte de uma única configuração daquele curso, então

$$\sum_{w \in \Omega_k} b_{s,w} = 1 \qquad \forall s \in \mathcal{S}, k \in \mathcal{K}_s : |\Omega_k| > 1. \tag{4.96}$$

Os alunos devem frequentar exatamente uma turma de cada uma das subpartes da configuração em que fazem parte. Além disso, se um aluno não estiver participando de uma configuração, nenhuma aula das subpartes dessa configuração poderá ser assistida. Neste sentido, tem-se que

$$\sum_{c \in \mathcal{C}_\zeta} e_{s,c} = b_{s,w} \qquad \forall k \in \mathcal{K}_s : |\Omega_k| > 1, w \in \Omega_k, \zeta \in Z_w, s \in \mathcal{S}_k. \tag{4.97}$$

Conflitos de estudantes

Para os conflitos de estudantes a variável $\chi_{s,c_i,c_j} \in \{0,1\}$ é definida como

$$\chi_{s,c_i,c_j} = \begin{cases} & \text{se houver um conflito para aluno } s \in \mathcal{S} \text{ entre as classes} \\ 1 & c_i \in \mathcal{C}_s \text{ e } c_j \in \mathcal{C}_s \\ 0 & \text{caso contrário.} \end{cases} \tag{4.98}$$

Esta variável depende das variáveis f_{s,c_i,c_j} e o_{c_i,c_j} definidas como

$$f_{s,c_i,c_j} = \begin{cases} & \text{se o estudante } s \in \mathcal{S} \text{ está matriculado na turma } c_i \in \mathcal{C}_s \\ 1 & \text{e } c_j \in \mathcal{C}_s \\ 0 & \text{caso contrário} \end{cases} \tag{4.99}$$

$$o_{c_i,c_j} = \begin{cases} 1 & \text{se as turmas } c_i in \mathcal{C} \text{ e } c_i in \mathcal{C} \text{ se sobrepõe no tempo} \\ 0 & \text{caso contrário.} \end{cases} \tag{4.100}$$

Para ocorrer um conflito de estudantes, ambas as variáveis f_{s,c_i,c_j} e o_{ci,c_j} devem ser iguais a 1. Sendo assim,

$$o_{c_i,c_j} + f_{s,c_i,c_j} - 1 \leq \chi_{s,c_i,c_j} \qquad \forall s \in \mathcal{S}, (c_i,c_j) \in \mathcal{C}_s : i < j. \tag{4.101}$$

Controlando as variáveis auxiliares

A variável f_{s,c_i,c_j} depende da variável $e_{s,c}$. Desse modo,

$$e_{s,c_i} + e_{s,c_j} - 1 \leq f_{s,c_i,c_j} \qquad \forall s \in \mathcal{S}, (c_i,c_j) \in \mathcal{C}_s : i < j \tag{4.102}$$

$$e_{s,c_i} \geq f_{s,c_i,c_j} \qquad \forall s \in \mathcal{S}, (c_i,c_j) \in \mathcal{C}_s : i < j \tag{4.103}$$

$$e_{s,c_j} \geq f_{s,c_i,c_j} \qquad \forall s \in \mathcal{S}, (c_i,c_j) \in \mathcal{C}_s : i < j. \tag{4.104}$$

A variável o_{c_i,c_j} é dependente do tempo e da sala de ambas as aulas. Note que se trata de um cenário semelhante à restrição de distribuição *SameAttendees*, mas aqui é dividida em dois tipos de restrições, uma para os tempos que se sobrepõem e outra para os

tempos que não se sobrepõem, mas as salas atribuídas causam uma sobreposição. Portanto, tem-se

$$y_{c_i,t_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ t.Overlap(t_j)}} y_{c_j,t_j} - 1 \leq o_{c_i,c_j} \qquad \forall c_i, c_j \in \mathcal{C} : i < j, t_i \in \mathcal{T}_{c_i} \tag{4.105}$$

$$x_{c_i,t_i,r_i} + \sum_{\substack{t_j \in \mathcal{T}_{c_j}: \\ \neg t_i.Overlap(t_j), \\ r_j \in \mathcal{R}_{c_j}: \\ t_i.Overlap(t_j,r_j,r_i)}} x_{c_j,t_j,r_j} - 1 \leq o_{c_i,c_j} \qquad \forall c_i, c_j \in \mathcal{C} : i < j, t_i \in \mathcal{T}_{c_i}, r_i \in \mathcal{R}_{c_i}. \tag{4.106}$$

5 Metodologia

Os métodos utilizados neste trabalho para solucionar o Problema de Programação de Horários Universitários da ITC2019 podem ser divididos em três etapas. A primeira é o pré-processamento das instâncias para remover redundâncias e variáveis desnecessárias do problema. A segunda etapa consiste em gerar uma solução inicial válida para cada instância do problema. Por último, tem-se a etapa de execução do algoritmo Fixa-e-Otimiza proposto, utilizando o modelo apresentado no Capítulo 4. Cada uma dessas etapas serão apresentadas neste capítulo.

Ao decorrer do trabalho, outras técnicas foram exploradas para contornar o problema referente ao tamanho dos modelos, como os métodos *clique_merge* e *lazy_constrs_generator* da classe *Model* do pacote *Python–MIP*. O *clique_merge* realiza a busca por restrições com variáveis conflitantes e tenta agrupar essas restrições em restrições maiores com todos os conflitos mesclados. Já com a utilização do *lazy_constrs_generator*, o modelo é gerado de forma incompleta. As restrições faltantes são adicionadas ao modelo a medida que uma inviabilidade referente a estas restrições é detectada. Porém, devido ao custo computacional e ao tamanho das instâncias, estas técnicas não puderam ser aplicadas ao problema.

5.1 Pré-Processamento das Instâncias

Uma característica do problema abordado neste trabalho é o tamanho das instâncias, que na maioria dos casos envolvem milhares de turmas e estudantes. Este fato leva os modelos construídos a partir da formulação do Capítulo 4 conterem milhões de variáveis e restrições, demandando mais memória que o disponível na maioria dos computadores convencionais. Sendo assim, faz-se necessário realizar um pré-processamento nas instâncias para reduzir o seu tamanho e, consequentemente, reduzir o tamanho de seus respectivos modelos. O trabalho de Holm et al. (2020b) dedicou grande esforço ao pré-processamento dos modelos, porém as técnicas aplicadas não foram piblicadas até o presente momento. Esta seção propõe um conjunto de técnicas para reduzir o tamanho das instâncias sem que as mesmas percam suas características.

5.1.1 Indisponibilidade de Salas

Cada turma do problema possui um conjunto de salas e tempos em que a mesma pode ser alocada. Cada sala, por sua vez, possui um conjunto de horários em que ela não está disponível. Se uma turma possui apenas uma sala possível e algum dos tempos possíveis desta turma se sobrepõe a algum dos tempos em que a sala esteja indisponível, este tempo pode ser removido da turma. Caso a turma possua mais de uma possibilidade de sala, e um tempo possível desta turma se sobreponha a pelo menos um tempo de cada uma das salas possíveis, o tempo pode ser removido da turma. Este pré-processamento é realizado em $O(|C| \times |\overline{T_c}| \times |\overline{R_c}|)$, onde $|C|$ é a quantidade de turmas da instância, $|\overline{T_c}|$ é a quantidade média de tempos nas turmas e $|\overline{R_c}|$ é a quantidade média de salas nas turmas.

5.1.2 Restrições de Distribuição

Nas restrições de distribuição fortes, *SameStart*, *SameTime*, *DifferentTime*, *SameDays*, *DifferentDays*, *SameWeeks*, *DifferentWeeks*, *Overlap*, *NotOverlap*, *SameRoom*, *DifferentRoom*, *SameAttendees* e *Precedence*, os conflitos ocorrem por pares de turmas. Sendo assim, dado duas turmas em uma restrição dos tipos anteriormente descritos, qualquer tempo de uma das turmas deve respeitar essa restrição com pelo menos um tempo de outra classe. Caso isso não ocorra, o tempo da primeira turma pode ser removido. Por exemplo, supondo que as classes c_1 e c_2 pertencem a uma restrição *SameStart*. Se um tempo t_i qualquer da classe c_1 não atender à restrição para qualquer tempo t_j de c_2, o tempo t_i de c_1 pode ser removido. Para o caso da restrição *SameRoom*, ao invés do tempo, leva-se em consideração as salas.

A complexidade dessa etapa depende da restrição a qual o pré-processamento está sendo feito. Pra a restrição *SameAttendees*, que envolve tanto os tempos quanto as salas das turmas, o pré-processamento é realizado em $O(|\delta| \times |\overline{C_\delta}|^2 \times |\overline{T_c}|^2 \times |\overline{R_c}|^2)$, onde o δ é o conjunto das restrições de mesmo tipo, neste caso, *SameAttendees*, e $|\overline{C_\delta}|$ é a média de turmas em cada restrição do tipo em questão. Para as restrições do tipo *SameRoom* ou *DifferentRoom*, que envolvem somente as salas, o pré-processamento é realizado em $O(|\delta| \times |\overline{C_\delta}|^2 \times |\overline{R_c}|^2)$. Como as demais restrições descritas nesta seção envolvem somente o tempo das turmas, o seu pré-processamento é realizado em $O(|\delta| \times |\overline{C_\delta}|^2 \times |\overline{R_c}|^2)$.

5.1.3 Restrições Redundantes

Algumas das restrições de distribuição das instâncias, tanto fracas como fortes, são subconjuntos de outra restrição do mesmo tipo. Neste caso, a restrição que possui o subconjunto de classes de uma outra restrição do mesmo tipo pode ser excluída. Por exemplo: caso haja uma restrição *SameDays* com as turmas c_1, c_3, c_5 e c_7 e outra restrição *SameDays* com as turmas c_1, c_2, c_3, c_4, c_5, c_6 e c_7, a primeira restrição pode ser removida do problema sem a perda das características da instância. Este pré-processamento necessita apenas de comparar cada par de restrições do mesmo tipo, sendo assim, o seu ele é realizado em $O(|\delta|^2)$.

5.1.4 Pares de Turmas Redundantes

Como mencionado anteriormente, nas restrições de distribuição *SameStart*, *SameTime*, *DifferentTime*, *SameDays*, *DifferentDays*, *SameWeeks*, *DifferentWeeks*, *Overlap*, *NotOverlap*, *SameRoom*, *DifferentRoom*, *SameAttendees* e *Precedence*, os conflitos ocorrem por pares de turmas. É possível então selecionar todas as restrições de um mesmo tipo e fazer cada par de turmas nas restrições serem uma restrição individual. Por exemplo: seja uma restrição *SameWeeks* contendo as turmas c_1, c_2 e c_4 e uma outra restrição *SameWeeks* contendo as turmas c_2, c_3, c_4. É possível então transformar essas duas restrições em 6 outras restrições do tipo *SameWeeks*, onde cada uma dessas restrições tem os seguintes pares de turma: $\mathcal{C}_1 = (c_1, c_2)$, $\mathcal{C}_2 = (c_1, c_4)$, $\mathcal{C}_3 = (c_2, c_4)$, $\mathcal{C}_4 = (c_2, c_3)$, $\mathcal{C}_5 = (c_2, c_4)$ e $\mathcal{C}_6 = (c_3, c_4)$. Note que os pares $\mathcal{C}_3$ e $\mathcal{C}_5$ são repetidos. Neste caso, pode-se remover um, tendo como resultado os seguintes pares que representam as restrições *SameWeeks*: $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$ e $\mathcal{C}_6$.

Uma vez transformadas as restrições em pares de classe, é necessário verificar cada um desses pares, identificando aqueles que em momento algum irão se conflitar. Por exemplo: dado o par de turmas $\mathcal{C}_1 = (c_1, c_2)$ que deve satisfazer à restrição *SameWeeks*. Caso todos os tempos de c_1 tenham as mesmas semanas que todos os tempos de c_2, esse par de turmas pode ser removido das restrições *SameWeeks*.

É importante lembrar que este pré-processamento pode ser realizado tanto nas restrições fracas quanto nas fortes. Mas, para as restrições fracas, deve-se levar em consideração o valor da penalidade. No exemplo anterior, foram transformadas duas restrições em seis. Cada uma das novas restrições contém a sua respectiva penalidade, de acordo com a restrição que a gerou. Assim sendo, removendo uma restrição redundante, a restrição resultante deve conter a soma das penalidades daquelas que foram removidas.

Este pré-processamento precisa percorrer, em um primeiro momento, todas as restrições, transformando-as em restrições contendo duas turmas. Esta etapa é realizada em $O(|\delta| \times |\overline{C_\delta}|^2)$. A segunda etapa consiste em comparar cada par de restrições do mesmo tipo resultante da primeira etapa. Então, cada tipo de restrição em que se aplica este pré-processamento é realizado em $O(|\delta|^2)$.

5.1.5 Pré-processamento da Etapa dos Estudantes

As Equações 4.105 e 4.106 têm seu funcionamento baseado em pares de turmas. Para este caso, todos os pares de turmas presentes na instância devem ser levados em consideração. Tal fato leva o modelo a ter milhões de restrições e variáveis. O objetivo é reduzir a quantidade de pares de classes que devem ser levados em consideração para a construção dessas restrições.

As restrições dos estudantes podem ser divididas em dois tipos. Um tipo que leva em consideração se as turmas se sobrepõem no tempo e outro tipo leva em consideração os tempos de deslocamentos entre as salas. Para a restrição da Equação 4.105, para cada par de turmas (c_i, c_j), é necessário verificar se algum dos tempos de c_i se sobrepõe a algum dos tempos de c_j. Caso as turmas não tenham tempos que se sobrepõem, não é necessário criar uma restrição para tal par. O mesmo funciona para a distância de separação entre as turmas, que é dada na restrição da Equação 4.106. Caso não haja possibilidade das turmas de um par (c_i, c_j) serem alocadas em horários e salas que não dê tempo de se locomover de uma pra outra, não é necessário criar restrições para este par de classes. Este pré-processamento pode ser realizado em $O(|C|^2 \times |\overline{T_c}|^2 \times |\overline{R_c}|^2)$.

Um pré-processamento adicional é ainda realizado levando em consideração as turmas que cada um dos estudantes pode cursar. Caso um par de turmas qualquer não apareça simultaneamente para nenhum estudante, não é necessário criar restrições para este par.

5.2 Geração da Solução Inicial

Após a etapa de pré-processamento, é preciso criar uma solução inicial factível que será utilizada como entrada pelo algoritmo de Fixa-e-Otimiza. Para tal finalidade, foi desenvolvido um algoritmo heurístico que é dividido em duas etapas, as quais podem ser executadas separadamente. A primeira etapa é responsável por alocar uma sala e um horário para cada turma. A segunda é responsável por definir os alunos que irão assistir as aulas de uma turma.

Alocar uma sala e um horário a uma turma

O método utilizado para alocar uma sala e um horário para uma turma é baseado no *Iterative Forward Search Algorithm* (IFS), ou Algoritmo Iterativo de Busca Direta. Esta técnica é utilizada na geração de soluções iniciais de diversos problemas (MÜLLER, 2004). A sua principal característica é que, apesar de ser baseado em busca local, o algoritmo trabalha com soluções incompletas. Em contrapartida, a solução parcial deve ser viável a todo o momento, ou seja, todas as restrições rígidas devem ser atendidas. Dessa forma, a criação se dá de forma dinâmica, onde, a cada iteração, um objeto tem seus recursos alocados.

De forma complementar ao IFS, uma técnica de estatísticas baseadas em conflitos é utilizada para selecionar os melhores recursos a serem alocados a cada iteração. O principal objetivo desta estratégia é memorizar os conflitos que levam à inviabilidade de uma solução, de forma a evitar este conflito no futuro.

O IFS e o método de estatísticas baseadas em conflitos foram combinados para criar um método capaz de gerar soluções iniciais para o problema. O Algoritmo 1 apresenta a implementação proposta do IFS para o Problema de Agendamento de Horários da ITC2019. O algoritmo recebe como entrada um conjunto com todas as turmas do problema e como saída, o algoritmo retorna o conjunto de todas as turmas alocadas. A Linha 1 é responsável por iniciar a estrutura de conflito usada pela técnica de estatísticas baseadas em conflitos. A Linha 2 cria um conjunto de turmas pré-alocadas, sendo que essas turmas são aquelas que possuem apenas um horário e uma sala possível. A Linha 3 inicia o conjunto de turmas não alocadas com as turmas pré-alocadas. A Linha 4 cria o conjunto de turmas alocadas, onde este conjunto é iniciado com a diferença entre o conjunto contendo todas as turmas e o conjunto de turmas pré-alocadas. A Linha 5 executa as ações das Linhas 6 e 7 até que o conjunto de turmas não alocadas $\mathcal{C}_\text{U}$ esteja vazio. A Linha 6 seleciona a turma com a menor quantidade de pares de tempo/sala do conjunto de turmas não alocadas $\mathcal{C}_\text{U}$. Na Linha 7, a função `AssignResources`, apresentada no Algoritmo 2, é chamada para escolher um recurso para a turma. Finalmente, a Linha 8 retorna o conjunto de turmas alocadas.

Algoritmo 1: Aloca tempo e sala para uma turma.

Entrada: Conjunto de classes $\mathcal{C}$.
Saída: conjunto de classes alocadas $\mathcal{C}_\text{A}$.
▷ Contador de conflitos de cada par (t, r) da turma c
1 $X_{c,(t,r)} \leftarrow 0$ para cada turma $c \in \mathcal{C}$ e par de tempo-sala $(t, r) \in \mathcal{T}_c \times \mathcal{R}_c$
2 $\mathcal{C}_\text{P} \leftarrow$ `TurmasPrealocadas`$(\mathcal{C})$ ▷ Conjunto de turmas pré-alocadas
3 $\mathcal{C}_\text{A} \leftarrow \mathcal{C}_\text{P}$ ▷ Conjunto de turmas alocadas
4 $\mathcal{C}_\text{U} \leftarrow \mathcal{C} - \mathcal{C}_\text{P}$ ▷ Conjunto de turmas não alocadas
5 **enquanto** $\mathcal{C}_U \neq \emptyset$ **faça**
6 $c \leftarrow$ `SelecionaTurma`$(\mathcal{C}_\text{U})$ ▷ Seleciona a turma com menos opções de pares sala-tempo
7 $\mathcal{C}_\text{U}, \mathcal{C}_\text{A} \leftarrow$ `AlocaRecurso`$(c, \mathcal{C}_\text{U}, \mathcal{C}_\text{A}, X)$
8 **retorna** $\mathcal{C}_\text{A}$

O Algoritmo 2 é responsável por alocar uma sala e um horário para uma turma. Este algoritmo recebe como entrada a turma c a ser alocada, o conjunto de turmas não alocadas, $\mathcal{C}_{\mathrm{U}}$, as turmas que já foram alocadas, $\mathcal{C}_{\mathrm{A}}$, e a estrutura usada pela técnica de estatísticas baseadas em conflitos, X. Como saída, o algoritmo retorna o conjunto de turmas não alocadas, o conjunto de turmas alocadas e a estrutura de conflitos. A Linha 1 passa por todos os pares tempo-sala da turma c. A Linha 2 obtém as turmas que entram em conflito com a turma c se o par tempo-sala em questão for selecionado. A Linha 3 verifica se a lista de turmas conflitantes obtida na Linha 2 está vazia. Se esta lista estiver vazia, a Linha 4 atribui o par tempo-sala atual para a turma c. A Linha 5 remove a turma c do conjunto de turmas não alocadas e a Linha 6 adiciona a turma c ao conjunto de turmas alocadas. A Linha 7 retorna os conjuntos de turmas não alocadas, turmas alocadas e a estrutura X. A partir da Linha 8, a execução só ocorre se todos os pares tempo-sala da turma c causarem algum conflito com as turmas já alocadas. A Linha 8 seleciona o par tempo-sala que causou menos vezes a desalocação de c. Essa quantidade é armazenada na estrutura fornecida por X. A Linha 9 obtém as turmas que estão em conflito com a turma c para o par tempo-sala selecionado na Linha anterior. A Linha 10 atribui o par tempo-sala à turma c. A linha 12 remove a turma c do conjunto de turmas não alocadas. A Linha 12 adiciona a turma c ao conjunto de turmas já alocadas. A Linha 13 percorre todas as turmas presentes na lista de conflitos do par tempo-sala alocado à c. A Linha 14 desaloca o par tempo-sala de cada turma c no conjunto de turmas conflitantes. A Linha 15 remove cada uma dessas turmas do conjunto de turmas alocadas e a Linha 16 adiciona cada turma à lista de turmas não alocadas. A Linha 17 incrementa o contador, presente na estrutura de conflitos X, responsável por armazenar a quantidade de vezes que um par tempo-sala teve que ser desalocado a uma turma. Este contador é usado na Linha 8 para selecionar qual recurso será atribuído à turma c. Por fim, a Linha 18 retorna o conjunto de turmas não alocadas, o conjunto de turmas alocadas e a estrutura de conflitos.

De forma geral, cada par tempo-sala pertencente à turma c é analisado levando em consideração se o par causa algum conflito com as turmas já alocadas. Caso nenhum conflito aconteça com o par tempo-sala selecionado, este par é alocado à turma c, e c é removida da lista de turmas não alocadas e adicionada à lista de turmas alocadas. Caso nenhum par tempo-sala da turma c possa ser alocado, a lista de pares tempo-sala de c é analisada e o recurso que causou o menor número de desalocações no passado é alocado à turma c. A quantidade de desalocações é dada pelo número de vezes que o recurso foi alocado à turma em questão e a mesma teve de ser desalocada por causa de conflitos. Cada uma das turmas que conflitam com c quando o par tempo-sala é alocado são desalocadas e, consequentemente, cada uma dessas turmas são removidas da lista de turmas alocadas e adicionada à lista de turmas não alocadas. Ao desalocar um recurso de uma turma, este recurso tem o seu contador de desalocação incrementado. Este contador é usado para analisar qual recurso será escolhido caso todos os recursos da turma causem algum conflito.

Algoritmo 2: Atualiza os conflitos e atribuições de tempo e sala.

Entrada: Turma c; Turmas não alocadas C_{U}; Turmas alocadas C_{A}; Contador de conflitos X.
Saída: C_{U}; C_{A}; e X atualizados.

1 **para** $(t, r) \in \mathcal{T}_c \times \mathcal{R}_c$ **faça** ▷ Opções de par tempo-sala para a turma c
2 $C_{\mathrm{C}} \leftarrow$ `TurmasConflitantes`$(c, t, r, C_{\mathrm{A}})$ ▷ Conjunto de turmas conflitantes com c
3 **se** $C_C = \emptyset$ **então** ▷ Não há conflito na alocação de (t, r) para c
4 `AlocaTempoSala`(t, r, c) ▷ Aloca o par tempo-sala (t, r) para c
5 $C_{\mathrm{U}} \leftarrow C_{\mathrm{U}} - \{c\}$
6 $C_{\mathrm{A}} \leftarrow C_{\mathrm{A}} \cup \{c\}$
7 **retorna** C_{U}, C_{A}, X
8 $(t, r) \leftarrow$ `SelecionaTempoSala`(c, X_c) ▷ Seleciona o par tempo-sala com menos conflitos
9 $C_{\mathrm{C}} \leftarrow$ `TurmasConflitantes`$(c, t, r, C_{\mathrm{A}})$
10 `AlocaTempoSala`(t, r, c)
11 $C_{\mathrm{U}} \leftarrow C_{\mathrm{U}} - \{c\}$
12 $C_{\mathrm{A}} \leftarrow C_{\mathrm{A}} \cup \{c\}$
13 **para** $c \in C_C$ **faça**
14 $(t, r) \leftarrow$`DesalocaTempoSala`(c) ▷ Desaloca a turma conflitante c
15 $C_{\mathrm{A}} \leftarrow C_{\mathrm{A}} - \{c\}$
16 $C_{\mathrm{U}} \leftarrow C_{\mathrm{U}} \cup \{c\}$
17 $X_{c,(t,r)} \leftarrow X_{c,(t,r)} + 1$ ▷ Incrementa o contador de conflitos para o par (t, r) da turma c
18 **retorna** C_{U}, C_{A}, X

A cada iteração o algoritmo seleciona uma turma e realiza a sua alocação. É importante lembrar que as turmas são escolhidas de acordo com a quantidade de recursos que cada uma possui, sendo que as turmas com menos recursos são alocadas primeiro. O algoritmo se encerra quando a lista de turmas não alocadas estiver vazia.

Alocar alunos às turmas de um curso

Cada aluno possui um conjunto de cursos nos quais deve estar matriculado. O Algoritmo 3 é utilizado para atribuir os alunos às turmas de acordo com a estrutura hierárquica de cada curso. O algoritmo recebe como entrada o conjunto de alunos $\mathcal{S}$ e o conjunto de cursos que o aluno deve fazer $\mathcal{K}_s$. A saída é o conjunto de alunos $\mathcal{S}$ já atribuídos às suas turmas. A Linha 1 percorre todos os alunos do problema. A Linha 2 percorre todos os cursos que um aluno s deve cursar. A função da Linha 3 retorna todas as possibilidades válidas que um aluno possui para um curso. Cada possibilidade é um conjunto de turmas que leva em consideração a estrutura hierárquica de cada curso as restrições do problema. A Linha 4 seleciona a melhor possibilidade válida, que é aquela que causa o menor número de conflitos para o aluno. A Linha 5 percorre cada turma presente na possibilidade selecionada na linha anterior. A Linha 6 atribui o aluno a cada turma c da possibilidade selecionada. Finalmente, a Linha 10 retorna os alunos já atribuídos às suas turmas.

Algoritmo 3: Aloca os alunos às suas turmas.

Entrada: Conjunto de alunos $\mathcal{S}$; Conjunto de cursos que um aluno deve cursar $\mathcal{K}_s$.
Saída: $\mathcal{S}$ atualizado.

```
1 para s ∈ S faça
2     para k ∈ K_s faça
3         P_c ← CombinacaoTurmas(k)        ▷ Conjunto de possibilidades de turmas do curso k
4         p ← SelecionaMelhor(P_c, s)      ▷ Seleciona a possibilidade que causa menos conflitos s
5         para c ∈ p faça                  ▷ Para cada turma c da possibilidade p
6             AlocaEstudante(s, c)         ▷ Aloca o aluno s à turma c
7 retorna S
```

A Linha 3 do Algoritmo 3 é responsável por gerar um conjunto de possibilidades válidas para um curso. Cada curso possui uma estrutura hierárquica que especifica como o aluno irá participar de cada turma do curso. Assim sendo, levando em consideração esta estrutura, para cada curso de cada aluno é gerado um conjunto de possibilidades de turmas que o mesmo pode cursar. Cada uma dessas possibilidades possui um conjunto de turmas dado pela estrutura do curso em questão. A estrutura de um curso qualquer, apresentada na Figura 4, servirá como exemplo para as explicações sobre a dinâmica do algoritmo proposto.

Como o aluno deve ser matriculado em uma turma de cada subparte de uma única configuração de cada curso, tem-se as seguintes possibilidades:

i) Turma 1 e Turma 3;

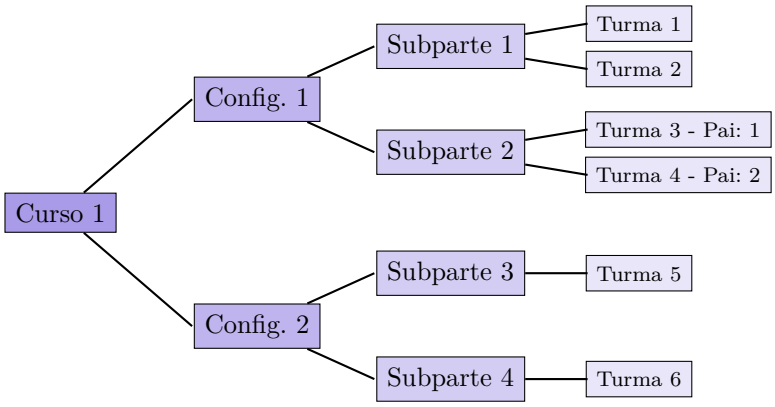
ii) Turma 1 e Turma 4;

iii) Turma 2 e Turma 3;

iv) Turma 2 e Turma 4;

v) Turma 5 e Turma 7.

Figura 4 – Exemplo de estrutura de cursos.



As quatro primeiras possibilidades são referentes à Configuração 1 e a última referente à Configuração 2. Tendo essas possibilidades, cada uma é analisada de acordo com a hierarquia pai-filho especificada no problema. Caso uma possibilidade não atenda essa hierarquia, a mesma é descartada. Por exemplo: tem-se a possibilidade *iii* no exemplo da Figura 4, mas a "Turma 2" tem a "Turma 4" como pai. Por conseguinte, a possibilidade "Turma 2 e Turma 3" não é válida, assim como a possibilidade *ii*. Em cada uma das possibilidades é analisado também se alguma das turmas presentes já está com a capacidade máxima. Caso isto ocorra, esta possibilidade também é excluída.

Das possibilidades válidas encontradas, a melhor delas é selecionada, onde o critério é a menor quantidade de conflitos causados para o aluno. Essa possibilidade possui todas as turmas que o aluno deve fazer parte. Deste modo, um aluno s deve ser matriculado em cada turma c da possibilidade p, de acordo com os cursos escolhidos pelo aluno. O algoritmo se encerra quando todos os alunos são matriculados em todos os cursos solicitados.

5.3 Algoritmo de Fixa-e-Otimiza

A heurística matemática Fixa-e-Otimiza foi proposta por Gintner, Kliewer e Suhl (2005) e, ao mesmo tempo, de forma independente, por Pochet e Wolsey (2006). Esta técnica tem como principal característica a decomposição de um PLIM em problemas menores. Em cada iteração do algoritmo, algumas variáveis do problema são selecionadas e fixadas de acordo com a solução corrente e o restante das variáveis são liberadas. Essas variáveis devem ser diferentes a cada iteração. Desta forma, a cada passo um subproblema é otimizado, mantendo o atendimento das restrições do problema original. A melhor solução encontrada ao longo de todas as iterações do algoritmo é retornada como solução para o problema.

Neste trabalho, as variáveis escolhidas para serem fixadas são as variáveis de decisão $x_{c,t,r}$ e $e_{s,c}$, onde a primeira indica que uma turma c foi alocada em um horário t e uma sala r e a segunda indica se um aluno s assiste ou não as aulas de uma turma c. Essas variáveis são selecionadas de acordo com um conjunto de turmas e alunos, onde estes são escolhidos por meio das estruturas de vizinhança que variam conforme o problema tratado. Portanto, a partir de um conjunto de turmas, é possível fixar um conjunto de variáveis x e, a partir de um conjunto de alunos, fixar um conjunto de variáveis e.

O Algoritmo 4 apresenta a abordagem proposta para o algoritmo Fixa-e-Otimiza. Este algoritmo recebe como entrada a instância do problema $\mathbb{P}$, uma solução inicial s, o limite de tempo t_{max}, o limite de tempo para resolver cada subproblema (iteração) t_{it}, o tamanho inicial do subproblema n e o número de estruturas de vizinhança k. Como saída, o algoritmo retorna a melhor solução encontrada s. Inicialmente, o modelo e sua solução inicial são carregados (Linhas 1 e 2). O algoritmo é executado até que um limite de tempo seja atingido (Linha 3). A cada iteração, uma vizinhança aleatória é selecionada (Linha 4). As classes e os alunos a serem otimizados nesta iteração são selecionados de acordo com a vizinhança v (Linha 5), e as variáveis relacionadas a esses conjuntos de classes e alunos são selecionadas (Linha 6). Posteriormente, todas as variáveis, exceto as selecionadas, são fixadas em seus valores atuais (Linha 7). Este subproblema é resolvido pelo solucionador MIP (Linha 8), e todas as variáveis são liberadas para a próxima iteração (Linha 9). Finalmente, se o status do solucionador para a iteração atual for "*Ótimo*", significa que o subproblema é fácil de ser resolvido e seu tamanho é aumentado em um fator de α (Linhas 10 e 11); caso contrário, o subproblema é difícil, e seu tamanho é reduzido em um fator de β (Linhas 12 e 13). Os valores de α e β escolhidos foram de 5% e 10%, respectivamente. Esses valores foram definidos empiricamente a partir de experimentos realizados com alguns valores de aumento entre 1% e 10 % e valores de redução entre 10% e 50%. Quando o limite de tempo é atingido, a melhor solução encontrada s é retornada (Linha 14).

Algoritmo 4: Executa a heurística Fixa-e-Otimiza proposta.

Entrada: Instância do problema $\mathbb{P}$; Solução inicial s; Tempo limite t_{max}; Tempo limite de cada iteração t_{it}; Tamanho do subproblema n; Número de estruturas de vizinhas k; α Fator de aumento do subproblema; β Fator de redução do subproblema.

Saída: Melhor solução s encontrada.

1 $\mathbb{M} \leftarrow$ Carrega o modelo MIP para $\mathbb{P}$ com s
2 $\mathcal{X} \leftarrow$ Carrega os valores das variáveis da solução s
3 **enquanto** tempo decorrido $\leq t_{max}$ **faça**
  ▷ Selecione as variáveis a serem fixadas e liberadas
4 $k_{it} \leftarrow$ `VizinhancaAleatoria`(k)
5 $(\mathcal{C}^{free}, \mathcal{S}^{free}) \leftarrow$ `SelecionaTurmasEAlunos`$(\mathbb{M}, k_{it}, n)$
6 $\mathcal{V} \leftarrow$ Variáveis relacionadas com $\mathcal{C}^{free}$ e $\mathcal{S}^{free}$
  ▷ Fixa/libera as variáveis e soluciona o modelo
7 Fixa as variáveis $\mathcal{X} \setminus \mathcal{V}$ com seu valor atual
8 $(s, status) \leftarrow$ Soluciona $\mathbb{M}$ com tempo limite t_{it}
9 Libera as variáveis fixadas em $\mathbb{M}$
  ▷ Ajuste do tamanho do subproblema
10 **se** *status* = *"Ótimo"* **então**
11 $n \leftarrow n * \alpha$
12 **senão**
13 $n \leftarrow n * \beta$
14 **retorna** s

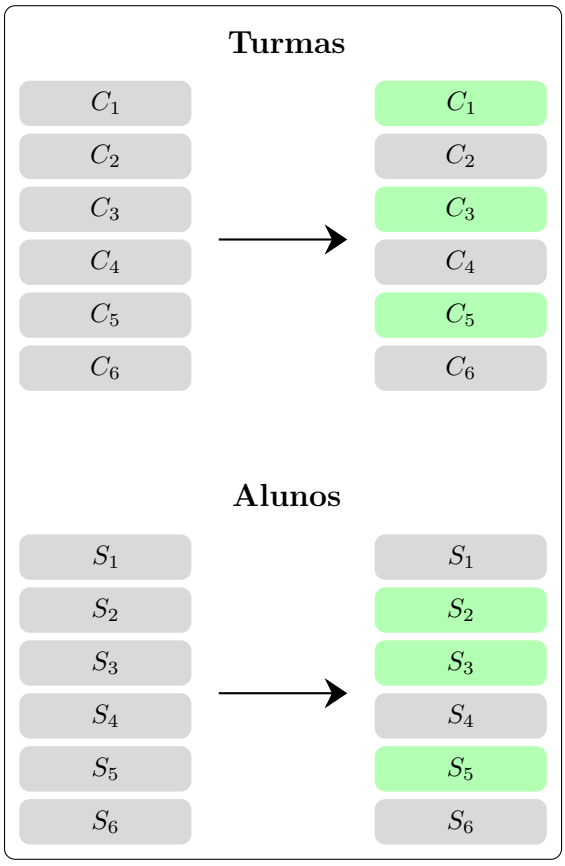
A seleção das turmas e alunos a terem suas variáveis fixadas ou liberadas deve ser realizada obedecendo às quantidades estabelecidas a cada iteração e levando em consideração a estrutura de vizinhança selecionada. Ao todo foram implementadas seis estruturas de vizinhança diferentes que serão apresentadas na sequência.

5.3.1 Seleção Aleatória das Turmas e dos Alunos

As turmas que terão suas variáveis x liberadas são selecionadas de forma aleatória. As demais variáveis relacionadas com as turmas restantes devem ser fixadas na otimização. Para as escolha das variáveis e a serem liberadas, um conjunto de alunos aleatórios é selecionado e as variáveis e relacionados com estes alunos são selecionadas para otimização. As variáveis e relacionadas com os demais alunos que não estão neste conjunto são fixadas.

A Figura 5 apresenta um exemplo desta vizinhança. No total são seis turmas (C_1, C_2, C_3, C_4, C_5, e C_6) e seis alunos (S_1, S_2, S_3, S_4, S_5 e S_6), também apresentados na Figura 5. Para este e os demais exemplos, a metade das turmas e dos alunos deve ser escolhida para ter suas variáveis liberadas, enquanto a outra metade terá suas variáveis fixadas. De forma aleatória, as turmas C_1, C_3 e C_4 e os alunos S_2, S_3 e S_4, que estão em vermelho, foram escolhidos para terem suas variáveis liberadas. As demais turmas (C_2, C_4 e C_6) e os demais alunos (S_1, S_4 e S_6) terão suas variáveis fixadas.

Figura 5 – Seleção aleatória das turmas e alunos.



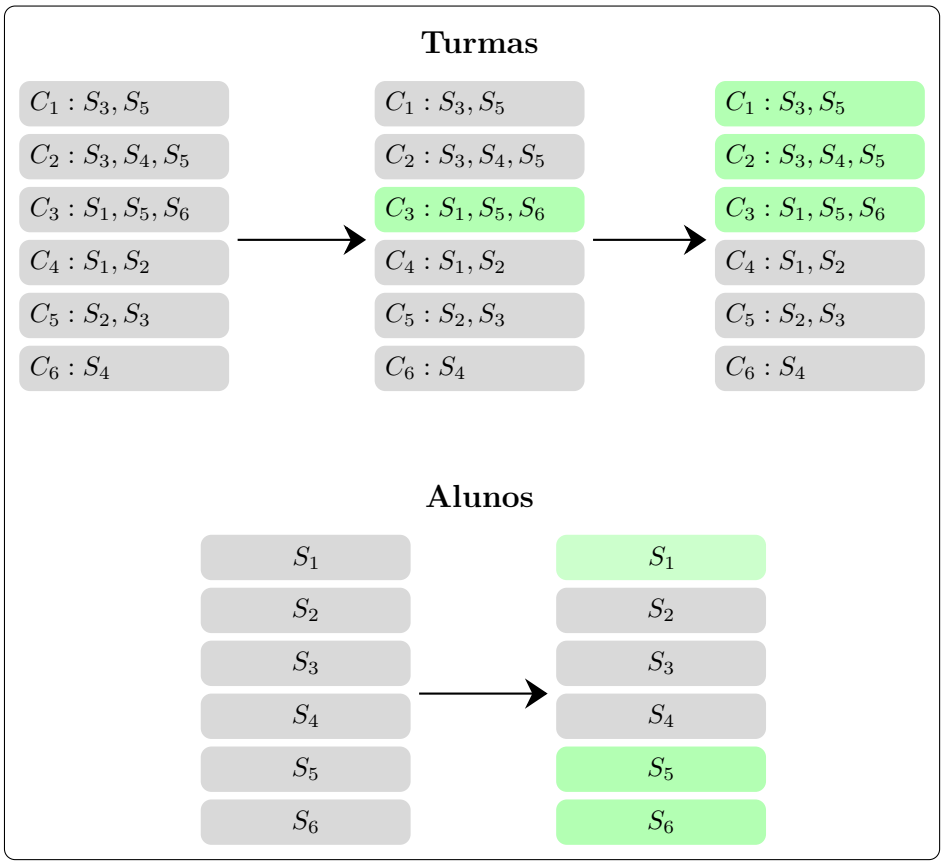
5.3.2 Seleção das Classes que Possuem Alunos em Comum

Neste caso, as classes são selecionadas de acordo com os alunos que podem cursá-las. Inicialmente, uma classe é selecionada aleatoriamente e as classes que possuem algum aluno em comum com ela também são selecionadas. Cada um dos alunos pertencentes a estas classes também são selecionados. Estes passos são repetidos até que um dado número de classes seja selecionado. As variáveis x e e relacionadas com essas classes e com esses estudantes são selecionadas para serem liberadas. As demais variáveis são fixadas.

Um exemplo desta vizinhança é apresentado na Figura 6. Primeiramente, das seis turmas, é preciso escolher uma de forma aleatória, neste caso a turma escolhida foi a C_3. Essa turma tem S_1, S_5 e S_6 como possíveis alunos, sendo assim, as turmas que possuem algum desses como possíveis alunos também devem ser selecionadas. Neste caso, as turmas C_1, C_2 e C_4 também devem ser escolhidas. Mas, como só pode ser selecionada a metade das turmas, ou seja, três, a última turma escolhida, C_4, deve ser removida, restando somante as turmas C_1, C_2 e C_3 que terão suas variáveis liberadas. Os alunos que terão suas variáveis liberadas são selecionados de acordo com as turmas escolhidas na etapa anterior, ou seja, C_1, C_2 e C_3.

Os alunos S_1, S_3, S_4, S_5 e S_6 estão em algumas dessas turmas, mas devem ser selecionados apenas três desses. A escolha será feita selecionando os primeiros alunos das primeiras turmas que foram selecionadas anteriormente. A primeira turma a ser escolhida foi a C_3, que foi selecionada aleatoriamente, seguido da turma C_1, C_2 e C_4, seguindo a ordem em que aparecem na lista de turmas da Figura 6. Como a turma C_3 possui três possíveis alunos, os escolhidos são S_1, S_5 e S_6.

Figura 6 – Seleção das turmas a partir de alunos em comum e dos alunos a partir das turmas.



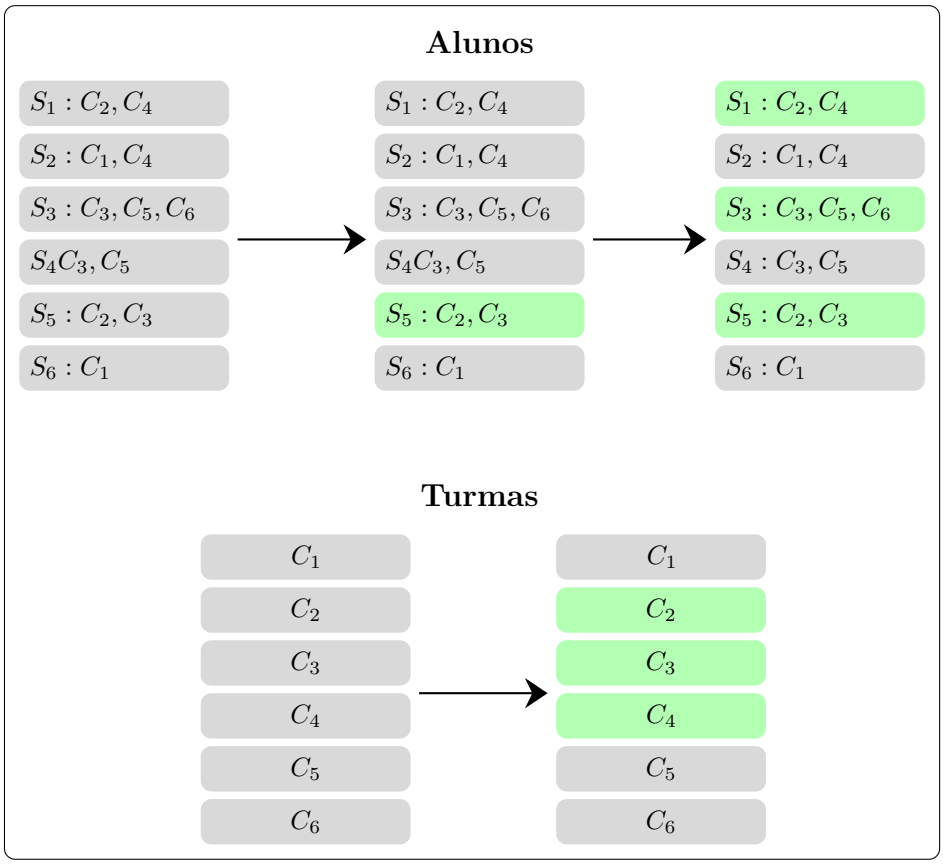
5.3.3 Seleção dos Alunos que Possuem um Curso em Comum

Dado um aluno, selecionado de forma aleatória, os alunos que devem cursar algum curso em comum com este aluno são selecionados para o conjunto de alunos que devem ter suas variáveis liberadas. As turmas pertencentes aos cursos desses alunos também são selecionadas. As variáveis x e e relacionadas com as classes e estudantes destes conjuntos serão selecionadas para serem liberadas. As demais variáveis pertencentes ao restante dos alunos são fixadas.

Esta vizinhança é o oposto da anterior, onde, em um primeiro momento, um aluno é escolhido aleatoriamente, como apresentado na Figura 7. Após a seleção deste aluno, neste caso o S_5, os demais alunos são escolhidos de acordo com as turmas em comum que cada um possui. Como o aluno S_5 possui as turmas C_2 e C_3 como turmas possíveis, os alunos S_1, S_3 e S_4 também são escolhidos. Mas como só é possível escolher três, os primeiro visitados serão escolhidos para terem suas variáveis liberadas, neste caso, S_1, S_3 e S_5, como mostra a Figura 7.

A seleção das turmas é feita levando em consideração os alunos escolhidos. O primeiro aluno selecionado foi o S_5, então, as turmas C_2 e C_3 serão selecionadas. O segundo aluno escolhido foi o S_1, então, a última turma escolhida será a C_4, que é a primeira turma que aparece em S_4. Sendo assim, as turmas escolhidas para terem suas variáveis liberadas são C_2, C_3 e C_4, como mostra a Figura 7.

Figura 7 – Seleção dos alunos a partir de turmas em comum e das turmas a partir dos alunos.

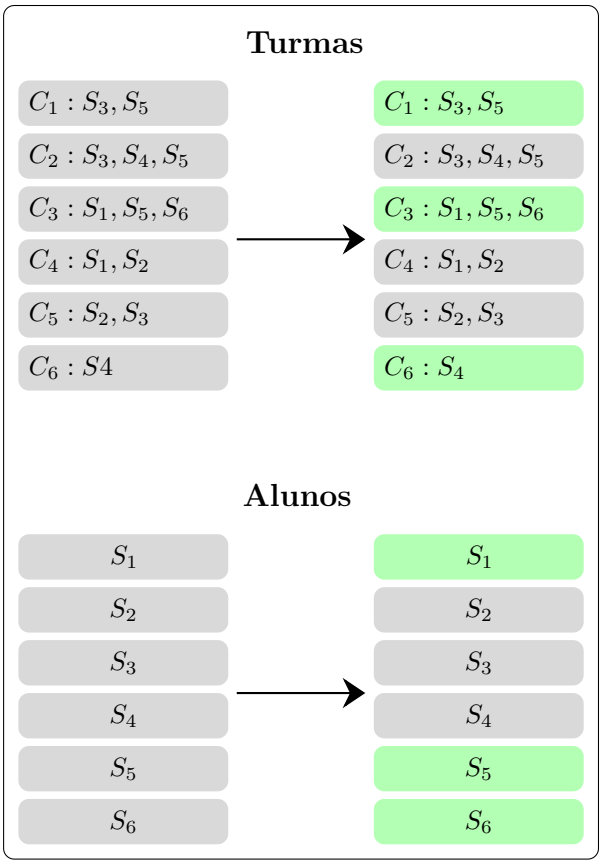


5.3.4 Seleção das Turmas a Partir de um Conjunto de Alunos Aleatórios

Dado um conjunto de alunos selecionados de forma aleatória, para cada um desses alunos, além das suas variáveis *e* serem selecionadas para serem liberadas, as turmas as quais estes alunos podem fazer parte também são selecionadas para terem as suas variáveis liberadas. As demais variáveis são fixadas.

A Figura 8 apresenta um exemplo dessa vizinhança. Inicialmente, as turmas C_1, C_3 e C_6 são selecionadas aleatoriamente. A turma C_1 foi a primeira a ser selecionada, então, os alunos S_1 e S_2, que são possíveis alunos de C_1, são escolhidos. A segunda turma escolhida foi a C_3, que possui S_1, S_5 e S_6 como alunos possíveis. Como o aluno S_1 já foi escolhido, o próximo aluno a ser selecionado é o S_5. Tem-se então as três turmas e os três alunos selecionados, como pode ser observado na Figura 8.

Figura 8 – Seleção aleatória das turmas e seleção dos alunos a partir das turmas.

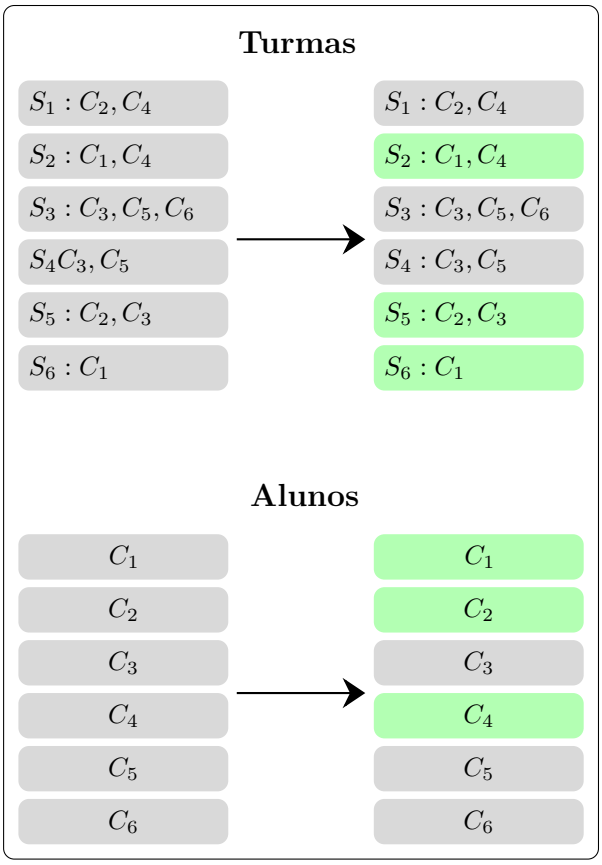


5.3.5 Seleção dos Alunos a Partir de um Conjunto de Turmas Aleatórias

Dado um conjunto de turmas selecionadas aleatoriamente, para cada uma dessas turmas, além das suas variáveis x serem selecionadas para otimização, os alunos que podem vir a fazer parte dessa turma terão as suas variáveis e selecionadas para serem liberadas. O restante das variáveis do problema são selecionadas para serem fixadas.

Um exemplo dessa vizinhança é apresentado na Figura 9. Inicialmente, são selecionados três alunos de forma aleatória, S_2, S_5 e S_6. Cada um desses alunos possui um conjunto de turmas que ele pode fazer parte. O primeiro aluno selecionado, S_2, possui as turmas C_1 e C_4, como pode ser observado na Figura 9. Sendo assim, as turmas C_1 e C_4 serão escolhidas para terem suas variáveis liberadas. O segundo aluno escolhido foi o S_5, que possui as turmas C_2 e C_3. A primeira turma que aparece em S_5 é C_2, que é escolhida para completar as três turmas necessárias para a liberação de suas variáveis. Sendo assim, os alunos S_2, S_5 e S_6 e as turmas C_1, C_2 e C_4 são os escolhidos para terem as variáveis liberadas. As demais turmas e alunos terão suas variáveis fixadas.

Figura 9 – Seleção aleatória dos alunos e seleção das turmas a partir dos alunos.

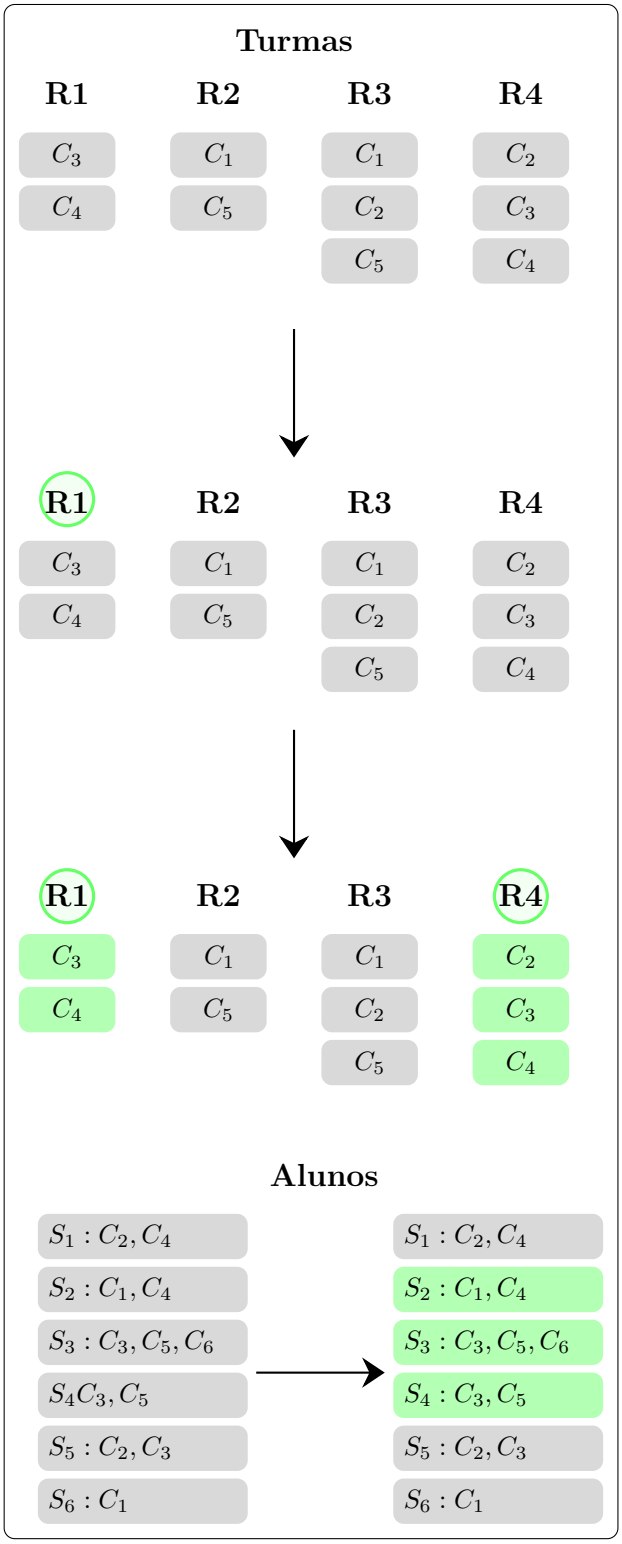


5.3.6 Seleção das Turmas de Acordo com as Restrições de Distribuição

Nesta vizinhança, as variáveis x serão selecionadas de acordo com as restrições de distribuição, dadas por cada instância. A princípio, uma turma é selecionada de forma aleatória. Em seguida, cada uma das restrições de distribuição são analisadas e as turmas que podem vir a conflitar com a turma selecionada são escolhidas para ter suas variáveis liberadas. Esses passos são repetidos até que o número de turmas desejadas seja satisfeito. Os alunos são selecionados de acordo com as turmas escolhidas anteriormente. Portanto, os alunos que podem vir a fazer parte dessa turma terão as suas variáveis e selecionadas para serem liberadas, respeitando a quantidade máxima de alunos que podem ser escolhidos. As variáveis x e e, que estão presentes nesses conjuntos, são selecionadas para serem liberadas.

O exemplo desta última vizinhança é apresentado na Figura 10. Cada restrição possui um conjunto de turmas. Neste exemplo, tem-se um total de quatro restrições (R_1, R_2, R_3 e R_4). Uma restrição é então escolhida de forma aleatória, por exemplo a restrição R_1. O próximo passo é selecionar as turmas dessa restrição, C_3 e C_4, e verificar quais restrições possuem turmas em comum. Neste caso, a única restrição que possui as turmas C_3 ou C_4 é a restrição R_4. Esta restrição possui as turmas C_2, C_3 e C_4, mas as turmas C_3 e C_4 já foram selecionadas, restando somente a turma C_2 para ser escolhida. Logo, as turmas que terão suas variáveis liberadas são C_2, C_3 e C_4, como mostra a Figura 10.

Figura 10 – Seleção das turmas a partir das restrições de distribuição e dos alunos a partir das turmas.



Os alunos são selecionados a partir das turmas que cada um pode cursar e as turmas que foram selecionadas na etapa anterior. Deste modo, os alunos que possuem como possibilidades as turmas C_2, C_3 ou C_4 serão selecionados, respeitando a quantidade máxima de alunos, que para o exemplo dado é três. Os primeiros alunos que possuem alguma dessas turmas, S_1, S_2 e S_3, são selecionados para terem as variáveis liberadas, como mostra a Figura 10.

6 Resultados

Este capítulo apresenta os resultados encontrados em cada uma das etapas citadas no Capítulo 5. De início, foram realizados testes com o pré-processamento, onde cada instância foi submetida e avaliada. Posteriormente, foi gerado o modelo para cada uma das instâncias já pré-processadas. Em seguida, cada instância foi utilizada para gerar uma solução inicial pelo método heurístico. Por fim, o algoritmo de Fixa-e-Otimiza foi executado e os resultados foram comparados com os disponíveis na literatura.

6.1 Configurações dos Experimentos

A abordagem proposta foi implementada em *Python 3.9* utilizando o interpretador *pypy3*, que se mostrou mais eficiente que o interpretador nativo. O modelo foi desenvolvido com o auxílio da biblioteca Python-MIP[1], que é utilizada para modelagem e solução de problemas de PLIM. O solucionador comercial Gurobi[2], na versão 9.1 com *MIPFocus* definido para viabilidade, foi empregado para resolver os modelos de PLIM. Os experimentos computacionais foram realizados em um *desktop* Ubuntu 18.04 equipado com um processador i7-3770 rodando a 3,40 GHz com 32 GB de RAM.

6.2 Instâncias Utilizadas nos Testes

Para realização dos experimentos, foram utilizadas as instâncias disponibilizadas por Holm et al. (2020b). Estas instâncias já possuem algum pré-processamento comparadas com as instâncias disponíveis no site da ITC2019. As instâncias de Holm et al. (2020b) foram pré-processadas utilizando as técnicas apresentadas na Seção 5.1.

[1] Página Python-MIP: <https://www.python-mip.com/>.
[2] Página do Gurobi: <https://www.gurobi.com/>.

A Tabela 7 apresenta o *status* de execução das 30 instâncias em cada etapa do problema. Para seis das 30 instâncias, a etapa de pré-processamento necessitou de mais de 24 horas para ser executada[3]. Para três das 30 instâncias, o algoritmo construtivo não pôde fornecer uma solução viável em 24 horas. Para 11 das 30 instâncias, o modelo matemático não pôde ser construído em menos de 24 horas ou a quantidade de memória RAM disponível não foi suficiente. Finalmente, para duas instâncias, o algoritmo de Fixa-e-Otimiza não pôde ser executado, pois excede o limite de memória disponível. As instâncias que se enquadram em algum desses casos não foram consideradas em alguns dos experimentos devido à impossibilidade de execução, seja pelo tempo, pela memória ou por ambos. Dadas as etapas anteriores, foi possível executar o algoritmo de Fixa-e-Otimiza para 15 das 30 instâncias da ITC2019.

6.3 Experimentos Computacionais

Esta seção apresenta os resultados obtidos para as etapas de pré-processamento, geração do modelo, geração da solução inicial e execução do algoritmo de Fixa-e-Otimiza descritos nos capítulos anteriores.

6.3.1 Pré-processamento e Geração do Modelo

A Tabela 8 apresenta o número de pares de classes em restrições fortes e fracas para as instâncias da ITC2019 originais, para as instâncias pré-processadas de Holm et al. (2020b), e para as instâncias pré-processadas neste trabalho. Para comparar os valores, uma média geométrica deslocada foi usada com um valor de deslocamento igual a 10 (ACHTERBERG, 2007). Pode-se observar a partir da análise desta tabela que as técnicas de pré-processamento propostas foram eficazes. As técnicas aplicadas foram capazes de reduzir em média 34,42% dos pares de classes das restrições fortes e 13,50% dos pares de classe das restrições fracas, em comparação com as técnicas de pré-processamento utilizadas por Holm et al. (2020b). Para as instâncias *mary-spr17*, *tg-fal17*, *muni-pdfx-fal17* e *tg-spr18*, foi alcançada uma redução de mais de 90% nos pares de classes das restrições fortes.

Da mesma forma, a Tabela 9 compara as técnicas de pré-processamento acerca do número médio de tempos e salas para as turmas de cada instância. Uma pequena redução média pode ser observada em comparação com o pré-processamento proposto por Holm et al. (2020b). No entanto, uma diferença significativa pode ser observada entre as instâncias pré-processadas e as instâncias ITC2019 originais. Certamente, algumas das técnicas de pré-processamento propostas são realizadas em ambas as abordagens.

[3] Nos casos em que não foi possível realizar o pré-processamento, as instâncias originais de Holm et al. (2020b) foram utilizadas para executar as outras etapas.

Tabela 7 – Status de execução das instâncias.

	Instância	PP	SI	CM	FO
Early	*agh-fis-spr17*	✓	✓	T.E.	✗
	agh-ggis-spr17	✓	✓	✓	✓
	bet-fal17	✓	T.E.	✓	✗
	iku-fal17	✓	✓	T.E.	✗
	mary-spr17	✓	✓	✓	✓
	muni-fi-spr16	✓	✓	✓	✓
	muni-fsps-spr17	✓	✓	✓	✓
	muni-pdf-spr16c	T.E.	✓	M.E.	✗
	pu-llr-spr17	✓	✓	✓	✓
	tg-fal17	✓	✓	✓	✓
Middle	*agh-ggos-spr17*	✓	✓	✓	M.E.
	agh-h-spr17	✓	✓	T.E.	✗
	lums-spr18	✓	✓	✓	✓
	muni-fi-spr17	✓	✓	✓	✓
	muni-fsps-spr17c	✓	✓	✓	✓
	muni-pdf-spr16	T.E.	✓	M.E.	✗
	nbi-spr18	✓	✓	✓	✓
	pu-d5-spr17	✓	✓	T.E.	✗
	pu-proj-fal19	T.E.	✓	M.E.	✗
	yach-fal17	✓	✓	✓	✓
Late	*agh-fal17*	T.E.	✓	M.E.	✗
	bet-spr18	✓	✓	✓	M.E.
	iku-spr18	✓	✓	T.E.	✗
	lums-fal17	✓	✓	✓	✓
	mary-fal18	✓	✓	✓	✓
	muni-fi-fal17	✓	✓	✓	✓
	muni-fspsx-fal17	✓	T.E.	✓	✗
	muni-pdfx-fal17	T.E.	T.E.	M.E.	✗
	pu-d9-fal19	T.E.	✓	M.E.	✗
	tg-spr18	✓	✓	✓	✓

PP pré-processamento;
SI solução inicial;
CM construção do modelo;
FO Fixa-e-Otimiza;
T.E. tempo esgotado ;
M.E. memória excedida;
✓ executado com sucesso;
✗ não executado devido falhas em etapas anteriores.

A Tabela 10 apresenta o tempo de pré-processamento para cada etapa apresentada na Seção 5.1. O tempo $\mathbf{T_1}$ refere-se ao pré-processamento de indisponibilidade de salas, $\mathbf{T_2}$ corresponde ao pré-processamento de restrições de distribuição, $\mathbf{T_3}$ refere-se ao pré-processamento de restrições redundantes, $\mathbf{T_4}$ é o tempo de pré-processamento para pares de classes redundantes e $\mathbf{T_5}$ refere-se à etapa de pré-processamento dos estudantes. A partir da análise desta tabela, em conjunto com as Tabelas 8 e 9, pode-se concluir que o pré-processamento de restrições redundantes e pares de classes redundantes foram os mais efetivos em relação ao tempo/benefício. As instâncias *tg-fal17*, *lums-spr18*, *lums-fal17* e *tg-spr18* não têm estudantes, então o tempo $\mathbf{T_5}$, referente ao pré-processamento da etapa dos estudantes, não foi informado. A etapa de pré-processamento dos estudantes, indicada pelo tempo $\mathbf{T_5}$, é a mais cara em termos de tempo. Em alguns casos, demoraria mais de 24 horas para realizar essa etapa, o que inviabiliza seu pré-processamento.

Tabela 8 – Número de pares de classes nas restrições.

	Instância	Pares de Classe Fortes				Pares de Classes Fracas			
		ITC	Holm	AP	Dif(%)	ITC	Holm	AP	Dif(%)
Early	*agh-fis-spr17*	3.521	3.396	2.870	15,49	769	677	583	13,88
	agh-ggis-spr17	29.475	29.137	3.945	86,46	2.046	1.302	1.189	8,68
	bet-fal17	5.773	4.734	4.191	11,47	6.564	6.514	6.017	7,63
	iku-fal17	26.768	22.924	11.650	49,18	1.117	638	463	27,43
	mary-spr17	40.242	40.176	1.007	97,49	14.903	14.835	1.578	89,36
	muni-fi-spr16	2.113	1.801	1.203	33,20	1.038	977	914	6,45
	muni-fsps-spr17	2.284	2.201	1.488	32,39	851	618	193	68,77
	muni-pdf-spr16c	12.574	12.249	10.624	13,27	3.214	2.930	2.864	2,25
	pu-llr-spr17	891	690	469	32,03	622	329	276	16,11
	tg-fal17	79.299	77.661	5.668	92,70	17.090	17.089	697	95,92
Middle	*agh-ggos-spr17*	6.745	6.568	4.787	27,12	828	782	606	22,51
	agh-h-spr17	2.366	2.325	1.949	16,17	1.247	1.024	895	12,60
	lums-spr18	2.942	2.934	1.540	47,51	651	568	545	4,05
	muni-fi-spr17	1.614	1.421	1.100	22,59	957	903	877	2,88
	muni-fsps-spr17c	6.952	6.907	6.124	11,34	397	183	183	0,00
	muni-pdf-spr16	6.836	6.423	5.215	18,81	2.723	2.050	2.011	1,90
	nbi-spr18	2.131	2.060	1.734	15,83	29	26	26	0,00
	pu-d5-spr17	2.734	2.128	1.640	22,93	6.268	5.798	5.669	2,22
	pu-proj-fal19	18.171	15.101	9.412	37,67	32.031	28.491	21.823	23,40
	yach-fal17	1.977	1.857	1.168	37,10	545	478	454	5,02
Late	*agh-fal17*	45.193	43.970	16.072	63,45	13.596	12.139	5.053	58,37
	bet-spr18	7.652	6.599	5.648	14,41	8.111	8.015	7.446	7,10
	iku-spr18	32.505	29.969	10.560	64,76	779	376	290	22,87
	lums-fal17	3.938	3.908	1.788	54,25	670	586	566	3,41
	mary-fal18	922	817	678	17,01	1.418	1.204	1.016	15,61
	muni-fi-fal17	1.805	1.548	1.060	31,52	1.019	964	907	5,91
	muni-fspsx-fal17	17.542	17.387	8.479	51,23	2.186	1.115	560	49,78
	muni-pdfx-fal17	42.650	41.271	2.587	93,73	7.446	6.229	4.387	29,57
	pu-d9-fal19	4.368	3.317	2.587	22,01	5.966	4.700	4.166	11,36
	tg-spr18	72.444	71.711	4.657	93,51	1.986	1.851	1.846	0,27
	Média*	7.122,22	6.544,82	2.969,66	34,42	1.908,59	1.537,35	1.067,06	13,50

ITC dados das intâncias da ITC2019;
Holm dados das intâncias de Holm et al. (2020b);
AP dados das instâncias da abordagem proposta;
Dif(%) diferença entre os resultados de Holm et al. (2020b) e os resultados obtidos neste trabalho.
Média* média geométrica deslocada com deslocamento igual a 100.000 (ACHTERBERG, 2007).

Finalmente, as Tabelas 11 e 12 comparam, respectivamente, o número de variáveis e restrições dos modelos PLIM gerados por Holm et al. (2020b) e a abordagem de pré-processamento proposta, antes e depois do *presolve* do *Gurobi*. Nesta etapa, apenas as instâncias que passaram sem problemas de tempo ou memória por todas as etapas foram utilizadas. Desta comparação, é possível concluir que, para a maioria das instâncias, as técnicas de pré-processamento propostas podem reduzir ainda mais as dimensões do modelo em relação às propostas por Holm et al. (2020b). Para alguns casos, a redução no número de variáveis antes do *presolve* foi considerável, como no caso da instância *agh-ggis-spr17*, que teve uma redução de cerca de $72,62\%$ em relação aos valores de Holm et al. (2020b). Em relação às restrições, para esta mesma instância, foi possível obter um ganho de $50,57\%$.

Tabela 9 – Média de tempos e salas por classe.

	Instâncias	Média de Tempo por Turma				Média de Salas por Turma			
		ITC	Holm	AP	Dif(%)	ITC	Holm	AP	Dif(%)
Early	*agh-fis-spr17*	117,73	107,00	106,94	0,06	15,92	7,45	7,45	0,00
	agh-ggis-spr17	25,20	19,72	19,70	0,10	7,28	6,35	6,35	0,00
	bet-fal17	23,77	20,78	20,78	0,00	25,43	22,11	22,11	0,00
	iku-fal17	35,36	30,99	30,97	0,06	30,76	27,89	27,88	0,04
	mary-spr17	13,98	13,74	13,74	0,00	13,57	12,22	12,22	0,00
	muni-fi-spr16	16,62	13,99	13,99	0,00	4,82	4,25	4,25	0,00
	muni-fsps-spr17	20,24	16,40	16,39	0,06	3,15	2,60	2,60	0,00
	muni-pdf-spr16c	59,61	45,26	41,21	8,95	11,82	10,92	10,92	0,00
	pu-llr-spr17	9,28	9,21	9,21	0,00	15,23	13,99	13,99	0,00
	tg-fal17	25,86	17,65	17,60	0,28	4,41	3,85	3,85	0,00
Middle	*agh-ggos-spr17*	93,58	86,52	86,52	0,00	10,82	9,29	9,29	0,00
	agh-h-spr17	236,35	227,92	227,47	0,20	25,47	16,15	16,15	0,00
	lums-spr18	43,86	41,59	41,59	0,00	27,19	26,00	26,00	0,00
	muni-fi-spr17	18,92	16,10	15,97	0,81	5,25	5,14	5,14	0,00
	muni-fsps-spr17c	124,74	102,91	100,24	2,59	5,06	4,96	4,96	0,00
	muni-pdf-spr16	32,76	31,75	31,75	0,00	17,47	15,96	15,96	0,00
	nbi-spr18	38,09	36,53	36,48	0,14	4,83	4,68	4,68	0,00
	pu-d5-spr17	11,79	10,89	10,89	0,00	8,77	7,59	7,59	0,00
	pu-proj-fal19	13,43	12,49	12,49	0,00	9,83	8,73	8,73	0,00
	yach-fal17	43,98	42,41	42,30	0,26	4,61	4,48	4,48	0,00
Late	*agh-fal17*	75,55	68,41	68,38	0,04	10,52	8,91	8,91	0,00
	bet-spr18	23,17	20,35	20,35	0,00	25,15	22,01	22,01	0,00
	iku-spr18	32,72	25,62	25,60	0,08	27,72	25,41	25,41	0,00
	lums-fal17	43,50	41,94	41,94	0,00	26,54	25,95	25,95	0,00
	mary-fal18	11,37	10,94	10,94	0,00	15,11	13,56	13,56	0,00
	muni-fi-fal17	16,30	14,08	14,08	0,00	4,94	4,89	4,89	0,00
	muni-fspsx-fal17	67,85	56,11	54,57	2,74	4,42	3,22	3,22	0,00
	muni-pdfx-fal17	66,74	51,56	48,17	6,57	18,48	16,95	16,95	0,00
	pu-d9-fal19	13,89	13,20	13,20	0,00	14,24	13,43	13,43	0,00
	tg-spr18	23,37	14,16	14,14	0,14	5,67	4,63	4,63	0,00
	Média*	34,76	30,47	30,21	0,63	12,02	10,54	10,54	0,00

ITC dados das intâncias da ITC2019;
Holm dados das intâncias de Holm et al. (2020b);
AP dados das instâncias da abordagem proposta;
Dif(%) diferença entre os resultados de Holm et al. (2020b) e os resultados obtidos neste trabalho;
Média* média geométrica deslocada com deslocamento igual a 10 (ACHTERBERG, 2007).

Na média, se comparado com os resultados de Holm et al. (2020b), foi possível obter um ganho de 22,03% e 6,57% em relação ao número de variáveis antes e depois do *presolve* do *Gurobi*, respectivamente. Em relação ao número de restrições, foi possível obter uma redução de 7,65% antes do *presolve*. Esses resultados podem ser explicados pelo menor número de pares de classes nas restrições fortes e fracas devido ao pré-processamento realizado nas instâncias, conforme mostrado na Tabela 8. Além disso, na construção do modelo, foram criadas apenas as variáveis necessárias para a instância, o que reduziu a quantidade em alguns casos. O principal ponto em que o pré-processamento de Holm et al. (2020b) se mostra mais eficiente é na fase de criação de restrições relacionadas à alocação dos alunos, o que leva seu modelo a ter uma quantidade menor de restrições e variáveis em alguns casos. Vale ressaltar também que as técnicas de pré-processamento aplicadas por Holm et al. (2020b) ainda não foram publicadas.

Tabela 10 – Tempos de pré-processamento.

	Instância	T_1	T_2	T_3	T_4	T_5
Early	*agh-fis-spr17*	2,10	2,63	0,11	1,31	3.004,29
	agh-ggis-spr17	0,13	1,82	0,34	9,87	1.521,35
	bet-fal17	0,14	3,00	0,10	2,16	57.578,84
	iku-fal17	0,15	8,19	0,33	8,56	N.A.
	mary-spr17	0,06	1,31	1,71	5,06	3.271,82
	muni-fi-spr16	0,05	0,59	0,04	0,39	248,04
	muni-fsps-spr17	0,04	0,74	0,04	0,43	50,02
	muni-pdf-spr16c	0,28	3,22	0,21	6,62	T.E.
	pu-llr-spr17	0,06	0,61	0,03	0,05	901,59
	tg-fal17	0,06	2,88	0,03	41,95	N.A.
Middle	*agh-ggos-spr17*	0,58	1,99	0,16	2,46	28.250,17
	agh-h-spr17	3,88	2,71	0,03	0,60	13.793,43
	lums-spr18	0,18	1,25	0,08	0,38	N.A.
	muni-fi-spr17	0,04	0,61	0,06	0,27	410.52
	muni-fsps-spr17c	0,09	1,53	0,10	2,22	2.986,02
	muni-pdf-spr16	0,11	2,62	0,05	2,58	T.E.
	nbi-spr18	0,05	0,64	0,06	0,39	306,57
	pu-d5-spr17	0,08	0,66	0,11	2,35	928,67
	pu-proj-fal19	0,39	3,69	1,28	32,33	T.E.
	yach-fal17	0,02	0,68	0,06	0,29	116,67
Late	*agh-fal17*	10,47	6,25	1,38	39,43	T.E.
	bet-spr18	0,13	4,03	0,11	3,28	57.835,13
	iku-spr18	0,15	7,14	0,46	9,90	N.A.
	lums-fal17	0,22	0,76	0,10	0,52	N.A.
	mary-fal18	0,06	0,63	0,03	0,59	4.777,07
	muni-fi-fal17	0,06	0,62	0,07	0,25	257,27
	muni-fspsx-fal17	0,12	2,63	0,14	17,49	3.974,99
	muni-pdfx-fal17	2,12	6,68	0,38	66,05	T.E.
	pu-d9-fal19	0,17	1,21	0,17	1,36	T.E.
	tg-spr18	0,07	2,19	0,02	23,09	N.A.

T_1 pré-processamento da indisponibilidade de salas;
T_2 pré-processamento das restrições de distribuição;
T_3 pré-processamento das restrições redundantes;
T_4 pré-processamento dos pares de turmas redundantes;
T_5 pré-processamento de estudantes;
T.E. tempo esgotado;
N.A. não se aplica, pois a instância não possui estudantes;
Tempo em segundos.

Tabela 11 – Número de variáveis do modelo de PLIM.

Instâncias	**Antes do *Presolve***			**Após o *Presolve***		
	Holm	**AP**	**Dif(%)**	**Holm**	**AP**	**Dif(%)**
agh-ggis-spr17	10.532.658	2.883.679	**72,62**	1.961.118	1.878.653	**4,20**
mary-spr17	1.287.460	648.320	**49,64**	525.927	481.897	**8,37**
muni-fi-spr16	4.350.531	4.447.153	-2,22	2.705.734	2.695.684	**0,37**
muni-fsps-spr17	717.147	496.449	**30,77**	367.581	363.626	**1,08**
pu-llr-spr17	5.542.707	2.861.485	**48,37**	1.890.047	1.795.031	**5,03**
tg-fal17	232.749	73.380	**68,47**	126.314	58.386	**53,78**
lums-spr18	458.344	595.902	-30,01	435.420	400.985	**7,91**
muni-fi-spr17	5.711.500	6.986.041	-22,32	3.878.005	3.858.259	**0,51**
muni-fsps-spr17c	1.466.645	1.285.365	**12,36**	862.299	833.634	**3,32**
nbi-spr18	631.823	590.344	**6,56**	200.008	196.984	**1,51**
yach-fal17	1.336.902	1.145.585	**14,31**	1.010.854	962.742	**4,76**
lums-fal17	448.222	631.032	-40,79	432.121	399.702	**7,50**
mary-fal18	3.307.703	2.727.488	**17,54**	2.013.267	1.997.739	**0,77**
muni-fi-fal17	7.384.314	8.517.121	-15,34	4.950.779	4.927.103	**0,48**
tg-spr18	130.533	64.736	**50,41**	71.679	48.332	**32,57**
Média*	1.544.239,41	1.203.987,94	**22,03**	855.814,57	799.619,98	**6,57**

Holm resultados de Holm et al. (2020b);
AP resultados da abordagem proposta;
Dif(%) diferença entre os resultados de Holm et al. (2020b) e os resultados obtidos;
Média* média geométrica deslocada com deslocamento igual a 100.000 (ACHTERBERG, 2007).

Tabela 12 – Número de restrições do modelo de PLIM.

Instância	**Antes do *Presolve***			**Após o *Presolve***		
	Holm	**AP**	**Dif(%)**	**Holm**	**AP**	**Dif(%)**
agh-ggis-spr17	10.971.106	5.423.267	**50,57**	1.966.325	3.163.392	-60,88
mary-spr17	1.716.685	1.347.270	**21,52**	774.553	940.000	-21,36
muni-fi-spr16	4.958.093	8.213.405	-65,66	3.114.910	5.353.770	-71,88
muni-fsps-spr17	1.003.857	1.124.988	-12,07	532.588	712.561	-33,79
pu-llr-spr17	8.980.995	6.962.711	**22,47**	4.691.996	4.123.078	**12,13**
tg-fal17	597.723	326.845	**45,32**	120.314	81.714	**32,08**
lums-spr18	589.187	516.513	**12,33**	487.767	350.369	**28,17**
muni-fi-spr17	6.475.267	12.018.341	-85,60	4.424.788	7.696.457	-73,94
muni-fsps-spr17c	17.717.840	8.915.121	**49,68**	13.143.821	6.493.444	**50,60**
nbi-spr18	1.955.308	1.771.697	**9,39**	1.220.781	791.460	**35,17**
yach-fal17	4.407.666	3.608.185	**18,14**	3.321.465	2.999.812	**9,68**
lums-fal17	525.567	570.538	-8,56	416.315	353.722	**15,04**
mary-fal18	3.754.624	5.538.220	-47,50	2.316.193	3.883.300	-67,66
muni-fi-fal17	8.032.513	14.850.998	-84,89	5.407.354	9.731.501	-79,97
tg-spr18	645.009	279.478	**56,67**	71.313	69.766	**2,17**
Média*	2.779.945,06	2.567.298,42	**7,65**	1.451.546,61	1.571.806,87	-8,28

Holm resultados de Holm et al. (2020b);
AP resultados da abordagem proposta;
Dif(%) diferença entre os resultados de Holm et al. (2020b) e os resultados obtidos neste trabalho;
Média* média geométrica deslocada com deslocamento igual a 100.000 (ACHTERBERG, 2007).

O modelo PLIM foi implementado em *Python* usando a biblioteca *Python-MIP*. Os tempos de construção e memória utilizada (em Megabytes) dos modelos de cada instância são apresentados na Tabela 13. Os modelos para instâncias que não estão presentes na tabela não foram gerados, uma vez que o tempo limite de 24 horas ou o limite de memória de 32 GB foi atingido. O modelo gerado para a instância *muni-fi-fal17* tem o maior tempo e a maior quantidade de memória necessária para armazenamento. Este modelo levou cerca de 13,5 horas para ser gerado e ocupa um espaço de 4.200 MB. A instância que gerou o modelo em menor tempo (85,72 segundos) e ocupou uma menor quantidade de memória (44 MB) foi a instâncias *tg-fal17*. Um dos fatores que levam a estes resultados para esta instância é a falta de estudantes e a quantidade de restrições de distribuição na instância.

Tabela 13 – Tempo e tamanho do modelo.

Instância	Tempo	Tamanho
agh-ggis-spr17	16.374,49	584
mary-spr17	1.495,44	282
muni-fi-spr16	14.792,22	775
muni-fsps-spr17	322,79	96
pu-llr-spr17	25.483,89	796
tg-fal17	85,71	44
lums-spr18	529,05	1.200
muni-fi-spr17	20.636,44	3.800
muni-fsps-spr17c	4.897,01	702
nbi-spr18	426,24	342
yach-fal17	1.186,88	485
lums-fal17	515,48	1.200
mary-fal18	18.202,11	567
muni-fi-fal17	48.756,24	4.200
tg-spr18	90,31	45

Tempo em segundos;
Tamanho em MB.

6.3.2 Algoritmo Construtivo

A Tabela 14 apresenta o valor da função objetivo e o tempo para construir soluções viáveis iniciais para instâncias da ITC2019. Para gerar a solução inicial, foram utilizadas as instâncias fornecidas por Holm et al. (2020b). Até mesmo a geração de soluções iniciais viáveis (ou seja, soluções que atendam a todas as restrições rígidas) é uma tarefa desafiadora para várias instâncias da ITC2019. Algumas dessas instâncias levaram mais de 1 hora para encontrar uma solução factível. Para as instâncias *bet-fal17*, *muni-fspsx-faal17* e *muni-pdfx-fal17*, não foi possível encontrar uma solução inicial viável em menos de 24 horas. Isso se deve ao grande tamanho das instâncias e à existência de várias restrições difíceis de serem atendidas.

Tabela 14 – Tempo e objetivo das soluções iniciais.

	Instância	Tempo	Objetivo
Early	*agh-fis-spr17*	81,86	33.715
	agh-ggis-spr17	353,00	178.615
	bet-fal17	T.E.	T.E.
	iku-fal17	1.457,34	70.722
	mary-spr17	11,05	56.294
	muni-fi-spr16	3,19	17.243
	muni-fsps-spr17	2,63	275.822
	muni-pdf-spr16c	239,62	46.136
	pu-llr-spr17	4,18	78.227
	tg-fal17	45,73	14.456
Middle	*agh-ggos-spr17*	227,81	66.024
	agh-h-spr17	191,70	49.639
	lums-spr18	8,00	540
	muni-fi-spr17	3,36	19.210
	muni-fsps-spr17c	18,75	721.474
	muni-pdf-spr16	19,82	230.219
	nbi-spr18	3,74	61.260
	pu-d5-spr17	10,69	45.359
	pu-proj-fal19	4.649,49	4.649
	yach-fal17	34,18	25.787
Late	*agh-fal17*	2.277,92	483.139
	bet-spr18	1.594,23	447.951
	iku-spr18	4.036,95	94.944
	lums-fal17	15,43	1.383
	mary-fal18	3,80	43.879
	muni-fi-fal17	3,52	23.540
	muni-fspsx-fal17	T.E.	T.E.
	muni-pdfx-fal17	T.E.	T.E.
	pu-d9-fal19	461,19	432.701
	tg-spr18	6,42	71.024

Tempo em segundos;
T.E. tempo esgotado.

6.3.3 Algoritmo de Fixa-e-Otimiza Multi-Vizinhança

O algoritmo de Fixa-e-Otimiza foi executado primeiro com um limite de tempo de uma hora e, em seguida, com um limite de tempo de 24 horas. O tempo de execução de cada subproblema resultante da configuração de algumas variáveis foi escolhido de acordo com o número de restrições do modelo de cada instância antes do *presolve* do *Gurobi*. Para instâncias com até um milhão de restrições, foi definido um tempo de 100 segundos. Para instâncias que têm entre um e cinco milhões de restrições, foi escolhido um tempo de 200 segundos. Finalmente, para instâncias com mais de cinco milhões de restrições, um tempo de 300 segundos foi escolhido. A Tabela 15 apresenta os tempos utilizados para cada subproblema de cada instância. Essa estratégia foi utilizada porque, em alguns casos, mesmo sendo um subproblema, o solucionador não conseguia passar da fase de relaxação linear em tempos menores.

Os resultados encontrados são mostrados na Tabela 16. Para efeito de comparação, também foram apresentados os resultados dos 1º, 2º e 3º lugares da ITC2019, que podem ser consultados no site do concurso[4]. Como esperado, em todos os casos, os resultados encontrados com o tempo limite de 1 hora não foram melhores do que os resultados de Holm et al. (2020a), 1º colocado, e os resultados de Rappos et al. (2020), 2º colocado. Comparando com o 3º colocado, Gashi e Sylejmani (2020), foi possível obter um resultado melhor para as instâncias *muni-fsps-spr17*, *tg-fal17* e *nbi-spr18*. Em 24 horas de execução, o algoritmo proposto alcançou resultados competitivos. O algoritmo foi capaz de obter melhores soluções para 13 das 15 instâncias, quando comparado com Gashi e Sylejmani (2020) (3º colocado), e quatro instâncias, quando comparado com Rappos et al. (2020) (2º colocado). Este é um resultado notável, pois no ITC2019 nenhuma restrição de tempo ou ambiente computacional foi imposta aos participantes. Seus solucionadores podem ter sido executados por várias semanas ou até meses. Infelizmente, até onde se sabe, nenhum resultado com restrição de tempo foi relatado na literatura para esses casos.

Tabela 15 – Tempo limite para a execução do subproblema no Fixa-e-Otimiza em cada instância.

Instância	**Tempo**
agh-ggis-spr17	300
mary-spr17	200
muni-fi-spr16	200
muni-fsps-spr17	200
pu-llr-spr17	300
tg-fal17	100
lums-spr18	100
muni-fi-spr17	300
muni-fsps-spr17c	300
nbi-spr18	200
yach-fal17	200
lums-fal17	100
mary-fal18	200
muni-fi-fal17	300
tg-spr18	100

Tempo em segundos.

A Figura 11 apresenta gráficos de convergência do algoritmo de Fixa-e-Otimiza proposto para as instâncias *mary-spr17*, *pu-llr-spr17*, *muni-fi-spr17* e *muni-fi-fal17*. A linha pontilhada vertical identifica o tempo de uma hora, e a linha horizontal identifica o melhor resultado encontrado na literatura. Para a instância *mary-fal17*, o algoritmo apresenta uma boa melhora até o tempo de uma hora. Após esse tempo, o objetivo é aprimorado gradativamente. O algoritmo apresenta melhoras consideráveis em até 12 horas de execução para *pu-llr-spr17*. As demais instâncias apresentam melhorias mais significativas em até quatro horas após o início da execução.

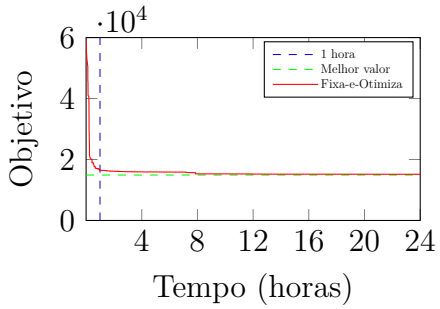
[4] Resultados da ITC2019: <https://www.itc2019.org/results>.

Tabela 16 – Resultados do Fixa-e-Otimiza.

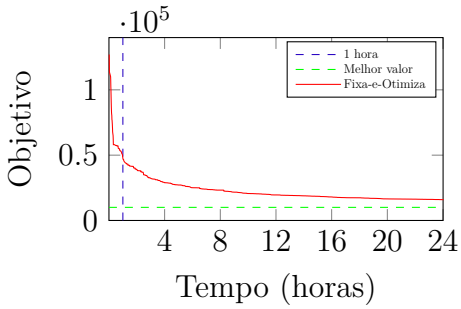
Instância	Holm	Rappos	Gashi	1h	24h
agh-ggis-spr17	34.285	36.616	77.932	88.028	41.270
mary-spr17	14.910	15.021	15.894	21.472	15.158
muni-fi-spr16	3.756	3.844	5.006	5.675	4.325
muni-fsps-spr17	868	883	1.938	1.007	881
pu-llr-spr17	10.038	13.385	16.874	59.433	15.932
tg-fal17	4.215	4.215	8.044	4.215	4.215
lums-spr18	95	114	107	205	110
muni-fi-spr17	3.825	4.289	4.692	6.503	4.479
muni-fsps-spr17c	2.596	3.303	9.222	122.906	5.084
nbi-spr18	18.014	19.055	26.517	19.440	18.059
yach-fal17	1.239	1.844	1.727	5.440	1.941
lums-fal17	349	386	486	511	433
mary-fal18	4.422	5.637	7.199	14.025	5.720
muni-fi-fal17	2.999	3.794	4.712	7.998	4.017
tg-spr18	12.704	12.856	15.992	20.264	13.424

Holm resultados de Holm et al. (2020a) (1º colocado);
Rappos resultados de Rappos et al. (2020) (2º colocado);
Gashi resultados de Gashi e Sylejmani (2020) (3º colocado).

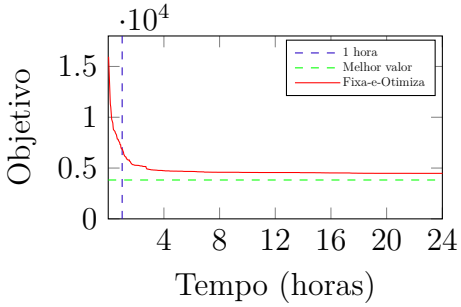
Figura 11 – Gráficos de convergência do algoritmo Fixa-e-Otimiza para algumas instâncias da ITC2019.



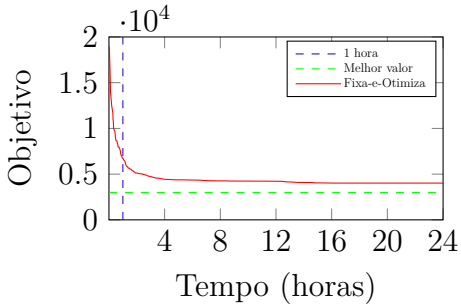
(a) mary-spr17;



(b) pu-llr-spr17;



(c) muni-fi-spr17;



(d) muni-fi-fal17.

As estruturas de vizinhança do algoritmo de Fixa-e-Otimiza apresentados na metodologia deste trabalho (Capítulo 5) foram analisadas separadamente para verificar a eficácia de cada uma. A Tabela 17 apresenta a quantidade de iterações em que cada vizinhança foi chamada, além do número de iterações em que houve alguma melhora no objetivo da solução para um tempo de execução de 24h. Utilizando os dados de todas as instâncias, é possível obter quais foram as estruturas de vizinhança que melhor se adaptaram ao problema como um todo.

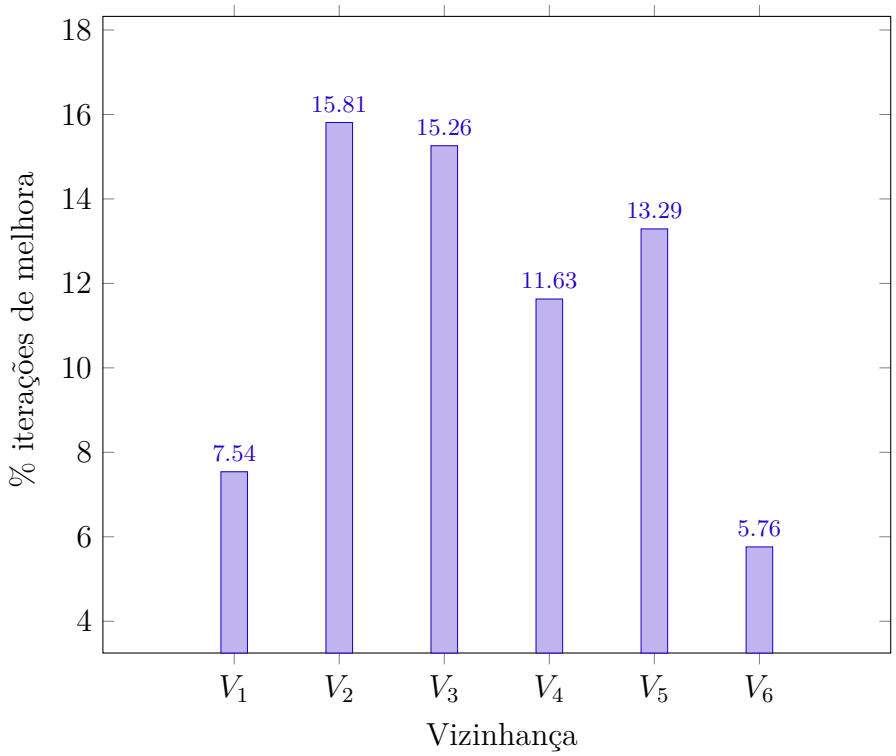
Tabela 17 – Estatísticas sobre as estruturas de vizinhança em cada instância.

Instância	$\mathbf{V_1}$		$\mathbf{V_2}$		$\mathbf{V_3}$		$\mathbf{V_4}$		$\mathbf{V_5}$		$\mathbf{V_6}$	
	It.	**Mel.**	**It.**	**Mel.**	**It.**	**Mel.**	**It.**	**Mel.**	**It.**	**Mel.**	**It.**	**Mel.**
agh-ggis-spr17	280	63	222	59	220	58	232	65	265	65	240	53
mary-spr17	441	35	436	31	414	52	419	47	412	38	421	40
muni-fi-spr16	145	25	159	30	124	34	178	23	146	23	149	25
muni-fsps-spr17	271	12	270	14	280	9	287	17	270	9	177	8
pu-llr-spr17	128	122	115	98	107	76	130	39	110	70	139	99
tg-fal17	1.101	11	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	1.107	23
lums-spr18	1.712	18	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	1.840	8
muni-fi-spr17	130	32	93	17	108	30	108	16	88	20	124	23
muni-fsps-spr17c	588	60	614	37	577	30	632	25	611	27	580	25
nbi-spr18	249	25	234	27	245	20	264	30	256	25	263	28
yach-fal17	370	28	335	33	380	21	349	20	355	15	349	11
lums-fal17	1.280	60	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	1.268	48
mary-fal18	372	106	350	87	399	83	356	59	352	73	380	62
muni-fi-fal17	94	29	88	28	115	40	90	13	85	27	112	30
tg-spr18	1.663	39	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	1.687	26
Total	8.824	665	2.916	461	2.969	453	3.045	392	2.950	373	8.836	509

$\mathbf{V_1}$ seleção aleatória das turmas e dos alunos;
$\mathbf{V_2}$ seleção das classes que possuem alunos em comum;
$\mathbf{V_3}$ seleção dos alunos que possuem um curso em comum;
$\mathbf{V_4}$ seleção das turmas a partir de um conjunto de alunos aleatórios;
$\mathbf{V_5}$ seleção dos alunos a partir de um conjunto de turmas aleatórias;
$\mathbf{V_6}$ seleção das turmas de acordo com as restrições de distribuição;
N.A. não se aplica, pois a instância não possui alunos.

A Figura 12 apresenta um gráfico com a relação entre o número de iterações de uma vizinhança e o número de iterações de melhoria. Como pode ser observado, V_2 e V_3 apresentaram o maior número de iterações de melhoria, enquanto V_1 e V_6 mostraram os piores resultados, pois são vizinhanças com um alto grau de aleatoriedade. Acredita-se que o algoritmo proposto teria um desempenho ainda melhor se algumas vizinhanças menos eficazes fossem desconsideradas.

Figura 12 – Porcentagem das iterações de melhora para cada vizinhança.



7 Considerações Finais

O objetivo deste trabalho foi desenvolver uma heurística matemática do tipo Fixa-e-Otimiza e técnicas de pré-processamento para o Problema de Agendamento de Horários de Cursos Universitários apresentado na ITC2019. Este problema se mostrou complexo e desafiador, onde as características advindas das restrições de distribuição trazem consigo a dificuldade de se criar um modelo de programação matemática simples e eficiente. Outra dificuldade está relacionada ao tamanho das instâncias que, para a maioria dos casos, geram modelos com milhões de restrições e variáveis. As instâncias ainda se mostram mostraram mal formuladas, contendo, por exemplo, uma grande quantidade de restrições redundantes.

Para solucionar a problemática de dimensão das instâncias, uma série de estratégias de pré-processamento foram propostas. Tais estratégias foram capazes de reduzir o tamanho das instâncias, e consequentemente, o tamanho dos modelos. Comparando com os resultados obtidos por Holm et al. (2020b) antes do *presolve* do *Gurobi*, foi possível obter uma redução de 22,03% e 7,65% no número de variáveis e restrições, respectivamente. Após a realização do pré-processamento, um modelo já existente na literatura foi implementado juntamente com uma técnica de Fixa-e-Otimiza Multi-Vizinhança, além de uma heurística construtiva capaz de fornecer soluções iniciais para as instâncias. Todas as estratégias apresentaram bons resultados, sendo que, em alguns casos, foi possível superar os resultados encontrados pelo segundo e terceiro colocados da ITC2019.

Devido ao tamanho e características de algumas instâncias, não foi possível realizar os experimentos para todas as 30 instâncias disponibilizadas, uma vez que metade consumiram uma quantidade de memória maior que a disponível ou demandavam mais de 24 horas na etapa de pré-processamento ou construção do modelo. Então, o algoritmo de Fixa-e-Otimiza foi executado apenas para 15 das 30 instâncias disponibilizadas pela ITC2019.

7.1 Trabalhos Futuros

Como sugestão de trabalhos futuros, tem-se:

- Criação de novas técnicas de pré-processamento que possam auxiliar na redução das instâncias, contribuindo para uma redução no tamanho do modelo;
- Implementação dos algoritmos e das técnicas propostas em uma linguagem mais eficiente, como C++;

- Realização de experimentos para determinar quais estruturas de vizinhança devem ser mantidas no algoritmo proposto;
- Criação de novas estruturas de vizinhança que se adéquem melhor ao problema;
- Criação de estratégias para reformulação ou decomposição do modelo matemático;
- Avaliação de novas técnicas para a geração de solução inicial.

Referências

ABDULLAH, S.; TURABIEH, H. On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. *information sciences*, Elsevier, v. 191, p. 146–168, 2012.

ABRAMSON, D. Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, INFORMS, v. 37, n. 1, p. 98–113, 1991.

ABRAMSON, D.; KRISHNAMOORTHY, M.; DANG, H. et al. Simulated annealing cooling schedules for the school timetabling problem. *Asia Pacific Journal of Operational Research*, Citeseer, v. 16, p. 1–22, 1999.

ACHTERBERG, T. *Constraint Integer Programming*. Tese (Doutorado) — Technische Universität Berlin, 2007.

AL-YAKOOB, S. M.; SHERALI, H. D.; AL-JAZZAF, M. A mixed-integer mathematical modeling approach to exam timetabling. *Computational Management Science*, Springer, v. 7, n. 1, p. 19, 2010.

AMARAL, P.; PAIS, T. C. Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling. *Computers & Operations Research*, Elsevier, v. 72, p. 160–174, 2016.

ANDRADE, P. R. d. L.; STEINER, M. T. A.; GÓES, A. R. T. Optimization in timetabling in schools using a mathematical model, local search and iterated local search procedures. *Gestão & Produção*, SciELO Brasil, v. 26, n. 4, 2019.

ARBAOUI, T.; BOUFFLET, J.-P.; MOUKRIM, A. Preprocessing and an improved MIP model for examination timetabling. *Annals of Operations Research*, Springer, v. 229, n. 1, p. 19–40, 2015.

BADONI, R. P.; GUPTA, D. K.; MISHRA, P. A new hybrid algorithm for university course timetabling problem using events based on groupings of students. *Computers & Industrial Engineering*, Elsevier, v. 78, p. 12–25, 2014. ISSN 0360-8352.

BATTISTUTTA, M.; SCHAERF, A.; URLI, T. Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annals of Operations Research*, Springer, v. 252, n. 2, p. 239–254, 2017.

BELLIO, R.; CESCHIA, S.; GASPERO, L. D.; SCHAERF, A.; URLI, T. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research*, Elsevier, v. 65, p. 83–92, 2016. ISSN 0305-0548.

BOLAND, N.; HUGHES, B. D.; MERLOT, L. T. G.; STUCKEY, P. J. New integer linear programming approaches for course timetabling. *Computers & Operations Research*, Elsevier, v. 35, n. 7, p. 2209–2233, 2008. ISSN 0305-0548. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

BORCHANI, R.; ELLOUMI, A.; MASMOUDI, M. Variable neighborhood descent search based algorithms for course timetabling problem: Application to a Tunisian University. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 58, p. 119–126, 2017. ISSN 1571-0653. 4th International Conference on Variable Neighborhood Search.

BRITO, S. S.; FONSECA, G. H.; TOFFOLO, T. A.; SANTOS, H. G.; SOUZA, M. J. A SA-VNS approach for the high school timetabling problem. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 39, p. 169–176, 2012.

BULCK, D. V.; GOOSSENS, D.; BELIEN, J.; DAVARI, M. The fifth international timetabling competition (itc 2021): Sports timetabling. In: UNIVERSITY OF READING. *MathSport International 2021*. [S.l.], 2021. p. 117–122.

BURKE, E. K.; ECKERSLEY, A. J.; MCCOLLUM, B.; PETROVIC, S.; QU, R. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, Elsevier, v. 206, n. 1, p. 46–53, 2010.

BURKE, E. K.; NEWALL, J. P. A multistage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 3, n. 1, p. 63–74, 1999.

CARVALHO, R. de. *Abordagem heurística para o problema de programação de horários de cursos*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, 2012.

CATALDO, A.; FERRER, J. C.; MIRANDA, J.; REY, P. A.; SAURÉ, A. An integer programming approach to curriculum-based examination timetabling. *Annals of Operations Research*, Springer, v. 258, n. 2, p. 369–393, 2017.

CESCHIA, S.; GASPERO, L. D.; SCHAERF, A. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, Elsevier, v. 39, n. 7, p. 1615–1624, 2012. ISSN 0305-0548.

CHEN, M. C.; SZE, S. N.; GOH, S. L.; SABAR, N. R.; KENDALL, G. A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. *IEEE Access*, IEEE, v. 9, p. 106515–106529, 2021.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. Metaheuristics for high school timetabling. *Computational Optimization and Applications*, Springer, v. 9, n. 3, p. 275–298, 1998.

COSTA, D. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, Elsevier, v. 76, n. 1, p. 98–110, 1994.

CSIMA, J.; GOTLIEB, C. C. Tests on a computer method for constructing school timetables. *Communications of the ACM*, Association for Computing Machinery, New York, v. 7, n. 3, p. 160–163, 1964. ISSN 0001-0782.

DASKALAKI, S.; BIRBAS, T. Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, Elsevier, v. 160, n. 1, p. 106–120, 2005.

DEMIR, L.; TUNALI, S.; ELIIYI, D. T. An adaptive tabu search approach for buffer allocation problem in unreliable non-homogenous production lines. *Computers & Operations Research*, Elsevier, v. 39, n. 7, p. 1477–1486, 2012.

EL-SHERBINY, M. M.; ZEINELDIN, R. A.; EL-DHSHAN, A. M. Genetic algorithm for solving course timetable problems. *International Journal of Computer Applications*, Citeseer, v. 124, n. 10, 2015.

ELLOUMI, A.; KAMOUN, H.; JARBOUI, B.; DAMMAK, A. The classroom assignment problem: Complexity, size reduction and heuristics. *Applied Soft Computing*, Elsevier, v. 14, p. 677–686, 2014.

ER-RHAIMINI, K. Forest growth optimization for solving timetabling problems. *Proceedings of the ITC 2019: International Timetabling Competition*, 2020.

FENG, X.; LEE, Y.; MOON, I. An integer program and a hybrid genetic algorithm for the university timetabling problem. *Optimization Methods and Software*, Taylor & Francis, v. 32, n. 3, p. 625–649, 2017.

FERLAND, J. A.; ROY, S. Timetabling problem for university as assignment of activities to resources. *Computers & Operations Research*, Elsevier, v. 12, n. 2, p. 207–218, 1985.

FONG, C. W.; ASMUNI, H.; MCCOLLUM, B. A Hybrid Swarm-Based Approach to University Timetabling. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 19, n. 6, p. 870–884, 2015.

FONSECA, G. H. G.; SANTOS, H. G. Variable Neighborhood Search based algorithms for high school timetabling. *Computers & Operations Research*, Elsevier, v. 52, p. 203–208, 2014.

FONSECA, G. H. G.; SANTOS, H. G.; CARRANO, E. G.; STIDSEN, T. Modelling and solving university course timetabling problems through xhstt. In: *PATAT*. Údine, Itália: PATAT-2016, 2016. v. 16, p. 127–138.

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

GASHI, E.; SYLEJMANI, K. Simulated Annealing with Penalization for University Course Timetabling. *Proceedings of the ITC 2019: International Timetabling Competition*, ITC2019, 2020.

GASPERO, L. D.; MCCOLLUM, B.; SCHAERF, A. *The Second International Timetabling Competition (ITC-2007): Curriculum-Based Course Timetabling (Track 3)*. Queen's University, Belfast, United Kingdom, 2007.

GASPERO, L. D.; SCHAERF, A. Multi-neighbourhood local search with application to course timetabling. In: SPRINGER. *International Conference on the Practice and Theory of Automated Timetabling*. Berlin, Heidelberg, 2002. p. 262–275.

GINTNER, V.; KLIEWER, N.; SUHL, L. Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, Springer, v. 27, n. 4, p. 507–523, 2005. ISSN 1436-6304.

GÓES, A. R. T.; COSTA, D. M. B.; STEINER, M. T. A. Otimização na programação de horários de professores/turmas: Modelo matemático, abordagem heurística e método misto. *Sistemas & Gestão*, v. 5, n. 1, p. 50–66, 2010.

GOTLIEB, C. C. The construction of class-teacher timetables. In: *Proceedings of the IFIP Congress*. Amsterdam, Holland: North-Holland Pub. Co, 1962. p. 73–77.

GUNAWAN, A.; NG, K. M.; POH, K. L. A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem. *Computers & Operations Research*, Elsevier, v. 39, n. 12, p. 3074–3088, 2012. ISSN 0305-0548.

HADDADI, S.; CHERAITIA, M. Iterated local and very-large-scale neighborhood search for a novel uncapacitated exam scheduling model. *International Journal of Management Science and Engineering Management*, Taylor & Francis, v. 13, n. 4, p. 286–294, 2018.

HERTZ, A. Finding a feasible course schedule using Tabu Search. *Discrete Applied Mathematics*, Elsevier, v. 35, n. 3, p. 255–270, 1992.

HOLM, D. S.; MIKKELSEN, R. Ø.; SØRENSEN, M.; STIDSEN, T. J. R. A MIP based approach for International Timetabling competition 2019. *Proceedings of the ITC 2019: International Timetabling Competition*, ITC2019, 2020.

HOLM, D. S.; MIKKELSEN, R. Ø.; SØRENSEN, M.; STIDSEN, T. J. R. *A MIP Formulation of the International Timetabling Competition 2019 Problem*. Denmark, 2020.

HOOSHMAND, S.; BEHSHAMEH, M.; HAMIDI, O. A tabu search algorithm with efficient diversification strategy for high school timetabling problem. *arXiv preprint arXiv:1309.3285*, 2013.

ISLAM, T.; SHAHRIAR, Z.; PERVES, M. A.; HASAN, M. University timetable generator using tabu search. *Journal of Computer and Communications*, Scientific Research, v. 4, n. 16, p. 28–37, 2016.

ITC2019. *ITC 2019: International Timetabling Competition*. 2019. Disponível em: <itc2019.org>.

JAT, S. N.; YANG, S. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *Journal of Scheduling*, Springer, v. 14, n. 6, p. 617–637, 2011. ISSN 1099-1425.

JUNGINGER, W. Timetabling in Germany – A survey. *INFORMS Journal on Applied Analytics*, INFORMS, v. 16, n. 4, p. 66–74, 1986.

KHADER, A. T.; SEE, A. S. Improving exam timetabling solution using TABU Search. *Journal of Digital Information Management*, Digital Information Research Foundation, v. 3, n. 4, p. 250, 2005.

KRISTIANSEN, S.; SØRENSEN, M.; STIDSEN, T. R. Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, Springer, v. 18, n. 4, p. 377–392, 2015.

LEI, Y.; GONG, M.; JIAO, L.; ZUO, Y. A memetic algorithm based on hyper-heuristics for examination timetabling problems. *International Journal of Intelligent Computing and Cybernetics*, Emerald Group Publishing Limited, 2015.

LEITE, N.; FERNANDES, C. M.; MELICIO, F.; ROSA, A. C. A cellular memetic algorithm for the examination timetabling problem. *Computers & Operations Research*, Elsevier, v. 94, p. 118–138, 2018.

LEITE, N.; MELÍCIO, F.; ROSA, A. C. A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, Elsevier, v. 122, p. 137–151, 2019.

LEMOS, A.; MONTEIRO, P. T.; LYNCE, I. Introducing UniCorT: an iterative university course timetabling tool with MaxSAT. *Journal of Scheduling*, Springer, p. 1–20, 2021. ISSN 1099-1425.

LEWIS, R.; PAECHTER, B.; MCCOLLUM, B. Post enrolment based course timetabling: A description of the problem model used for track two of the second International Timetabling Competition. Cardiff University, 2007.

LÜ, Z.; HAO, J.-K. Adaptive Tabu Search for course timetabling. *European Journal of Operational Research*, Elsevier, v. 200, n. 1, p. 235–244, 2010. ISSN 0377-2217.

MCCOLLUM, B.; MCMULLAN, P.; BURKE, E. K.; PARKES, A. J.; QU, R. *The second International Timetabling Competition: Examination Timetabling Track*. Queen's University, Belfast, 2007.

MOREIRA, J. J. A system for automatic construction of exam timetable using genetic algorithms. *Tékhne-Revista de Estudos Politécnicos*, Instituto Politécnico do Cávado e do Ave, n. 9, p. 319–336, 2008.

MOSCATO, P.; SCHAERF, A. Local search techniques for scheduling problems. In: *Notes of the tutorial given at the 13th European Conference on Artificial Intelligence, ECAI*. Brighton, UK: [s.n.], 1998.

MUKLASON, A.; IRIANTI, R. G.; MAROM, A. Automated Course Timetabling Optimization using Tabu-Variable Neighborhood Search based Hyper-Heuristic algorithm. *Procedia Computer Science*, Elsevier, v. 161, p. 656–664, 2019. ISSN 1877-0509.

MÜLLER, T. Iterative Forward Search Algorithm: Combining Local Search with Maintaining Arc Consistency and a Conflict-Based Statistics. In: *Principles and Practice of Constraint Programming – CP 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 802–802. ISBN 978-3-540-30201-8.

MÜLLER, T.; RUDOVÁ, H.; MÜLLEROVÁ, Z. University Course Timetabling and International Timetabling Competition 2019. In: *Proceedings of 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018)*. Vienna, Austria: PATAT, 2018. p. 5–31.

NGUYEN, K.; NGUYEN, Q.; TRAN, H.; NGUYEN, P.; TRAN, N. Variable Neighborhood Search for a Real-World Curriculum-Based University Timetabling Problem. In: IEEE. *2011 Third International Conference on Knowledge and Systems Engineering*. Hanoi, Vietnam, 2011. p. 157–162.

PAECHTER, B.; GAMBARDELLA, L. M.; ROSSI-DORIA, O. *International Timetabling Competition*. Manno, Switzerland: PATAT, 2002. Disponível em: <http://sferics.idsia.ch/Files/ttcomp2002/oldindex.html>.

PAPOULIAS, D. B. The assignment-to-days problem in a school time-table, a heuristic approach. *European Journal of Operational Research*, Elsevier, v. 4, n. 1, p. 31–41, 1980. ISSN 0377-2217.

PHILLIPS, A. E.; WALKER, C. G.; EHRGOTT, M.; RYAN, D. M. Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, Springer, v. 252, n. 2, p. 283–304, 2017. ISSN 1572-9338.

POCHET, Y.; WOLSEY, L. A. Mixed integer programming algorithms. In: *Production Planning by Mixed Integer Programming*. New York, NY: Springer New York, 2006. p. 77–113. ISBN 978-0-387-33477-6.

POST, G.; GASPERO, L. D.; KINGSTON, J. H.; MCCOLLUM, B.; SCHAERF, A. The third International Timetabling Competition. *Annals of Operations Research*, Springer, v. 239, n. 1, p. 69–75, 2016.

QU, R.; BURKE, E. K.; MCCOLLUM, B.; MERLOT, L. T. G.; LEE, S. Y. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, Springer, v. 12, n. 1, p. 55–89, 2009. ISSN 1099-1425.

RAGHAVJEE, R.; PILLAY, N. A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem. *ORiON*, ORSSA, v. 31, n. 1, p. 39–60, 2015.

RAPPOS, E.; THIÉMARD, E.; ROBERT, S.; HÊCHE, J.-F. International Timetabling Competition 2019: A Mixed Integer Programming Approach for Solving University Timetabling Problems. *Proceedings of the ITC 2019: International Timetabling Competition*, 2020.

SAVINIEC, L.; CONSTANTINO, A. A. Effective local search algorithms for high school timetabling problems. *Applied Soft Computing*, Elsevier, v. 60, p. 363–373, 2017.

SCHAERF, A. Tabu search techniques for large high-school timetabling problems (technical report cs-r9611 1996). *Computer Science/Department of Interactive Systems, Centrum Voor Wiskunder en Informatica (CWI). ISSN*, 1996.

SCHAERF, A. A survey of Automated Timetabling. *Artificial Intelligence Review*, Springer, v. 13, n. 2, p. 87–127, 1999. ISSN 1573-7462.

SORIA-ALCARAZ, J. A.; ÖZCAN, E.; SWAN, J.; KENDALL, G.; CARPIO, M. Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing*, Elsevier, v. 40, p. 581–593, 2016. ISSN 1568-4946.

SOUZA, M. J. F. *Programação de horários em escolas: uma aproximação por metaheurísticas*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2000.

SOUZA, M. J. F.; MACULAN, N.; OCHI, L. S. A GRASP-Tabu Search Algorithm for solving School Timetabling Problems. In: *Metaheuristics: Computer decision-making*. Boston, MA: Springer, 2003. p. 659–672.

TAN, J. S.; GOH, S. L.; KENDALL, G.; SABAR, N. R. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, Elsevier, v. 165, p. 113943, 2021. ISSN 0957-4174.

TASSOPOULOS, I. X.; BELIGIANNIS, G. N. A hybrid particle swarm optimization based algorithm for high school timetabling problems. *Applied Soft Computing*, Elsevier, v. 12, n. 11, p. 3472–3489, 2012.

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

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.

ZUTERS, J. An ensemble of Neural Networks as part of a GA-based model to solve the School Timetabling Problem. In: *Local proceedings of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2006)*. Riga, Letônia: IOS Press, 2006. p. 175–182.