UNIVERSIDADE LUSÓFONA

LUCIANO RIGOLIN DE ALMEIDA

ALOCAÇÃO DE HORÁRIOS UNIVERSITÁRIOS COM PROGRAMAÇÃO INTEIRA

Orientador: Professor Doutor Ricardo Vicente Raposo Crespo de Oliveira

Universidade Lusófona de Humanidades e Tecnologias
Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação
Departamento de Engenharia Informática e Sistemas de Informação

Lisboa
2022

LUCIANO RIGOLIN DE ALMEIDA

ALOCAÇÃO DE HORÁRIOS UNIVERSITÁRIOS COM PROGRAMAÇÃO INTEIRA

Dissertação defendida em provas públicas na Universidade Lusófona de Humanidades e Tecnologias no dia 30/05/2022, perante o júri, nomeado pelo Despacho de Nomeação n.º: 110/2022, de 28/03/2022, com a seguinte composição:

Presidente:

Prof. Doutor Paulo Jorge Tavares Guedes (ULHT)

Vogais:

Prof. Doutor Marko Beko (IST/UL) – Arguente

Orientador:

Prof. Doutor Ricardo Vicente Raposo Crespo de Oliveira (ULHT)

Universidade Lusófona de Humanidades e Tecnologias
Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação
Departamento de Engenharia Informática e Sistemas de Informação

Lisboa
2022

Resumo

A alocação de horários universitários é um problema enfrentado pelas instituições de ensino em todo o mundo, no inicio de cada período, mobilizando uma quantidade significativa de pessoas, tempo e esforço. Não há garantias de que o resultado é um horário de qualidade, atendendo as necessidades dos alunos e professores. Da mesma forma, não há garantias que os recursos necessários (prédios, salas, laboratórios, etc) são alocados da melhor forma. Aplicando técnicas de programação linear, desenvolvemos uma solução genérica para produzir de forma automática, horários escolares considerando as limitações dos professores, necessidades dos alunos e disponibilidade de salas.

Fizemos adaptações na solução genérica para dois casos de estudo. O primeiro caso de estudo foi para os cursos de licenciatura do Departamento de Engenharia Informática e Sistemas de Informação da Universidade Lusófona de Humanidades e Tecnologia, situada em Lisboa. O segundo caso de estudo foi para os datasets do ITC 2007, concurso internacional do problema de alocação do horário universitário.

Para a Universidade Lusófona a nossa formulação obteve um horário de qualidade significativamente superior do que aquele que elaborado de forma manual pela direção dos cursos. Para o ITC 2007 obtivemos resultados próximos aos trabalhos que ficaram nos primeiros lugares da competição.

Palavras-chave— Programação Inteira, Horário Universitário, Otimização

Abstract

The allocation of university timetables is a problem faced by educational institutions around the world, at the beginning of each period, mobilizing a significant amount of people, time and effort. There are no guarantees that the result is a quality schedule, meeting the needs of students and teachers. Likewise, there are no guarantees that the necessary resources (buildings, rooms, laboratories, etc.) are allocated in the best way. Applying linear programming techniques, we developed a generic solution to automatically produce school schedules considering teachers' limitations, students' needs and room availability.

We made adaptations of the generic solution to two case studies. The first case study was for the degree courses of the Department of Computer Engineering and Information Systems at the Lusófona University of Humanities and Technology, located in Lisbon. The second case study was for the ITC 2007 datasets, international competition on the problem of allocation of university timetable.

For Universidade Lusófona, our formulation obtained a timetable of significantly higher quality than the one prepared manually by the direction of the courses. For the ITC 2007 we obtained results close to the works that took the first places in the competition.

Keywords— Integer Programming, University Timetabling, Optimization

Índice Geral

Índice de Quadros

Índice de Figuras

Introdução

Este trabalho foi uma contribuição para a resolução do problema de alocação do horário universitário. O problema de alocação do horário universitário é conhecido na literatura científica como University Course Timetabling Problem - UCTP. É uma derivação do problema do horário.

Para Andrea Schaerf [38], um dos principais pesquisadores da área, o problema do horário consiste no agendamento de uma sequência de disciplinas entre professores e alunos em um período de tempo determinado (normalmente uma semana), satisfazendo um conjunto de restrições. Em outras palavras, elaborar um horário consiste em alocar determinados recursos em um determinado espaço-tempo, seguindo algumas restrições que devem ser satisfeitas tanto quanto possível.

A alocação do horário é um problema enfrentado pelas instituições de ensino em todo o mundo. Em cada início de período letivo, são mobilizadas quantidades significativas de pessoas, tempo e esforço e nem sempre há garantias de que o resultado que se obtém é um horário de qualidade, atendendo as necessidades dos alunos e professores. Também não há garantias que os recursos necessários (prédios, salas, laboratórios, etc) são alocados da melhor forma. [16]

O problema do horário é dividido em três principais categorias: Horário do Ensino Secundário, Horário do Curso Universitário (no qual esse trabalho se enquadra) e Horário do Exame. Existem semelhanças entre o horário do ensino secundário e o horário do curso universitário, porém existem algumas diferenças significativas, que acabam por exigir tratamentos diferenciados nas soluções apresentadas. No horário universitário há certa flexibilidade em relação aos horários das disciplinas, enquanto que no ensino secundário os horários (de início e fim das aulas) são rigorosos e devem ser compactos, ou seja, sem intervalos horários que não tenham aulas.

Outra característica importante do ensino secundário é que as escolas criam turmas e os alunos são incluídos nessas turmas e devem frequentar todas as cadeiras estabelecidas para essa turma. Essa característica também pode existir no curso universitário, porém não é uma regra. Algumas instituições permitem que o aluno defina quais cadeiras pretende estudar, tornando o horário individual para cada aluno.

Outra diferença são os professores. Em geral, no ensino secundário, os professores atuam em tempo integral na escola, ao contrário das universidades, nas quais grande parte dos professores o ensino costuma ser parte da carga de trabalho. [5] [18]

Além das diferenças entre o horário do ensino secundário e o horário do curso universitário, existem ainda, como destacou Souza [30], diferenças entre os diversos regimes educacionais, com variações de região para região e também entre as características de cada instituições de ensino. Isso deixa difícil a generalização da programação do horário, sendo comum o desenvolvimento de soluções específicas que

atendam cada necessidade.

Apesar da dificuldade de generalização, algumas regras e políticas estabelecidas pelas instituições de ensino são comuns e estão presentes em qualquer problema de horário. Em termos científicos, as regras e políticas são chamadas de restrições e estão divididas em duas categorias: restrições rígidas (hard constraints) e restrições flexíveis (soft constraints).

As hard constraints são aquelas restrições que não podem ser violadas. Qualquer horário produzido que atenda todas as hard constraints é considerado um horário factível. Já as soft constraints são aquelas restrições para as quais é permitida a ocorrência de violações. Existirão situações nas quais não é possível atende-las em sua totalidade. Porém, quanto maior o número de soft constraints atendidas, melhor é o resultado do horário. E será considerado ótimo qualquer horário produzido que atenda todas as soft constraints possíveis de serem atendidas.

As restrições comuns a qualquer problema de horário são chamadas de conflitos de primeira ordem. Por exemplo, nenhuma pessoa pode estar em mais de um lugar ao mesmo tempo. Nos problemas de horário, essa regra se aplica a alunos e professores. Da mesma forma, uma sala de aula não pode ter duas aulas no mesmo intervalo de tempo.

Os conflitos de primeira ordem são restrições que devem ser incluídas e tratadas por qualquer solução do problema de horário. São consideradas hard constraints. Por outro lado, as instituições de ensino podem estabelecer restrições adicionais. O objetivo é atender necessidades próprias e também obter um horário de melhor qualidade. As restrições adicionais podem ser hard constraints ou soft constraints (restrições flexíveis).

O problema do horário pode ser tratado como um problema de atribuição multidimensional (Multi-dimensional assignment problems MAP). Os problemas de atribuição multidimensional são uma extensão dos problemas de atribuição, enquanto que, os problemas de atribuição são um caso especial dos problemas de transportes. Os problemas de transportes são problemas de otimização de programação linear inteira com origem na necessidade de transporte de mercadorias. Em sua formulação básica, o problema de transporte procura atender as demandas dos destinos através das capacidades das origens ao menor custo possível.

O problema de atribuição por sua vez, lidam com a questão de atribuir um conjunto de tarefas a um conjunto de agentes com um custo de atribuição associado. É chamado de Problema Generalizado de Atribuição (PGA) e lidam com duas dimensões (tarefas x agentes). Já os problemas de atribuição multidimensional lidam com três ou mais dimensões.

Geralmente o problema de horário possui no mínimo três dimensões: aulas são atribuídas a cadeiras (ou disciplinas), salas e slots de tempo. Dessa forma o problema do horário é classificado como MAP. As dimensões não se limitam às citadas e podem variar conforme as necessidades de cada instituição de ensino. Outras dimensões encontradas na literatura cientifica são: professores, dia de semana, hora, etc.

Para resolver problemas de atribuição são utilizados algoritmos que estão classificados em duas categorias: métodos exatos e métodos heurísticos. Os algoritmos de métodos exatos são aqueles que encontram a melhor solução (solução ótima) para o problema, enquanto que o os algoritmos de métodos heurísticos são aqueles que procuram uma solução factível, mas não garantem que será encontrada a solução ótima. Um dos principais algoritmo de método exato é o Simplex. Já para o método heurístico existem diversos algoritmos, entre eles, algoritmo de busca local, algoritmo de inteligência de enxame e

algoritmos genéticos.

Problemas de atribuição multidimensional estão na lista dos algoritmos considerados mais difíceis de serem resolvidos. Na teoria relativa a complexidade computacional é considerado NP-Hard. Um dos principais teóricos sobre complexidade computacional, Richard Karp [25], estabeleceu uma lista de 21 problemas considerados NP-Complete. Entre os problemas classificados por Karp estão os problemas chamados de 3-dimensional matching (ou acoplamento tridimensional) no qual estão incluídos os problemas de atribuição multidimensional.

Apesar dos problemas de atribuição multidimensional (MAP) terem sido classificados por Karp como NP-Complete, eles são considerados NP-Hard pelo fato dos algoritmos existentes não serem eficiente em resolve-los. O Simplex resolve problemas de atribuição (PGA) com eficiência. Esses problemas possuem uma matriz totalmente unimodular, permitindo resolver pelo Simplex aplicando um relaxamento, que desconsidera a restrição de integralidade das variáveis. No MAP, a matriz não é unimodular e por isso não pode ser resolvido de forma fácil pelo Simplex, sendo necessário aplicar técnicas auxiliares, como por exemplo branch and bound ou branch and cut, tornando-os de difícil resolução.

Além disso, a inclusão de restrições adicionais pelas instituições de ensino tornam o problema ainda mais difícil de resolver, principalmente com a inclusão de soft constraints. As restrições adicionais por si só já incluí novas restrições ao problema, mas as soft contraints também altera a função objetivo. De modo geral, no problema do horário a função objetivo são definidas pelas soft constraints, que estabelecem custos de atribuição das aulas ou penalizações adicionais. O custo de atribuição estabelece um custo de atribuir aulas (a uma determinada sala e slot de tempo, por exemplo) e seguem as necessidades de cada instituição. São aplicados custos maiores para os elementos da matriz multidimensional para os quais se deseja evitar atribuição de aulas.

As penalizações adicionais na função objetivo são penalizações não aplicáveis a atribuição da aula e sim com o intuito de ter um horário melhor e mais organizado. São aplicáveis em diversas situações, um exemplo é obter um horário compacto, evitando slot de tempos vagos entre as aulas atribuídas. Cada slot de tempo vago é considerado uma penalização na função objetivo. Em uma função objetivo de minimização, quanto menor for o número de slots vagos, melhor é o horário produzido.

O problema de horário é classificado como um algoritmo NP - Hard. Em Ciências da Computação, os algoritmos podem ser classificados, conforme a sua complexidade computacional, em algoritmos polinomiais (P) e algoritmos polinomiais não determinísticos (Non-deterministic Polynomial - NP).

Algoritmos polinomiais (P) são aqueles cujo problema pode ser considerado tratável se, e somente se, existir um algoritmo para sua solução cujo tempo de execução é limitado por um polinômio no tamanho da entrada. O tempo de execução cresce de forma polinomial conforme a dimensão dos dados de entrada. Estes são os problemas tipicamente considerados como tratáveis.

Por outro lado, os algoritmos polinomiais não determinísticos (NP) são aqueles que não podem ser resolvidos em tempo polinomial, mas para os quais, de forma simplificada, existem algoritmos que permitem a verificação da solução em tempo polinomial. O tempo de execução cresce de forma supra-polinomial (e.g. exponencial) com o crescimento dos dados de entrada. Por definição, o conjunto P está contido em NP (se existe um algoritmo de complexidade polinomial para a resolução de um problema, então este pode naturalmente ser utilizado para a verificação do mesmo).

Alguns algoritmos NP podem ainda ser classificados como NP-Complete. Os algoritmos NP-Complete

são aqueles para os quais se consegue verificar uma solução de forma eficiente (e.g. em tempo polinomial) e que podem ser usados para simular qualquer outro problema em NP. Por consequência, encontrar uma solução com complexidade polinomial para um problema NP-complete implicaria encontrar uma solução de complexidade polinomial para qualquer problema em NP, tendo como consequência P=NP.

Existem ainda os algoritmos NP-Hard. Algoritmos NP-Hard são aqueles mais difíceis e complexos entre os algoritmos NP-Complete e não somente, pois existem problemas considerados NP-Hard que não estão no conjunto dos problemas NP. A figura 1 demonstra os conjuntos de complexidade computacional.

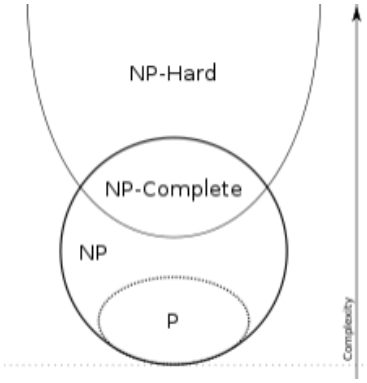

Figura 1: Complexidade computacional [34]

A principal obra sobre complexidade computacional pertence a Karp [25]. Karp usou o teorema de Stephen Cook de 1971 de que o problema de satisfatibilidade booleana (SAT) é NP-complete e classificou o SAT em 21 problemas. Entre os problemas classificados por Karp estão os problemas de 3-dimensional matching (ou acoplamento tridimensional), no qual o problema de atribuição multidimensional está incluído e por consequência, inclui-se também o problema de horário.

Um dos principais algoritmos para resolver problemas de programação linear é o Simplex. Apesar de teoricamente ser um algoritmo de complexidade exponencial, na esmagadora maioria dos problemas práticos a sua complexidade é polinomial. Apesar do problema de atribuição ser considerado um problema NP-Complete, o mesmo tem uma matriz totalmente unimodular. Segundo Hoffman [3], matriz unimodular é qualquer submatriz quadrada de uma matriz A totalmente unimodular com determinante -1, 0 ou 1. Dessa forma, o problema de atribuição pode ser resolvido por um Simplex realizando um relaxamento, que desconsidera a restrição de integralidade das variáveis.

Já nos problemas de atribuição multidimensional (MAP), a matriz não é unimodular e por isso não pode ser resolvido de forma fácil pelo Simplex, sendo necessário aplicar técnicas auxiliares, como por exemplo branch and bound ou branch and cut. Dessa forma, um problema de programação binária com uma matriz multidimensional é considerado um problema difícil de ser resolvido, sendo classificado como NP-Hard.

Pela dificuldade de resolução e de generalização, o problema de horário vem sendo tema de trabalho de diversos pesquisadores por todo o mundo. Bienalmente é realizada a Practice and Theory on Automated Timetabling - PATAT. PATAT é uma conferência internacional que serve como fórum para uma comunidade internacional de pesquisadores, profissionais e fornecedores sobre todos os aspectos da geração de horários auxiliados por computador [35]. Em paralelo, os organizadores da PATAT organizam a International Timetabling Competition - ITC. O ITC é uma competição internacional de horários no

qual os competidores desenvolvem soluções para resolver instâncias com dados reais disponibilizados pela organização e são premiados pelos seus resultados.

1. Trabalhos relacionados

Como já referido na introdução, em função da dificuldade de resolução e de generalização, o problema de alocação do horário universitário é tema de estudo de diversos pesquisadores. Os pesquisadores, em sua grande maioria, elaboram soluções para atender as necessidades das instituições de ensino utilizadas como fonte de pesquisa. Como também já foi referenciado, as necessidades de cada instituição de ensino são diferentes e uma solução que atende uma determinada instituição de ensino não irá atender as necessidades de outras instituições de ensino. Portanto, existem diversas soluções disponíveis.

As soluções desenvolvidas para resolver o problema de alocação do horário universitário estão classificados em duas categorias de algoritmos: métodos exatos e métodos heurísticos. Segundo Góes [5], os algoritmos de métodos exatos encontram a melhor solução (solução ótima) para o problema, quando esta solução existe, satisfazendo todas as restrições impostas. O segundo procura uma solução, mas não garante que esta seja a solução ótima.

A criação de um algoritmo envolve duas premissas: tempo de execução aceitável e que a solução desenvolvida encontre a solução ótima. A utilização de métodos exatos garante que a melhor solução será encontrada, porém nem sempre o tempo de execução para encontrar a solução ótima é um tempo aceitável. Já para os métodos heurísticos, em geral, os tempos de execução geralmente são tempos mais aceitáveis, porém, não há garantias que foi encontrada a solução ótima. Os algoritmos heurísticos exigem o estabelecimento de um critério de parada, enquanto os algoritmos exatos irão parar quando a solução ótima for encontrada.

Neste capítulo serão apresentadas as principais técnicas utilizadas para a sua resolução, demonstrando as suas vantagens e desvantagens.

1.1 Métodos Exatos

Como já foi referido anteriormente, os algoritmos de métodos exatos irão sempre procurar a melhor solução para o problema, a solução ótima. O problema a ser solucionado deve ser convertido em uma solução matemática (programação linear) e o mesmo é resolvido utilizando o algoritmo Simplex e suas ramificações.

Simplex

O principal método para resolver problemas de programação linear é o método Simplex. O Simplex criado por George B. Dantzig. O seu funcionamento consiste, de forma resumida, na construção de uma

solução básica inicial e, a partir desta, seguindo um critério, é escolhido uma nova solução básica que aumente (ou diminua) o valor da função objetiva. O método Simplex é detalhado no apêndice B.

O Simplex é eficaz em resolver problemas de programação linear. Porém, quando são introduzidas variáveis inteiras, como é o caso do problema do horário universitário, o problema torna-se muito mais difícil de resolver, sendo necessário a aplicação de alguns métodos auxiliares ao Simplex.

Método Branch and bound

Um desses métodos é o Branch and Bound. Foi criado com o objetivo de resolver problema de programação inteira. Segundo Alves [37], consiste na ramificação sucessiva do conjunto de soluções possíveis do problema em subconjuntos e na limitação do valor ótimo da função objetiva de modo a excluir os subconjuntos que não contenham a solução ótima. O passo a passo do Branch and Bound é detalhado no apêndice C.

Método dos Planos de Corte

Outro método para resolver problemas de programação inteira é o método dos Planos de Corte. Foi criado por Ralph E. Gomory em 1958. O método de Plano de Corte realiza cortes na solução principal. Os cortes são acrescenta, sucessivamente, novas restrições que eliminam parte das soluções, inclusive a solução ótima não-inteira, até encontrar a melhor solução inteira para o problema [5].

Este método não elimina qualquer solução inteira e atinge a solução ótima para o problema de programação inteira após um número finito de cortes. Segundo Alves [37], apesar dessa propriedade, o método dos Planos de Corte caiu em desuso devido ao grande esforço computacional envolvido para resolução de problemas de grandes dimensões. Apesar de o número de cortes ser finito, o número pode ser muito elevado e a cada novo corte, uma nova restrição é adicionada ao problema original, tornando a sua complexidade crescente. Outra desvantagem citada por Alves é que caso se interrompa o método antes de ele chegar ao fim, não se dispõe de nenhuma solução inteira, mesmo que não seja a solução ótima.

Método Branch and Cut

O método mais utilizado atualmente para resolver problemas de programação inteira é o método Branch and Cut. Branch and Cut é a combinação do método Branch and Bound com o método dos Planos de Corte. Métodos de Planos de Corte melhoram o relaxamento do problema para se aproximar mais do problema de programação inteira enquanto algoritmos Branch and Bound procedem por uma abordagem sofisticada de dividir e conquistar para resolver problemas [24].

Outros métodos exatos

Além do método Simplex, é considerado como método exato o Métodos de Penalidades e o Método de Barreira. São métodos que transformam o problema restrito original em um problema irrestrito equivalente. Dois problemas são equivalentes quando possuem a mesma solução.

Os métodos de penalidades e de barreira diferenciam entre si na penalização. Enquanto o método de penalidade penaliza a violação de qualquer restrição, o método de barreira penaliza qualquer tentativa de sair da região factível do problema.

Utilização de métodos exatos em UCTP

Diversos autores utilizam o método exato para resolver o problema de alocação do horário universitário. Em 2017, Guilherme Brandelli Bucco, Camilo José Bornia-Poulsen e Denise Lindstrom Bandeira [16] apresentaram uma solução utilizando programação linear que procurou elaborar o horário de uma Universidade Federal no Brasil. A principal característica da solução foi o foco na redução dos gastos. A solução permitiu a universidade reduzir em 87 % os gastos com aluguéis, limpeza, manutenção predial e segurança.

Os autores consideraram que a atribuição da sala de aula não era em si um problema. A quantidade de salas disponíveis superam o número necessário em função da carga horária total. Isso permitiu criar uma solução que foi dividida em dois subproblemas: 1 - Construção das grades horárias conforme as regras da universidade e 2 - Atribuição das salas de aulas.

$$min : \sum_{u \in U} a_u \cdot m_u \tag{1.1}$$

O subproblema 2 foi o responsável em reduzir os gastos ao incluir na função objetivo o custo de cada unidade predial. A formula 1.1 apresenta a função objetivo utilizada pelos autores, onde U é um conjunto das unidades prediais disponíveis, a_u é um vetor de custos de utilização da unidade u e m_u é um vetor de variáveis binárias que indicam se uma determinada unidade predial u tem aula atribuída ou não. É considerado que uma unidade predial tem aula atribuída quando é atribuído aula a qualquer sala dessa unidade predial.

Duas restrições foram incluídas para o correto preenchimento da variável m_u. A primeira restrição preenche a variável binária $v_{e,u}$, que indica que a turma e tem aula na unidade u. E por fim, a segunda restrição utiliza a variável $v_{e,u}$ para preencher a variável m_u e obter as unidades u com aulas atribuídas.

Ao utilizar o custo de utilização da unidade predial, permitiu não só reduzir a quantidade de unidades prediais com aulas atribuídas como também utilizar as unidades prediais com menor custo. Porém, a quantidade final de unidades prediais utilizadas pode ser não a menor possível, mas certamente o custo de utilização será o menor possível. Isso ocorre quando existem unidades prediais com grandes dimensões, mas com custo de utilização elevado e ao mesmo tempo, unidades prediais menores e com custo de utilização menor que somadas equivalem em capacidade a unidade predial maior, mas o custo somado é menor. Nesse caso, o resultado ótimo é atribuir aulas às unidades prediais menores. O número de unidades prediais utilizadas é maior com custo final menor.

Outra forma comum para elaborar a função objetivo é a utilização de penalizações. Gerald Lach e Marco E. Lubbecke [20] em seu trabalho utilizaram penalizações na função objetivo para resolver as instâncias do ITC 2007. Entre as restrições estabelecidas pelo ITC 2007, algumas são soft constraints, portanto, permite violações. Para cada soft contraint, Lach e Lubbecke estabeleceram variáveis de penalização e as incluíram na função objetivo.

$$min : \sum_{p \in P} \sum_{s \in S} \sum_{c \in C} obj_{s,c,p} \cdot y_{s,c,p} + \sum_{c \in C} 5 \cdot w_c + \sum_{cu \in CU} \sum_{p \in P} 2 \cdot v_{cu,p} \tag{1.2}$$

A formula 1.2 demonstra a função objetivo elaborada por Lach e Lubbecke. As variáveis de penalização são $y_{s,c,p}$, w_c e $v_{cu,p}$. A variável $y_{s,c,p}$ penaliza atribuir aulas para um curso c, no período p em uma sala com capacidade inferior ao número de alunos matriculados no curso. Já $obj_{s,c,p}$ representa o número de alunos acima da capacidade da sala. Cada aluno acima da capacidade gera uma penalização no resultado final.

Já variável w_c é responsável em penalizar a atribuição de aulas em um número de dias inferior ao estabelecido para a cadeira c. O resultado final da w_c é o cálculo do número de dias estabelecidos para a cadeira c menos o número de dias com aulas atribuídas. Portanto, cada dia a menos, é considerado uma penalização com peso 5 (peso que foi estabelecido pela organização do ITC 2007).

Por fim, a variável binária $v_{cu,p}$ penaliza aulas isoladas para o currículo cu no período p. É considerado um horário ótimo aqueles horários onde todos os currículos estabelecido não possuam aulas isoladas. Portanto, cada aula isolada gera uma penalização com peso 2. O resultado final da variável $v_{cu,p}$ indica se um determinado currículo possuí aula isolada em um determinado período.

O tratamento das aulas isoladas foi um dos destaques do trabalho de Lach e Lubbecke. Para resolver os autores incluíram duas restrições. A primeira restrição é o somatório das aulas para cada currículo e cada período com o objetivo de preencher a variável auxiliar $r_{cu,p}$. A variável $r_{cu,p}$ é uma variável binária e indica se há aula atribuída para um determinado currículo em um determinado período. Como na solução existe uma restrição para garantir que um currículo só terá no máximo uma aula atribuída em um determinado período, o somatório será sempre 0, quando não há aula atribuída no período, ou 1, quando há aula atribuída no período, garantido o correto preenchimento da variável $r_{cu,p}$.

A segunda restrição para tratar das aulas isoladas utiliza as variáveis $r_{cu,p}$ e irá preencher a variável $v_{cu,p}$, que é incluída na função objetivo, penalizando cada aula isolada encontrada. Para cada currículo e período, é criada a restrição $-r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} \leq 0$. Essa restrição atribuí valor 1 para a variável $v_{cu,p}$ quando existe aula atribuída em $r_{cu,p}$ e não há aulas atribuídas em $r_{cu,p-1}$ e $r_{cu,p+1}$.

Em 2008, M Akif Bakir e Cihan Aksop [27] desenvolveram um modelo de programação inteira binária para o agendamento de cursos do Departamento de Estatística da Universidade Gazi, Turquia. No modelo proposto pelos autores a função objetivo é formada por cinco termos que medem a insatisfação de docentes e alunos, ou seja, procura gerar um horário que seja adequado tanto para os alunos quanto para os professores.

Os cinco termos que determinam a insatisfação total são: insatisfação dos alunos do período matutino, insatisfação dos alunos do período noturno, insatisfação dos professores, insatisfação dos professores não docentes (professores que não possuem dedicação exclusiva com a Universidade) e insatisfação dos alunos reprovados.

A insatisfação dos alunos do período matutino é medida pelas aulas no período noturno e as dos alunos do período noturno é medida pelas aula ministradas no período da manhã. A insatisfação dos professores e professores não docentes é medida por períodos que não desejam dar aula. E a insatisfação dos alunos reprovados é medida pela sobreposição dos cursos reprovados com os cursos dos semestres subsequentes.

O objetivo é minimizar o total de insatisfações, porém, a medida da insatisfação não é a mesma entre alunos, professores e não docentes. A direção do departamento estabeleceu que a insatisfação dos professores e não docentes são maiores que as dos alunos. E indo além, a insatisfação dos alunos reprovados é superior aos demais alunos. Em outras palavras, um horário é considerado ótimo quando atende as exigências dos alunos, professores e não docentes. Porém, não sendo possível atender todas as exigências, um horário será melhor quando conseguir atender o máximo das exigências do professores e não docentes.

A solução adotada pelos autores foi atribuir pesos diferentes para cada um dos termos. Para os termos que medem a insatisfação dos professores e não docentes, foi adotado peso 10. Para o termo dos alunos reprovados, foi adotado peso 5 e os termos dos demais alunos (matutino e noturno), peso 1.

Em 2014, André Renato Villela da Silva [6] desenvolveu uma solução utilizando programação inteira mista para a Universidade Federal Fluminense. De certo modo, a função objetivo utilizada pelo autor também trata das insatisfação (ou satisfações) de docentes e alunos. Em relação aos docentes, o trabalho procura evitar aulas em excesso ao longo do dia. Por questões de saúde vocal cada docente pode indicar o número de aulas por dia que acredita ser o ideal para si. Cada aula em excesso ao fim de um dia equivale a uma penalização na função objetivo.

Ainda em relação aos docentes, a universidade permite aos docentes estabelecerem um número ideal de dias que podem dar aulas na semana. Cada dia que ultrapassar o número, estabelece uma penalização. Além disso, é ideal que os dias que cada docente tenham aulas atribuídas sejam de forma consecutivos, ou seja,dias sem aulas entre dias com aulas. Cada dia sem aula entre dias com aulas também representou uma penalização.

Quanto aos alunos, o excesso de aulas atribuídas em um determinado dia representa uma penalidade. Por questões didáticas, a universidade estabelece um limite diário de aulas que é considerado ideal. Cada aula além do limite estabelecido representa uma penalização.

Silva ainda incluiu duas outras penalizações. A Universidade Federal Fluminense considera que um horário é melhor quando não há buracos entre as aulas. É considerado um buraco quando há pelo menos uma hora vaga entre aulas atribuídas. No trabalho de Silva, foi levado em consideração buracos tantos para docentes como para os alunos.

Para tratar dos buracos dos docentes, Silva criou variáveis binárias $BD_{d,s,k,t}$ que indicam a existência de um buraco de tamanho k, após a aula atribuída em uma determinada hora s, para um determinado docente t em um determinado dia da semana d. Também foi permitido a cada docente t estabelecer um penalidade $pBD_{k,t}$ para cada tamanho de buraco k.

Em sua solução, Silva trabalhou com o número de horas por dia estabelecido em sete horas. Isso permitiu criar as variáveis $BD_{d,s,k,t}$, sendo que o máximo do tamanho k é 5 (quando o docente irá lecionar na primeira e última hora do dia). A restrição $y_{d,s,t} - y_{d,s+1,t} + y_{d,s+k,t} - BD_{d,s,k,t} \leq 1$ foi adicionada para obter $BD_{d,s,k,t}$. Na restrição $y_{d,s,t}$ são variáveis binárias que indicam que o docente t tem aula atribuída no dia d e na hora s.

Foi incluído na função objetivo a variável $BD_{d,s,k,t}$ com o peso $pBD_{k,t}$. Uma solução semelhante foi incluída para tratar dos buracos no horário do alunos. A penalização para buraco de tamanho k foi estabelecido pela direção da Universidade.

Neste presente trabalho também é apresentado uma solução para o tratamento de buracos para os

quais foi dado o nome de vazios. Diferente do trabalho de Silva, o tamanho dos vazios é irrelevante, sendo apenas considerados horários melhores aqueles horários que apresentam uma menor quantidade de vazios.

1.2 Métodos Heurísticos

Ao contrário do método exato, o método heurístico não garante a obtenção do resultado ótimo. De certo modo, apesar do resultado ótimo ser o desejável, nem sempre obter o resultado ótimo é o foco dos algoritmos heurísticos, ficando o foco em obter resultados de qualidade com tempos de execução aceitável.

Partindo de uma solução viável, os algoritmos heurísticos baseiam-se em sucessivas iterações em direção a um ponto ótimo. Soluções de boa qualidade serão geradas, melhores que as já conhecidas, porém, não há garantia de que irá encontrar a solução ótima. Para Bueno [19], essa subjetividade, ou falta de precisão dos métodos heurísticos, não se trata de uma deficiência, mas uma particularidade análoga à inteligência humana. Muitas vezes, no cotidiano, os problemas são resolvidos sem que se tenha conhecimento de qual seria o melhor resultado. Apesar de não serem resolvidos com o resultado ótimo, os problemas são resolvidos e com certa qualidade. Durante certo tempo, os resultados são melhorados de forma sucessivamente. Isso equivale, de certo modo, ao funcionamento dos algoritmos heurísticos que, a cada nova iteração melhores resultados são obtidos. As iterações ocorrem até que seja atingido o critério de parada estabelecido.

Diversos algoritmos são considerados como algoritmos heurísticos. A seguir são tratados os principais utilizados para resolver o problema do horário universitário.

Algoritmo de busca local

A pesquisa de busca local (Local Search) é uma função, método ou algoritmo que seleciona uma solução inicial a partir de um ponto qualquer de um conjunto, fazendo uma avaliação dos valores próximos a posição atual (chamados também de vizinhos ou vizinhança), elegendo uma solução possível (chamada de ótimo local). Após a escolha, é feito um salto entre os valores disponíveis, e novamente os vizinhos são avaliados. Essa operação é realizada repetidamente buscando gerar diversas soluções ótimas locais. [31]

A vizinhança é percorrida sempre em busca de uma solução melhor, repetindo o processo sempre que uma nova solução é encontrada. O algoritmo é encerrado quando não é mais encontrado uma solução melhor, nesse caso foi encontrada a solução ótima, ou conforme um critério de parada estabelecido. Quando o processo é encerrado através do critério de parada, não há garantias de que a solução ótima foi encontrada.

Tomas Muller and Roman Bartak Hana Rudova [41], em 2005, desenvolveram um algoritmo de busca local para a Universidade de Purdue, nos Estados Unidos. Partindo de uma solução viável, o algoritmo realiza iterações até que o critério de parada seja alcançado. Em cada iteração, são executadas as seguintes operações para a atribuição de uma aula: seleção da variável, escolha do valor para a variável selecionada, remoção das variáveis conflitantes da solução e atribuição do valor escolhido para a variável

selecionada. No algoritmo, cada variável equivale a uma aula e o valor corresponde a atribuição da variável (aula) a uma sala, dia e hora.

A seleção da variável é dividida em dois critérios. No primeiro critério são escolhidas as variáveis ainda sem atribuição. A variável pode ser selecionada de forma aleatória ou utilizando o principio de primeira falha (first-fail principle). O princípio da primeira falha estabelece a dificuldade de atribuição da variável. O segundo critério para a seleção da variável é selecionar uma variável já atribuída. Esse critério só ocorre quanto todas as variáveis já estão atribuídas, mas ainda não foi atingida a condição de parada. A variável escolhida é aquela cuja mudança de valor pode oferecer melhor oportunidade de melhoria da solução.

Após a seleção a variável, é realizado a escolha do valor. A escolha do valor consiste na procura de um valor que atenda os requisitos da variável e também que cause menos problemas durante a pesquisa futura. Ou seja, um valor com potencial mínimo para conflitos futuros com outras variáveis.

O próximo passo é a remoção das variáveis conflitantes da solução. A vizinhança é analisada a procura de variáveis já atribuídas cujo valor, em comparação ao valor escolhido no passo anterior, irá representar a violação de hard constraints. Essas variáveis então passa para o estado de não atribuídas.

E por fim, o valor escolhido é atribuído a variável selecionada e uma nova solução s_{new} é gerada. Se a nova solução gerada for de maior qualidade que a melhor solução já encontrada (s_{best}), então s_{new} passa a ser a solução s_{best}.

Uma característica interessante do trabalho de Muller e Rudova é que, atingido o critério de parada, o algoritmo sempre irá devolver uma solução viável, mesmo que essa solução seja incompleta. Os autores consideraram como viável aquelas soluções que não violem hard constraints. Se todas as variáveis estiverem atribuídas, essa solução é considerada completa. Uma solução é considerada incompleta quando existem variáveis não atribuídas.

Obter uma solução viável incompleta pode ser útil em casos complexos. Os autores defendem duas alternativas nesse caso. Concluir o horário de forma manual ou então executar novamente o algoritmo utilizando a solução incompleta obtida como solução de start do algoritmo, com critérios de parada diferente, tentando melhorar o resultado.

Os resultados obtidos apontaram que foi possível obter solução viável completa em 95% das necessidades da Universidade de Purdue.

Em 2013 Anmar Abuhamdah, Masri Ayob, Graham Kendall e Nasser R. Sabar [7] desenvolveram um algoritmo de busca local baseado em população (population based local search algorithm - PB-LS) para problemas do horário universitário. O PB-LS começa com uma solução inicial e explora iterativamente as soluções vizinhas, buscando uma melhor. A solução de vizinhança é obtida modificando a solução atual usando uma ou mais estruturas de vizinhança

O trabalho foi dividido e três fases: heurística de maior grau, busca por vizinhança e busca tabu. Na fase 1, os cursos são classificados conforme a quantidade de alunos com os quais têm em conflito com outros cursos. O curso com maior número de conflitos é selecionado primeiro e é atribuído ao mesmo uma sala e um timeslot (dia e hora), escolhidos de forma aleatória.

A fase 2 e 3 realizam movimentos de busca por vizinhança. Realizando movimentos de troca, novas soluções vizinhas são estabelecidas. Os autores estabeleceram dois tipos de movimentos. No primeiro tipo o movimento é realizado com a troca do dia e hora de um curso para um dia e hora vazio e o segundo

tipo acontece quando ocorre a troca da sala, dia e hora entre dois cursos. Em ambos os movimentos, os cursos são escolhidos aleatoriamente entre aqueles cursos que violam alguma restrição e a nova solução é aceita se o movimento não violar nenhuma restrição rígida. Na fase 3 o algoritmo mantém uma lista tabu com as soluções vizinhas com o objetivo de evitar que um determinado movimento volte a ocorrer. A fase 2 é encerrada após 10 iterações sem melhorias e a fase 3 após 1000 iterações sem melhorias.

O algoritmo PB-LS desenvolvido foi impulsionada pela adição de um algoritmo de busca local de emulação gravitacional (gravitational emulation local search - GELS). O algoritmo GELS é baseado nos princípios naturais da atração gravitacional. A atração gravitacional aumenta conforme o tamanho do objeto e pela sua proximidade à outros objetos. Isso significa que um determinado objeto será mais fortemente atraído por objetos maiores e mais próximos.

Os autores estabeleceram o valor de direção que indica a atração gravitacional entre cada solução vizinha produzida. O valor de direção é calculado pela formula $F = CU - CA$, onde CU corresponde ao valor solução atual S_{best} (melhor solução encontrada até o momento) e CA é o custo da solução candidata. O valor de direção mais alto indica o melhor potencial para melhorar a solução em vez de outras soluções.

É realizado o movimento de troca na solução vizinha S_0 com maior valor direção em relação a S_{best}. Se qualidade da nova solução vizinha gerada S_{0*} for maior que S_{best}, então S_{0*} passa a ser S_{best} ou se S_{0*} for melhor que S_0, então S_{0*} passa a ser S_0. Por fim, ao não trazer melhorias, S_{0*} é adicionada a lista tabu. S_0 será descartado após 10 iterações sem gerar um nova solução vizinha com maior qualidade.

A fim de avaliar a eficácia do algoritmo, a solução foi testada com os datasets utilizados por Socha [14]. Os resultados superaram o trabalho de Socha, indicando que a solução é adequada para resolver problemas de horários de cursos universitários.

Ayla Gülcü e Can Akkan [8] em 2018 apresentaram na Conferência Internacional sobre a Prática e Teoria do Calendário Automatizado (PATAT-2018) um algoritmo de busca local para resolver as instâncias do ITC 2007. Os autores assumiram que uma interrupção pode ocorrer na forma de um período para o qual um evento de um curso foi atribuído, deixando de ser viável para aquele evento. Nesse momento, o horário é atualizado, ou seja, ocorre um movimento por uma solução melhor. A avaliação da solução é realizada por dois critérios. Critério de penalidade das restrições flexíveis violadas e o critério pela robustez da solução.

O cálculo da robustez requer encontrar para cada aula interrompida o movimento que produz a penalidade incremental mínima e que restabelece a viabilidade. Foram estabelecidos três tipos de movimentos: (1) mover apenas a aula interrompida para um vazio (chamado de movimento simples), (2) trocar a aula interrompida por outra aula (chamada de movimento de troca) e (3) o movimento em cadeia Kempe que permite que uma palestra seja movida de forma mais flexível para um novo intervalo de tempo. O movimento em cadeia de Kempe consiste pelo estabelecimento de uma cadeia formada pela aula interrompida e por aulas vizinhas, escolhidas aleatoriamente. É realizado um movimento simples ou um movimento de troca de toda a cadeia. A robustez da solução é definida pelo o movimento que produz o custo mínimo de penalidade entre todos os movimentos viáveis.

O critério de parada utilizado pelos autores foi o tempo limite estabelecido pelos organizadores do ITC 2007. Os resultados obtidos colocariam o trabalho de Ayla Gülcü e Can Akkan entre os finalistas da competição.

Algoritmo de inteligência de enxame

Os algoritmos de inteligência de enxame pertencem à família de técnicas baseadas em população com base em algum tipo de agente interagindo localmente entre si e com seu ambiente.

Um dos principais algoritmos de inteligência de enxame são os algoritmos de otimização por colônia de formiga (Ant Colony Optmization - ACO). O algoritmo é baseado no comportamento de coleta de alimentos das formigas. As formigas, ao caminharem do ninho até à fonte de alimento, liberam uma substância química chamada de feromônios. Quanto maior a quantidade de feromônios em uma trilha, mais formigas são atraídas, reforçando a importância da mesma.

Serapião [2] em seu trabalho cita um experimento realizado com formigas reais que demonstrou como é o processo. Foi colocado um ninho de formigas em um aquário com uma fonte de alimentos na outra ponta. Existiam dois caminhos entre a colônia e a fonte de alimentos, com distâncias diferentes. Cada formiga segue um caminho aleatório para buscar comida pela primeira vez. Como o tempo para ir e voltar no caminho mais curto é menor, as formigas que escolheram esse caminho depositavam uma quantidade maior de feromônios em relação ao outro em um mesmo determinado intervalo de tempo. A intensidade de feromônios no caminho mais curto passa a estar tão alto, que a grande maioria das formigas passou a optar por esse caminho.

Os algoritmos de otimização por colônia de formiga simulam esse comportamento. São *"formigas artificiais"* que liberam *"feromônios artificiais"* durante o seu trajeto, criando *"trilhas de feromônios artificiais"*. As trilhas como maior quantidade de *"feromônios artificiais"* são os melhores resultados.

Dorigo [29] criou o que ele chamou de *Meta-heurística de Otimização da Colônia de Formigas*. É considerado por muitos autores como o algoritmo genérico para os algorítimos ACO. É composto pelos seguintes passos:

1. Definir parâmetros e inicializar trilhas de feromônio;
2. Soluções são construídas pelas formigas artificiais;
3. Aplica busca local (opcional);
4. Atualizar feromônios;
5. Critério de parada. O algoritmo deve parar? Em caso de resposta negativa volte ao passo 2, caso contrário a melhor solução construída pelas formigas é uma solução para o problema, PARE.

A construção de soluções consiste em, partindo de uma solução parcial vazia, um conjunto de M formigas artificiais constrói soluções a partir de elementos de um conjunto finito de componentes de solução disponíveis. O conjunto de componentes é composto por componentes que podem ser adicionados à solução parcial atual sem violar nenhuma das restrições. Em cada etapa de construção, a solução parcial é estendida com um novo componente.

Segundo Dorigo, a escolha de um componente da solução é guiada por um mecanismo estocástico, que é enviesado pelo feromônio associado a cada um dos componentes. A regra para a escolha estocástica dos componentes da solução varia entre os diferentes algoritmos ACO, mas, em todos eles, é inspirada no modelo de comportamento de formigas reais.

O próximo passo consiste em melhorar as soluções obtidas pelas formigas por meio de uma busca local. É considerado um passo opcional no algoritmo de ACO, mas que vem sendo bastante utilizado nas implementações de última geração.

O último passo consiste em atualizar os feromônios. O objetivo, segundo Dorigo, é aumentar os valores de feromônios associados a soluções boas ou promissoras e diminuir aqueles que estão associados a soluções piores. Nesse passo ocorre a diminuição de todos os valores de feromônios por meio da evaporação de feromônios e, em seguida, aumentando os níveis de feromônios associados a um conjunto escolhido de boas soluções.

Um dos primeiros trabalhos a utilizar um algoritmo de colônia de formiga para resolver o problema do horário foi o trabalho de Krzysztof Socha, Joshua Knowles, e Michael Sampels [14], em 2002. Apoiados pela Rede Metaheurística, um projeto da Comissão Europeia, os autores desenvolveram um algoritmo conhecido como MAX - MIN Ant System (MMAS). MMAS foi criado por T. Stutzle e H. H. Hoos [13].

No MMAS, somente a melhor formiga atualiza as trilhas. Para aliviar o problema relativo à estagnação inicial, são introduzidas intensidades nas trilhas máximas e mínimas, daí o nome sistema máximo mínimo. Um limite de intensidade máxima r_{max} é utilizado para a inicialização de todas as trilhas. Após cada iteração, a avaliação reduzirá a resistência das trilhas pelo fator p. Somente as melhores trilhas podem aumentar suas intensidades ou mantê-las em alto nível. Da mesma forma, são aumentadas ou mantidas as intensidades das piores trilhas. Para evitar que trilhas de baixa qualidade sejam atualizadas é estabelecido um limite de intensidade inferior r_{min}. O r_{min} é utilizado para a escolha da trilha de intensidade mínima.

No trabalho de Socha, as trilhas são representadas por grafos de construção, onde cada nó equivale a um timeslots (combinação de sala, dia e hora). As formigas se movem nos grafos de construção que correspondem a uma lista de eventos. Em cada iteração, cada formiga constrói uma horário completo. Ao caminhar nos grafos, as formigas vão atribuindo eventos em timeslots. A lista de eventos (ou grafos de construção) são ordenados de forma que os eventos mais difíceis sejam colocados no calendário primeiro, quando ainda há muitos timeslots livres. A escolha do timeslot t é realizado aleatoriamente com as probabilidades permitidas ao evento que dependem da matriz de feromônios.

A matriz de feromônios representam a posição absoluta onde os eventos devem ser colocados e está associado aos nós dos grafos. Os valores de feromônio são inicializados com um parâmetro r_{max} e, em seguida, atualizados por uma regra de atualização de feromônio global. No final de cada iteração cada irá gerar um solução candidata do horário. A melhor solução entre as soluções candidatas é escolhida. Se a solução for melhor do que a melhor solução global anterior, ela será substituída pela nova solução.

Por fim, são atualizados os valores de feromônio. A atualização é realizada usando a melhor solução global. Os valores dos feromônios correspondente à melhor solução global são aumentados. Cada nó do grafo da solução global com evento atribuído atualiza o feromônio do nó correspondente na matriz de feromônios. Em seguida todos os níveis de feromônios na matriz são reduzidos de acordo com o coeficiente de evaporação. Por fim, o valores de feromônio são ajustados para que todos fiquem dentro dos limites mínimo r_{min} definido. Todo o processo se repete, até que o tempo limite seja atingido.

Outro trabalho utilizando colônia de formigas foi apresentado, em 2012, por Clemens Nothegger e Gunther Raidl [10]. Os autores desenvolveram um algoritmo para resolver as instâncias do ITC 2007. A

principal característica do algoritmo de Nothegger e Raidl é o uso de duas matrizes simplificadas de feromônios distintas. As matrizes representam as relações evento-tempo e evento-sala. Tem como objetivo melhorar a convergência e fornecer flexibilidade para orientar efetivamente o processo de construção da solução. A construção da solução considera os eventos em uma ordem aleatória uniforme e atribui cada evento a uma sala viável e um intervalo de tempo viável de forma aleatória.

Nothegger e Raidl incluíram em seu algoritmo um procedimento de busca local. Foi chamado pelos autores de heurística de melhoria. A heurística de melhoria tenta mover os chamados eventos caros (ou seja, eventos que violam restrições flexíveis) para um intervalo de tempo diferente. Eventos caros são eventos que violam soft constraints. A troca só pode ocorrer se não violar as hard constraints.

Na solução cada formiga representa uma solução para o algoritmo. Ao final de cada iteração, cada formiga atualiza as matrizes de feromônios. Uma formula baseada no número de eventos não colocados e uma penalização por soft constraints violadas determinam a quantidade de feromônio proporcional à qualidade da solução para cada evento-tempo realizado e atribuições de evento-sala.

Os resultados obtidos indicaram uma instabilidade no algoritmo. Enquanto para algumas instâncias apresentou o melhor resultado da competição, para outras instâncias, obtiveram o pior resultado. Os autores concluem com a indicação de que o algoritmo foi ajustado para ser muito agressivo, favorecendo uma única solução boa em vez de soluções repetidamente boas. Resta ser examinado se o algoritmo também pode ser ajustado para mostrar menos variação na qualidade da solução.

Outros algoritmos são considerados de inteligência de enxame tais como, Fish Swarm Intelligent, que simula o comportamento de cardumes de peixes e Particle Swarm Optimization, que modela o *"comportamento social"* de espécies de pássaros.

Algoritmo Genético

Algoritmo Genético ou algoritmo evolutivo é um grupo de algoritmos baseados na seleção natural e genética natural das espécies da teoria de Darwin. Como na teoria de Darwin, os melhores indivíduos, que melhor se adaptam, sobrevivem.

O algoritmo consiste na criação de gerações de indivíduos (soluções para o problema). A partir de uma geração inicial, pequenas mutações genéticas são realizadas criando uma nova geração. Pressupõe que, o indivíduo que melhor se adaptar, no final é a solução para o problema.

Goes [5] detalha o passo a passo de um algoritmo genético:

1. Inicializa uma popula ção (A) através de uma heurística;
2. Avalia a população;
3. Seleciona indivíduo(s);
4. Mutação genética. Aplica operador genético, realizado através de uma heurística de melhoramento;
5. Avalia o resultado da nova população (B);
6. Critério de seleção. Se a população B for melhor que A, desconsidere a população A e a nova população passa a ser B. Caso contrário, desconsidera B e mantém a população A;

7. Critério de parada. O algoritmo deve parar? Em caso de resposta negativa volte ao passo 3, caso contrário esta é uma solução para o problema, PARE.

Quantas mais iterações (geração de novas populações), melhor será o resultado final do algoritmo genético. Porém, o melhoramento genético diminui a cada nova população geração. Nugraha [40] demonstrou isso em seu trabalho, no qual desenvolveu um algorítimo genético para resolver UCTP. Conforme a tabela 1.1, a evolução genética da população número 50 para a população número 100 foi de 90992 pontos. Enquanto que da população número 100 para população número 1000 foi de 1885 pontos.

Número Iteração (População)	Penalização
50	93502
100	2510
150	1515
200	1020
400	835
800	730
1000	625

Tabela 1.1: Tabela para demonstrar o melhoramento genético (Nugraha [40])

Em cada iteração é realizada uma operação genética. A operação genética é composta pelos seguintes passos: seleção, mutação, substituição geracional e pela medição de aptidão/violação.

A seleção orienta o algoritmo evolutivo para a solução ótima, preferindo cromossomos com baixas violações (penalidade). No algoritmo de Nugraha os cromossomos equivalem a um aula de uma determinada cadeira, em uma determinada sala em um determinado período. Após a seleção ocorre a mutação. A mutação é um operador de background que permite pesquisar cromossomos melhores. O objetivo é encontrar cromossomos possam melhorar a solução. Para isso são utilizados pontos de mutação que são selecionados conforme a probabilidade de mutação.

Uma vez executadas as operações de mutação, na substituição geracional o cromossomo que sofreu mutação torna-se inviável ou fora do espaço de busca. O próximo passo é a medição de aptidão / violação. O valor da medição é o resultado do número de violações multiplicado pela penalidade definida.

Um dos pontos chaves das soluções por algoritmo genético é a definição do critério de parada. Encontrar um bom critério de parada, pode determinar a qualidade final do trabalho. Nugraha estabelece a iteração 1000 como critério de parada. Obteve uma solução de qualidade, certamente melhor do que uma solução obtida de forma manual, sem auxilio computacional. Mas não pode garantir que é a solução ótima. Da mesma forma, não é possível saber se ao criar novas populações melhores resultados serão obtidos.

Outro trabalho que cabe destaque é o trabalho de Gozali [4]. Gozali utilizou um modelo de ilha para desenvolver um algoritmo genético para resolver UCTP.

O modelo de ilha consiste em criar uma ilha master e diversas ilhas slaves. As ilhas slave executam, independentemente das outras, o algoritmo genético desenvolvido. Em outras palavras, cada ilha slave faz o processo de criação e avaliação de populações e a mutação genética.

Em um determinado momento, as ilhas slaves trocam entre si os seus melhores indivíduos com o intuito de melhorar os resultados. Esse processo é controlado pela ilha master, que recebe os melhores

indivíduos das ilhas slaves e distribui para as outras ilhas slaves. A ilha master também é a responsável pelo resultado final, quando o critério de parada é atingido.

Gozali aplicou o seu modelo na Telkom University, na Indonésia e na Purdue University, nos Estados Unidos, conseguindo uma precisão de 99,74% e 96,80%, respectivamente (em comparação com outras soluções utilizando o método exato).

2. Alocação do horário universitário

Neste capítulo é apresentado em maior detalhes o problema da alocação do horário universitário e é apresentada uma solução genérica que serviu de núcleo para os casos de estudo que são apresentados nos capítulos 3 e 4.

A solução que será apresentada neste capítulo trata o problema de alocação do horário universitário como um problema de atribuição multidimensional. O objetivo final da solução é obter uma matriz de decisão multidimensional (com três ou mais dimensões), no qual aulas serão atribuídas a uma determinada cadeira, em uma determinada sala, em um determinado dia e em uma determinada hora. Para essa matriz serão atribuídos valores zero ou um para indicar se uma aula foi atribuída ou não (a matriz de decisão será detalhada em um tópico neste capítulo).

Foi adotado o método exato para resolver o problema. Uma solução de programação linear inteira foi desenvolvida utilizando a biblioteca MIP para Pythton [22] [23] e resolvida através do solver Gurobi (licença acadêmica), versão 9.1.1. O solver Gurobi utiliza o método Branch and Cut para resolver problemas de programação linear inteira. O código fonte completo está disponível no apêndice D.

Na seção seguinte é apresentada a caracterização genérica do problema de alocação do horário universitário e em seguida a solução completa adotada para resolver o problema.

2.1 O problema

Como já referenciado, elaborar um horário universitário consiste em alocar determinados recursos em um determinado dia e hora, seguindo restrições estabelecidas. Os recursos comuns a toda alocação do horário são: professores e salas de aulas.

Os professores são responsáveis por lecionar as aulas e tem atribuído para si uma ou mais cadeiras. Cada cadeira tem uma determinada carga horária que deve ser cumprida na sua totalidade. Ou seja, a solução deve se encarregar de atribuir quantas aulas necessárias para atingir a carga horária estabelecida. O não cumprimento inviabiliza a solução final.

Outra entidade que toda solução para elaborar o horário universitário deve abordar são os alunos. Os alunos podem ser tratados de duas abordagens. Na primeira abordagem a universidade estabelece antecipadamente a grade curricular das turmas e os alunos são matriculados nessas turmas, devendo cursar obrigatoriamente todas as cadeiras. A segunda abordagem é aquela na qual o aluno se matricula nas cadeiras disponíveis, estabelecendo uma grade curricular individual. A solução aqui apresentada leva em consideração a primeira abordagem: os alunos são matriculados em turmas e cursam a grade curricular da turma, não havendo grades curricular individuais.

2.1.1 Entidades

Para a solução genérica, foram mapeadas as seguintes entidades:

- **Turma**: entidade que representa as turmas. O aluno, ao ser matriculado, é inserido em uma determinada turma.
- **Cadeira**: entidade que representa as cadeiras que são lecionadas.
- **Professor**: entidade que representa os professores. Cada cadeira tem o seu professor estabelecido pela direção do curso. Um professor pode ministrar aulas de uma ou mais divisões de cadeiras.
- **Sala**: entidade que representa as salas nas quais são leccionadas aulas.
- **Dia de semana**: entidade que representa os dias de semana nos quais podem ser atribuídas uma aula.
- **Hora**: entidade que representa a hora de início da aula, com um espaçamento x de tempo. Corresponde aos intervalos de tempos para o qual uma aula pode ser atribuída. O tempo de uma hora pode variar conforme as regras de cada universidade e geralmente não equivale a 1 hora. Cada cadeira tem a sua carga horária (ou quantidade de horas).
- **Aula**: entidade que representa as aulas atribuídas. É a combinação das entidades cadeira, sala, dia de semana e hora.

A figura 2.1 demonstra o diagrama entidade relacionamento. No diagrama ficam evidenciadas os relacionamento entre as diversas entidades que compõem o problema.

As entidades listadas acima representam a solução final adotada. As entidades Dia de Semana e Hora em alguns estudos, entre eles o trabalho de Lach [20], são combinadas em uma só entidade que geralmente recebe o nome de Período. Neste trabalho, inicialmente foi proposto a criação de uma nova entidade chamada Slot, que em semelhança a entidade Período, agrupa as entidades Dia de Semana e Hora, mas ia além, incluindo também a entidade Sala. Slot então seria a combinação das entidades Sala, Dia de Semana e Hora. Dessa maneira, a entidade Aula era a combinação das entidades Cadeira e Slot.

A utilização da entidade Slot não traz nenhum beneficio evidente e muitas vezes dificulta a solução final. Um beneficio interessante seria se houvesse a redução do número de variáveis da entidade Aula, porém o número se mantém igual. Por outro lado, adotar a entidade Slot traz dificuldades na codificação da solução matemática do problema. Para adicionar restrições que obriguem a soma de uma das entidades envolvidas na entidade Slot, é necessário percorrer a entidade Slot para encontrar os slots e só então ir a entidade Aula. Com a solução final que foi adotada (sem a entidade Slot) para adicionar tal restrição basta percorrer a variável Aula.

2.1.2 Restrições (constraints)

Como referenciado anteriormente, em UCTP, restrições consistem em regras e políticas estabelecidas pelas universidades, bem como as preferências dos professores e alunos. Existem duas categorias de restrições: restrições rígidas (hard constraints) e restrições flexíveis (soft constraints).

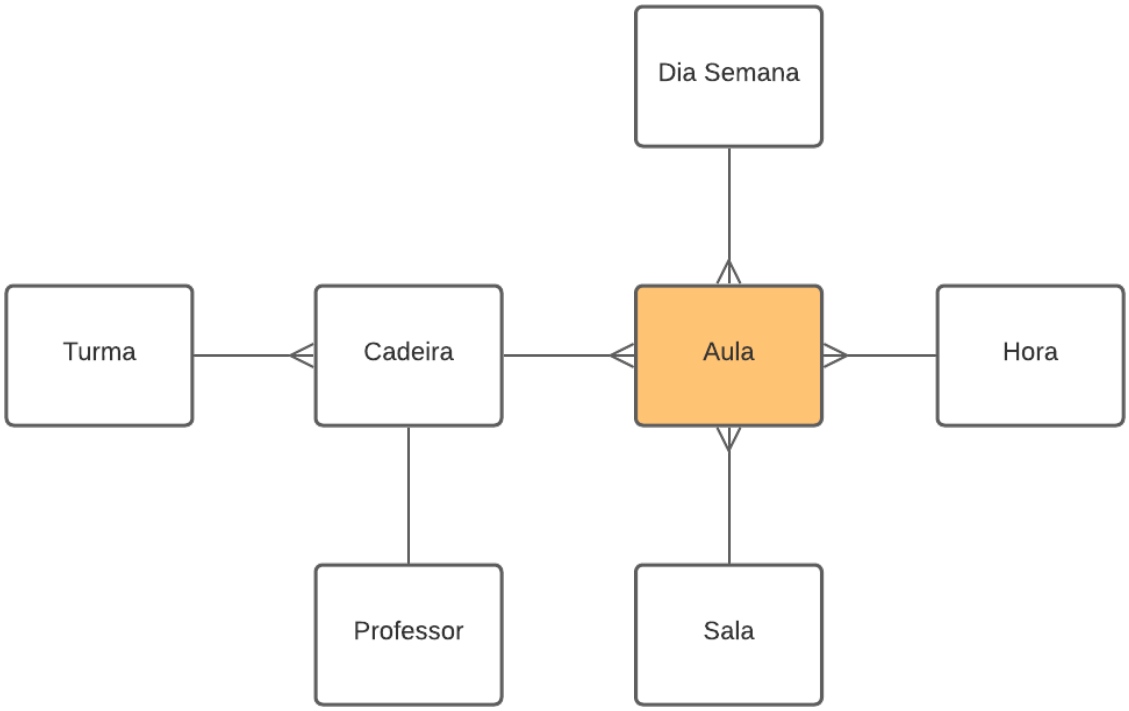

Figura 2.1: Diagrama entidade relacionamento da solução genérica

Na elaboração do horário universitário, as restrições rígidas são aquelas obrigatórias, que não podem ser violadas. Já as restrições flexíveis refletem situações desejáveis, mas que no limite podem não ser cumpridas. Para Azis [15] um horário universitário adequadamente bom é aquele com um resultado viável, mas uma qualidade superior é o que apresenta menos violações totais das restrições flexíveis. Isso significa que o processo de concepção do horário universitário deve tentar cumprir o máximo possível de restrições flexíveis para obter um resultado de qualidade superior. Segundo Azis, para casos específicos, as restrições rígidas também podem ser consideradas restrições flexíveis, a fim de encontrar uma solução viável

Pupeikienė [12] defende que as restrições (rígidas e flexíveis) devem ser classificadas usando pontos de penalidade. A qualidade do horário universitário é descrita pela soma total de todos os pontos de penalidade para todas as restrições. O melhor horário é um horário com a menor soma dos pontos de penalidade.

Em geral as restrições podem ser divididas em três classes (Santos [21]):

- organizacionais: relativos à instituição de ensino, nesse caso são incluídos os requerimentos que tratam da gestão de recursos, bem como do atendimento da legislação vigente.
- pedagógicos: pedidos importantes para o bom aproveitamento das aulas, como duração das aulas, intervalos, programação de exames.
- pessoais: de acordo com as preferências e necessidades pessoais dos professores e dos alunos

Conflitos

Conflitos são restrições que sempre se aplicam em qualquer problema de planejamento. Pelo fato de sempre estarem presentes nos problemas de planejamento, Kristiansen [18], os chamou de conflitos de primeira ordem. Por exemplo, nenhuma pessoa pode estar em mais de um lugar ao mesmo tempo.

Não é diferente para o problema do horário universitário, que também possui os seus conflitos de primeira ordem. Os conflitos não podem existir na solução final. São tratados no problema do horário universitário como hard constraints, ou seja, constraints que não podem ser violadas.

Foram identificados os seguintes conflitos:

- **Conflito das salas**: o conflito das salas tem como objetivo evitar que uma sala tenha duas ou mais aulas atribuídas no mesmo dia e hora.
- **Conflito das turmas**: o conflito das turmas tem como objetivo evitar que uma turma tenha duas ou mais aulas atribuídas no mesmo dia e hora.
- **Conflito dos professores**: um professor não pode lecionar duas ou mais aulas ao mesmo tempo. Portanto, as aulas atribuídas para as cadeiras de cada professor devem ser em dias e horas diferentes.

Restrições utilizadas no problema

A solução genérica da alocação do horário universitário procura resolver os problemas de conflitos referenciados no item anterior e também garantir o cumprimento da carga horária estabelecida para cada cadeira. Portanto, as restrições incluídas na solução genérica são somente hard constraints, ou seja, restrições que devem ser cumpridas obrigatoriamente.

- Restrições rígidas (hard constraints)
  - HC01 Garantir o cumprimento da carga horária das cadeiras.
  - HC02 Garantir apenas uma aula seja atribuída para uma determinada sala, dia da semana e hora (conflito das salas).
  - HC03 Garantir que cada professor tenha apenas uma aula atribuída para um determinado dia de semana e hora (conflito dos professores).
  - HC04 Garantir que cada turma tenha apenas uma aula atribuída para um determinado dia de semana e hora (conflito das turmas).

2.2 Solução

2.2.1 Conjuntos

Os conjuntos são as representações das entidades e seus relacionamentos. São também os parâmetros de entrada para o algoritmo. Na solução genérica, os conjuntos são os seguintes:

- **C - Cadeiras**: relação das cadeiras

$$c \in C = \{\text{Cadeira 1, Cadeira 2, ... C}\}$$

- **S - Sala de Aula**: relação das salas de aulas

$$s \in S = \{\text{Sala 1, Sala 2, ... S}\}$$

- **D - Dias de Semana**: relação dos dias de semana

$$d \in D = \{\text{Segunda, Terça, Quarta, Quinta, Sexta}\}$$

- **H - Hora**: relação das horas

$$h \in H = \{\text{Hora 1, Hora 2, ... H}\}$$

- **T - Turmas**: relação das turmas

$$t \in T = \{\text{Turma 1, Turma 2, ... T}\}$$

- **CT - Cadeiras da turma**: cada item do conjunto CT é um conjunto composto pelas cadeiras de cada turma do conjunto T, seguindo a mesma ordenação. O primeiro item do conjunto CT corresponde as cadeiras da primeira turma do conjunto T.

$$ct \in CT = \{\{\text{Turma 1: Cadeira 1, Cadeira 2, ...}\}, ..., \{\text{Turma T: Cadeira n, Cadeira n1, ...}\}\ \}$$

- **P - Professores**: relações dos professores

$$p \in P = \{\text{Professor 1, Professor 2, ... P}\}$$

- **CP - Cadeiras do professor**: cada item do conjunto CP é um conjunto composto pelas cadeiras de cada professor do conjunto P, seguindo a mesma ordenação. O primeiro item do conjunto CP corresponde as cadeiras do primeiro professor do conjunto P.

$$cp \in CP = \{\{\text{Professor 1: Cadeira 1, Cadeira 2, ...}\}, ..., \{\text{Professor P: Cadeira n, Cadeira n1, ...}\}\ \}$$

- **CH - Carga horária da cadeira**: cada item do conjunto CH contém a carga horária de cada cadeira do conjunto C, segundo a mesma ordenação.

$$ch \in CH = \{3, 3, ...CH\}$$

2.2.2 Matriz de decisão multidimensional

Na solução genérica, a matriz de decisão multidimensional é uma estrutura lógica binária fruto da combinação entre os conjuntos de cadeiras, salas, dias de semana e hora. Em um problema de programação linear a matriz de decisão é representada por variáveis, as quais se dá o nome de variáveis de decisão.

A matriz de decisão multidimensional é, em termos matemáticos, um tensor de atribuição. No tensor de atribuição cada elemento (cadeiras, salas, dias de semana e hora) da matriz de decisão é chamado de dimensão. A utilização de um tensor de atribuição permite a soma sobre uma ou mais dimensões com o objetivo de atender as hard constraints ou obter informações úteis sobre o horário gerado. Considerando que o tensor é formado variáveis binárias, resultado da soma será o total de aulas atribuídas para o critério utilizado na soma.

Por exemplo, a HC01, que tem como objetivo garantir o cumprimento da carga horária das cadeiras, é garantida com a soma no tensor das dimensões sala, dia de semana e hora para cada cadeira. Já a HC02, que tem como objetivo garantir que apenas uma aula seja atribuída para uma determinada sala, dia de semana e hora é garantida com a soma da dimensão cadeira para cada sala, dia de semana e hora.

Somar sobre as dimensões do tensor permite também obter informações úteis. Por exemplo, a taxa de ocupação das salas é obtida ao somar sobre a dimensão sala. Os horários de picos (com mais aulas atribuídas) obtém-se ao somar sobre as dimensões dia de semana e hora. Essas informações e outras auxiliam os gestores das universidades a tomada de decisões, principalmente voltada a questões recursos humanos necessários para o apoio as aulas, tais como equipa de administração e de limpeza.

São criadas variáveis de decisão para cada elemento do tensor. Dessa maneira, uma variável de decisão representa uma possível aula para uma determinada cadeira, em uma determinada sala, em um determinado dia e hora. Se no resultado final essa variável tiver valor um, significa que foi atribuída aula para aquela cadeira, na sala, dia e hora que a variável representa. Ao contrário, valor zero para a variável, indica que não foi atribuída a aula.

As variáveis de decisão são representadas matematicamente da seguinte forma:

$$X_{c,s,d,h}\{0, 1\} \quad \forall c \in C, \forall s \in S, \forall d \in D, \forall h \in H \tag{2.1}$$

A figura 2.3 demonstra como é composta a matriz de decisão multidimensional X. Conforme pode ser observado na figura, a matriz é composta por quatro matrizes. Na primeira matriz as linhas são representam as cadeiras. Já na segunda matriz, as linhas representam as salas. A terceira matriz, as linhas representam os dias da semana e por último, a quarta matriz representam as horas. Em termos computacionais, a matriz de decisão multidimensional é transformada em um vetor, conforme demonstrado na figura 2.2.

$c_1 s_1 d_1 h_1$	...	$c_1 s_1 d_1 h_n$	...	$c_1 s_1 d_n h_1$	...	$c_1 s_1 d_n h_n$	...
$c_1 s_n d_1 h_1$	...	$c_1 s_n d_1 h_n$	...	$c_1 s_n d_n h_1$	...	$c_1 s_n d_n h_n$	...
$c_n s_1 d_1 h_1$	...	$c_n s_1 d_1 h_n$	...	$c_n s_1 d_n h_1$	...	$c_n s_1 d_n h_n$	...
$c_n s_n d_1 h_1$	...	$c_n s_n d_1 h_n$	...	$c_n s_n d_n h_1$	...	$c_n s_n d_n h_n$	

Figura 2.2: Representação da transformação da matriz de decisão multidimensional X em um vetor

2.2.3 Custo de atribuição

Custo de atribuição (CA) é uma matriz multidimensional que indica o custo em atribuir uma aula a uma determinada cadeira em uma determinada sala, dia e hora. Como a solução é um problema de minimização, quanto menor o custo de atribuição, melhor é a combinação cadeira, sala, dia e hora. Cada elemento da matriz de decisão multidimensional X possui o seu custo de atribuição correspondente.

As regras para definição do custo de atribuição são diferentes para cada universidade. A definição do custo de atribuição apropriado é um ponto chave para um resultado de qualidade, pois é o custo de atribuição que irá determinar se atribuir aula a uma cadeira em uma determinada sala, dia e hora é mais vantajoso que atribuir aula para um outra sala, dia e hora.

A lista a seguir são exemplos como se pode definir o custo de atribuição. A lista não pretende ser definitiva e existem diversas outras formas de definição do custo de atribuição.

- Privilegiar uma sala em detrimento de outra para uma determinada cadeira, atribuindo pesos maiores para aquelas salas que não se pretende que a cadeira tenham aulas atribuídas.
- Evitar horas iniciais e finais do dia, atribuindo pesos maiores para as horas que estão compreendidas nesses horários.
- Evitar atribuir aulas em dias e horas que um determinado professor não esteja disponível, estabelecendo pesos maiores para esses dias e horas. O mesmo vale para os dias e horas que uma sala não esteja disponível.
- Estabelecer períodos em que uma turma irá ter aula. Pode-se querer que as aulas de uma determinada turma só ocorram no período da manhã. Nesse caso são estabelecidos pesos maiores para as horas que não são do período da manhã.

O custo de atribuição (CA) é representado matematicamente da seguinte forma:

$$CA_{c,s,d,h} \quad \forall c \in C, \forall s \in S, \forall d \in D, \forall h \in H \tag{2.2}$$

2.2.4 Função Objetivo

Na solução genérica, a função objetivo é a minimização do somatório de todas as variáveis de decisão X multiplicadas pelo seu custo de atribuição CA. Visa encontrar um horário que privilegie o critério definido pelo custo de atribuição.

A função objetivo é representada matematicamente da seguinte forma:

$$min : \sum_{c \in C} \sum_{s \in S} \sum_{d \in D} \sum_{h \in H} CA_{c,s,d,h} \cdot X_{c,s,d,h} \tag{2.3}$$

2.2.5 Hard Constraints

A seguir são apresentadas a formulação para programação linear das hards constraints. Na solução genérica as hards constraints são somentes aquelas constraints chamada de de primeira ordem. Constraints de primeira ordem são comuns em todos os problemas de horário, incluindo os dois estudos de casos apresentados nesse trabalho.

- **HC01** A hard constraint **HC01** tem como objetivo garantir o cumprimento da carga horária das cadeiras.

  Para isto, o somatório de aulas atribuídas para uma determinada cadeira tem que ser igual a carga horária estabelecida.

$$\sum_{s \in S} \sum_{d \in D} \sum_{h \in H} X_{c,s,d,h} = CH_c \quad \forall c \in C \tag{2.4}$$

- **HC02** A hard constraint **HC02** tem como objetivo garantir que apenas uma aula seja atribuída para uma determinada sala, dia da semana e hora.

  Para isto, o somatório das aulas atribuídas para uma determinada sala em um determinado dia da semana e em uma determinada hora deve ser igual ou menor que 1.

$$\sum_{c \in C} X_{c,s,d,h} \leq 1 \quad \forall s \in S, \forall d \in D, \forall h \in H \tag{2.5}$$

- **HC03** A hard constraint **HC03** tem como objetivo garantir que cada professor tenha apenas uma aula atribuída para um determinado dia de semana e hora.

  Para isto, o somatório de aulas atribuídas para um determinado professor em um determinado dia da semana e em uma determinada hora tem que ser igual ou menor que 1.

$$\sum_{c \in CP_p} \sum_{s \in S} X_{c,s,d,h} \leq 1 \quad \forall p \in P, \forall d \in D, \forall h \in H \tag{2.6}$$

- **HC04** A hard constraint **HC04** tem como objetivo garantir que cada turma tenha apenas uma aula atribuída para um determinado dia de semana e hora.

  Para isto, o somatório de aulas atribuídas para as divisões de cadeiras de uma determinada turma em um determinado dia da semana e em uma determinada hora tem que ser igual ou menor que 1.

$$\sum_{c \in CT_t} \sum_{s \in S} X_{c,s,d,h} \leq 1 \quad \forall t \in T, \forall d \in D, \forall h \in H \tag{2.7}$$

Casos de estudo

Nos dois próximos capítulos são apresentados dois casos de estudos nos quais a solução genérica do problema de alocação do horário universitário foi utilizada como núcleo da solução final sendo a mesma adaptada as necessidades de cada problema.

O primeiro caso de estudo foi desenvolvido uma solução para os cursos de licenciatura do Departamento de Engenharia Informática e Sistemas de Informação da Universidade Lusófona (DEISI), em Lisboa.

O segundo caso de estudo foi desenvolvido uma solução para resolver o problema proposto pelo ITC 2007. O ITC 2007 foi um concurso internacional do problema de alocação do horário universitário. Desenvolver uma solução para o ITC 2007 permitiu realizar comparação dos resultados da nossa solução com outras soluções já desenvolvidas pelo mundo acadêmico.

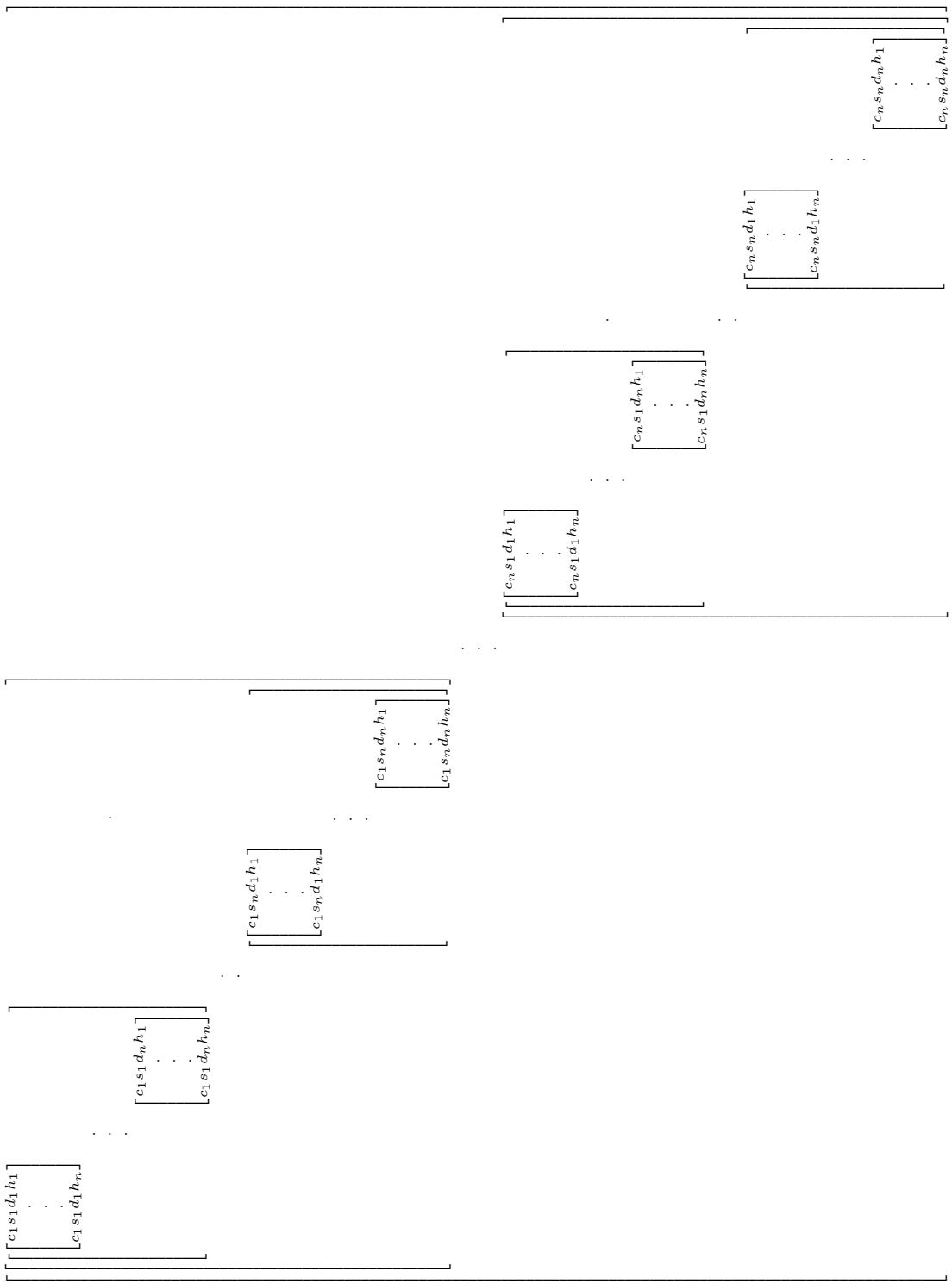

Figura 2.3: Representação da matriz de decisão multidimensional

3. Caso de estudo 1: UHLT

3.1 O problema

A solução aqui apresentada, procura resolver a calendarização dos cursos de licenciatura do Departamento de Engenharia Informática e Sistemas de Informação da Universidade Lusófona de Humanidades e Tecnologia (UHLT), situada em Lisboa, Portugal.

Os cursos são divididos em turmas semestrais. Cada turma possui em sua grade curricular um número determinado de cadeiras. Algumas cadeiras podem ser frequentadas por alunos de cursos/turmas diferentes. Um aluno, ao fazer a matrícula, é incluído em uma turma e deve obrigatoriamente frequentar as cadeiras atribuídas a sua turma.

As cadeiras geralmente são divididas em aulas teóricas, laboratoriais, práticas e teóricas-práticas. Cada uma dessas divisões possui a sua própria carga horária semanal e todas as horas dessa divisão de cadeira devem ser lecionadas de forma consecutiva em somente um dia da semana. Duas ou mais divisões de uma mesma cadeira não podem ser lecionadas no mesmo dia da semana.

A direção do curso pode definir dias de folgas para as turmas e quais as horas de preferência para cada divisão de cadeira. Como exemplo, em um curso de três anos, os alunos do primeiro e segundo ano tinham aulas no período diurno e em quatro dias da semana. Já os alunos do terceiro ano tinham aulas todos os dias e as aulas eram obrigatoriamente no período noturno.

Cada divisão de cadeira tem um professor atribuído, o qual é definido no início de cada período letivo pela direção do curso. Além disso, algumas divisões de cadeira podem possuir salas de aulas obrigatórias. Isso acontece geralmente para as aulas práticas, que necessitam que as aulas sejam lecionadas em um laboratório, por exemplo.

As horas são divididos em slots de 30 minutos de segunda a sexta, sendo que a primeira hora do dia disponível inicia as 08:00 h e a última as 23:30 h. Para as turmas que não possuem horas obrigatórias e sim preferência por determinadas horas, foram estabelecidos pesos para cada hora. Por exemplo, para as turmas do primeiro e segundo ano as aulas devem acontecer no período diurno, entre as 08:00 h e as 18:00 h. Dessa forma, as horas entre 08:00 h e 18:00 h possuem pesos menores do que as horas após as 18:00 h.

Há preferência por manchas de horários. É preferível que todas as aulas de uma determinada turma em um determinado dia ocorram em sequência.

Entidades

Uma vez definido o problema a ser resolvido o primeiro passo foi estabelecer as entidades e desenhar um diagrama dos seus relacionamentos. Em relação a solução básica, a solução desenhada para atender a Universidade Lusófona inclui a entidade divisão de cadeira.

As entidades identificadas são as seguintes:

- **Turma**: entidade que representa as turmas, que juntas, formam os cursos oferecidos pela universidade. O aluno, ao ser matriculado, é inserido em uma determinada turma.
- **Cadeira**: entidade que representa as cadeiras que são ministradas.
- **Divisão de Cadeira**: entidade que representa as divisões de uma cadeira, que juntas, formam a grade curricular das turmas. As cadeiras podem ser divididas em aulas teóricas, laboratoriais, práticas e teóricas-práticas.
- **Professor**: entidade que representa os professores. Cada divisão de cadeira tem o seu professor estabelecido pela direção do curso. Um professor pode ministrar aulas de uma ou mais divisões de cadeiras.
- **Sala**: entidade que representa as salas de aulas às quais uma aula pode ser atribuída. Algumas divisões de cadeiras só podem ser ministradas em determinadas salas.
- **Dia de semana**: entidade que representa os dias de semana aos quais podem ser atribuídas aulas.
- **Hora**: entidade que representa os intervalos de horas para o qual uma aula pode ser atribuída. As horas são divididas em intervalos de 30 minutos. Ex.: 08:00 - 08:30, 08:30 - 09:00, etc. Cada divisão de cadeira tem a sua carga horária (ou quantidade de horas).
- **Aula**: entidade que representa as aulas atribuídas. É a combinação das entidades divisão de cadeira, sala, dia de semana e hora.

A figura 3.1 demonstra o diagrama entidade relacionamento. No diagrama ficam evidenciadas os relacionamentos entre as diversas entidades que compõem o problema.

Lista das restrições utilizadas no problema

- Restrições rígidas (hard constraints)
  - HC01 Garantir o cumprimento da carga horária das divisões de cadeiras.
  - HC02 Garantir aulas de duas ou mais divisões de uma determinada cadeira não sejam atribuídas para um mesmo dia da semana
  - HC03 Garantir apenas uma aula seja atribuída para uma determinada sala, dia da semana e hora.
  - HC04 Garantir que cada professor tenha apenas uma aula atribuída para um determinado dia de semana e hora.

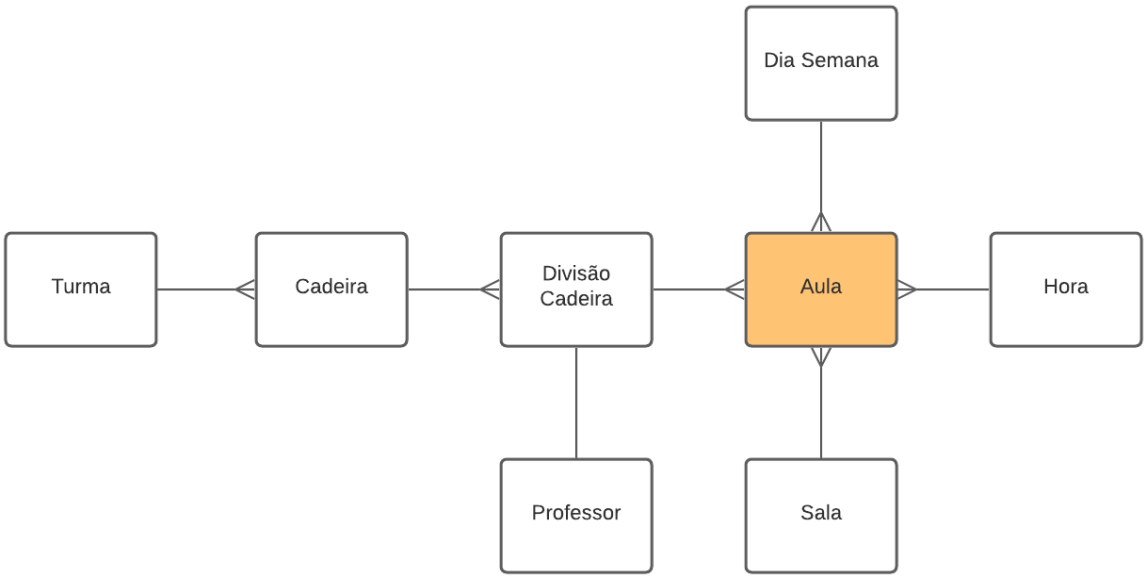

Figura 3.1: Diagrama entidade relacionamento

  - HC05 Garantir que cada turma tenha apenas uma aula atribuída para um determinado dia de semana e hora.
  - HC06 Garantir a atribuição das salas de aulas obrigatórias para uma determinada divisão de cadeira.
  - HC07 Garantir o cumprimento dos dias de folga das turmas
  - HC08 Garantir que as divisões de cadeira não tenham aulas atribuídas para as horas não permitidas
  - HC09 Garantir que todas as aulas de uma divisão de cadeira sejam consecutivas

- Restrições flexíveis (soft constraints)
  - SC01 Horário compacto de cada turma.

3.2 Solução

Conjuntos

- **C - Cadeiras**: relação das cadeiras

$$c \in C = \{\text{Cadeira 1, Cadeira 2, ... C}\}$$

- **DC - Divisões das Cadeiras**: cada item do conjunto DC é um conjunto composto pelas divisões de cada cadeira do conjunto C, seguindo a mesma ordenação. O primeiro item do conjunto DC corresponde as divisões da primeira cadeira do conjunto C.

$$dc \in DC = \{\{\text{Cadeira 1 Divisão 1, Cadeira 1 Divisão 2, ...}\}, ... \{\text{DC}\}\}$$

- **S - Sala de Aula**: relação das salas de aulas

$$s \in S = \{\text{Sala 1, Sala 2, ... S}\}$$

- **D - Dias de Semana**: relação dos dias de semana

$$d \in D = \{\text{Segunda, Terça, Quarta, Quinta, Sexta}\}$$

- **H - Horas**: relação de horas

$$h \in H = \{\text{08:00 - 08:30, 08:30 - 09:00, , ..., 23:30 - 00:00}\}$$

- **T - Turmas**: relação das turmas

$$t \in T = \{\text{Turma 1, Turma 2, ... T}\}$$

- **DCT - Divisões de cadeiras da turma**: cada item do conjunto DCT é um conjunto composto pelas divisões de cadeiras de cada turma do conjunto T, seguindo a mesma ordenação. O primeiro item do conjunto DCT corresponde as divisões das cadeiras da primeira turma do conjunto T.

$$dct \in DCT = \{\{\text{Turma 1: Cadeira 1 Divisão 1, Cadeira 2 Divisão 2, ...}\}, ...,$$

$$\{\text{Turma T: Cadeira n Divisão n, Cadeira n1 Divisão n1, ...}\}\ \}$$

- **P - Professores**: relações dos professores

$$p \in P = \{\text{Professor 1, Professor 2, ... P}\}$$

- **DCP - Divisões de cadeiras do professor**: cada item do conjunto DCP é um conjunto composto pelas divisões de cadeiras de cada professor do conjunto P, seguindo a mesma ordenação. O primeiro item do conjunto DCP corresponde as divisões das cadeiras do primeiro professor do conjunto P.

$$dcp \in DCP = \{\{\text{Professor 1: Cadeira 1 Divisão 1, Cadeira 2 Divisão 2, ...}\}, ...,$$

$$\{\text{Professor P: Cadeira n Divisão n, Cadeira n1 Divisão n1, ...}\}\ \}$$

- **DCS - Salas de aulas obrigatórias da divisão cadeira**: cada item do conjunto DCS é composto uma divisão de cadeira do conjunto DC e por um conjunto de salas de aulas obrigatórias para a divisão de cadeira. Se uma divisão de cadeira não tiver salas obrigatórias, a mesma não fará parte desse conjunto.

$$dcs \in DCS = \{\{\text{Cadeira 1 Divisão 1, } \{\text{Sala 1, Sala 2, ...}\}\}, ... \{\text{dc, } \{\text{s}\}\}\}$$

- **DCH - Horas não permitidas da divisão cadeira**: cada item do conjunto DCH é composto uma divisão de cadeira do conjunto DC e por um conjunto de horas não permitidas para a divisão de cadeira. Se uma divisão de cadeira não tiver horas não permitidas, a mesma não fará parte desse conjunto.

  O conjunto DCH tem duas funções principais. Primeiro, permitir à direção estabelecer os intervalos de horas de funcionamento de um determinado curso ou divisão de cadeira. A outra função desse conjunto é atender restrições de horários de um determinado professor, que por outros compromissos não está disponível em determinados horários.

$$dch \in DCH = \{\{\text{Cadeira 1 Divisão 1, } \{\text{18:00 - 18:30, 18:30 - 19:00, ...}\}\}, ... \{\text{dc, } \{\text{h}\}\}\}$$

- **TD - Dias da semana das turmas**: cada item do conjunto TD é um conjunto composto pelos dias de aula de cada turma do conjunto T, seguindo a mesma ordenação. O primeiro item do conjunto TD corresponde aos dias de aula da primeira turma do conjunto T.

  A principal função do conjunto TD é permitir a direção do curso estabelecer os dias de aula de uma determinada turma.

$$td \in TD = \{\{\text{Segunda, Terça, Quarta, Sexta}\}, ... \{\text{TD}\}\}$$

- **CH - Carga horária da divisão da cadeira**: cada item do conjunto CH é composto uma divisão de cadeira do conjunto DC e a sua carga horária.

$$ch \in CH = \{\{\text{Cadeira 1 Divisão 1, 3}\}, ... \{\text{dc, CH}\}\} \quad \forall dc \in DC$$

Parâmetros

- **PH(c, h)**: Peso de um determinada hora h para uma determinada cadeira c. O peso PH é determinado pela direção do curso e o principal objetivo é evitar que as primeiras e últimas horas tenham aulas atribuídas. Os pesos utilizados no problema estão demonstrados n tabela 3.1. O parâmetro PH compõem a função objetivo.

Turmas com aula entre 8h e 18h	
Hora	Peso
08:00 - 08:30	10
08:30 - 09:00	1
...	
16:30 - 17:00	1
17:00 - 17:30	1
17:30 - 18:00	1

Turmas com aula entre 18h e 23:59h	
Hora	Peso
18:00 - 18:30	1
18:30 - 19:00	1
...	
22:30 - 23:00	1
23:00 - 23:30	10
23:30 - 23:59	10

Tabela 3.1: Tabela para demonstrar peso turma x hora

- **UHD**: Parâmetro para indicar a última hora do dia.

Variáveis de decisão

As variáveis de decisão formam uma matriz binária multidimensional de aulas, fruto da combinação entre cadeiras, divisão de cadeiras, salas, dias de semanas e horas.

$$X_{c,dc,s,d,h}\{0,1\} \quad \forall c \in C, \forall dc \in DC, \forall s \in S, \forall d \in D, \forall h \in H \tag{3.1}$$

$X_{c,dc,s,d,h}$ receberá 1 caso tenha uma aula atribuída e 0 caso contrário.

Variáveis Auxiliares

- $Z_{c,dc,d}$ = Variável binária que indica se uma divisão de cadeira tem aula atribuída em um determinado dia.
- $P_{t,d}$ = Variável inteira que indica a primeira hora com aula atribuída em um determinado dia da semana para uma determinada turma.
- $U_{t,d}$ = Variável inteira que indica a última hora com aula atribuída em um determinado dia da semana para uma determinada turma.
- $T_{t,d}$: Variável inteira que indica o total de aulas atribuídas em um determinado dia da semana para uma determinada turma.
- $V_{t,d}$ = Variável inteira que indica o total de vazios em um determinado dia da semana para uma determinada turma.

Restrições

- **HC01** A hard constraint **HC01** tem como objetivo garantir o cumprimento da carga horária das divisões de cadeiras.

  Para isto, a somatória de aulas atribuídas para uma determinada divisão de cadeira tem que ser igual a carga horária estabelecida.

$$\sum_{s \in S} \sum_{d \in D} \sum_{h \in H} X_{c,dc,s,d,h} = CH_{dc} \quad \forall c \in C, \forall dc \in DC \tag{3.2}$$

- **HC02** A hard constraint **HC02** tem como objetivo garantir que aulas de duas ou mais divisões de uma determinada cadeira não sejam atribuídas para um mesmo dia da semana.

  Para isto, foi necessário criar a variável binária auxiliar $Z_{c,dc,d}$ que irá receber 1 caso seja atribuído alguma aula a divisão de cadeira dc da cadeira c no dia da semana d e 0 caso contrário.

$$X_{c,dc,s,d,h} - Z_{c,dc,d} \leq 0 \quad \forall c \in C, \forall dc \in DC, \forall s \in S, \forall d \in D, \forall h \in H \tag{3.3}$$

  E para garantir que não seja atribuídas aulas no mesmo dia da semana para duas ou mais divisões de cadeiras para um determinada cadeira, a somatória das variáveis $Z_{c,dc,d}$ para a cadeira c no dia da semana d tem que ser menor ou igual a 1.

$$\sum_{dc \in DC_c} Z_{c,dc,d} \leq 1 \quad \forall c \in C, \forall d \in D \tag{3.4}$$

- **HC03** A hard constraint **HC03** tem como objetivo garantir apenas uma aula seja atribuída para uma determinada sala, dia da semana e hora.

  Para isto, a somatória das aulas atribuídas para uma determinada sala em um determinado dia da semana e em uma determinada hora deve ser igual ou menor que 1.

$$\sum_{c \in C} \sum_{dc \in DC_c} X_{c,dc,s,d,h} \leq 1 \quad \forall s \in S, \forall d \in D, \forall h \in H \tag{3.5}$$

- **HC04** A hard constraint **HC04** tem como objetivo garantir que cada professor tenha apenas uma aula atribuída para um determinado dia de semana e hora.

Para isto, a somatória de aulas atribuídas para um determinado professor em um determinado dia da semana e em uma determinada hora tem que ser igual ou menor que 1.

$$\sum_{c,dc \in DCP_p} \sum_{s \in S} X_{c,dc,s,d,h} \leq 1 \quad \forall p \in P, \forall d \in D, \forall h \in H \tag{3.6}$$

- **HC05** A hard constraint **HC05** tem como objetivo garantir que cada turma tenha apenas uma aula atribuída para um determinado dia de semana e hora.

  Para isto, a somatória de aulas atribuídas para as divisões de cadeiras de uma determinada turma em um determinado dia da semana e em uma determinada hora tem que ser igual ou menor que 1.

$$\sum_{c,dc \in DCT_t} \sum_{s \in S} X_{c,dc,s,d,h} \leq 1 \quad \forall t \in T, \forall d \in D, \forall h \in H \tag{3.7}$$

- **HC06** A hard constraint **HC06** tem como objetivo garantir a atribuição das salas de aulas obrigatórias para uma determinada divisão de cadeira.

  Para isto, se uma determinada divisão de cadeira tiver uma ou mais salas obrigatórias, a somatória todas as aulas das demais sala que não estão na lista de salas obrigatórias desta divisão de cadeira devem ser igual a 0.

$$\sum_{s \in S, s \notin DCS_{dc}} \sum_{d \in D} \sum_{h \in H} X_{c,dc,s,d,h} = 0 \quad \forall c, dc \in DCS \tag{3.8}$$

- **HC07** A hard constraint **HC07** tem como objetivo garantir o cumprimento dos dias de folga das turmas.

  Para isto, a somatória das aulas das divisões de cadeiras de uma determinada turma naqueles dias que a turma não tiver aula deve ser igual a 0.

$$\sum_{c,dc \in DCT_t} \sum_{d \in D, d \notin TD_t} \sum_{h \in H} X_{c,dc,s,d,h} = 0 \quad \forall t \in T \tag{3.9}$$

- **HC08** A hard constraint **HC08** tem como objetivo garantir que as divisões de cadeira não tenham aulas atribuídas para as horas não permitidas.

Para isto, a somatória das aulas das horas não permitidas para uma determinada divisão de cadeira deve ser igual a 0.

$$\sum_{s \in S} \sum_{d \in D} \sum_{h \in DCH_{dc}} X_{c,dc,s,d,h} = 0 \quad \forall c, dc \in DCH \tag{3.10}$$

- **HC09** A hard constraint **HC09** tem como objetivo garantir que todas as aulas de uma determinada divisão de cadeira sejam consecutivas

  Foi utilizado uma adaptação que Lach [20] utilizou em seu trabalho para evitar aulas isoladas. Diferente da restrição HC09 que é uma hard constraint, no trabalho de Lach evitar aulas isoladas é uma soft constraint. O objetivo foi encontrar o total de aulas isoladas um determinado currículo e penalizar na função objetivo. Portanto, foi incluído nas equações uma variável auxiliar que apontava as aulas isoladas. Essa variável foi omitida na solução para a Universidade Lusófona, pois aqui deseja-se que todas as aulas de uma determinada divisão de cadeira sejam consecutivas.

  São necessários 3 conjuntos de equações para resolver a hard constraint.

  - Para a primeira hora do dia:

$$X_{c,dc,s,d,h} - X_{c,dc,s,d,h+j} \leq 0 \quad \forall c \in C, \forall dc \in DC, \forall s \in S, \forall d \in D, h = 1, \forall j \geq 1, j < CH_{dc} \tag{3.11}$$

    A equação (3.11) irá garantir que ao atribuir aula na primeira hora do dia ($h = 1$), as demais horas, até o limite da carga horária da divisão de cadeira ($h = CH_{dc}$), também terão aulas atribuídas. Isto é garantido pelo fato de que serão criadas tantas equações quantas necessárias (limitadas a carga horária de cada divisão de cadeira). O primeiro elemento da equação, $X_{c,dc,s,d,h}$, sempre será $h = 1$ e o segundo elemento, $X_{c,dc,s,d,h+j}$, será $h = 2$ na primeira equação, $h = 3$ na segunda equação e, assim por diante, até $h = CH_{dc}$ na última equação. Este conjunto de equações se repete para cada grupo de cadeira, divisão de cadeira, sala e dia da semana.

  - Para a última hora do dia:

$$X_{c,dc,s,d,UHD} - X_{c,dc,s,d,UHD-j} \leq 0 \quad \forall c \in C, \forall dc \in DC, \forall s \in S, \forall d \in D, j \geq 1, j < CH_{dc} \tag{3.12}$$

    A equação (3.12) é semelhante a equação (3.11), porém a base é a última hora do dia ($h = UHD$). Essa equação irá garantir que ao atribuir aula na última hora do dia, as horas imediatamente anteriores também terão aula atribuídas. Da mesma forma que a equação (3.11), serão criadas tantas equações quantas necessárias. O primeiro elemento da equação será

sempre $X_{c,dc,s,d,UHD}$ e o segundo elemento, $X_{c,dc,s,d,UHD-j}$, será $UHD-1$ na primeira equação, será $UHD-2$ na segunda equação e, assim por diante, até $UHD-(CH_{dc}-1)$ na última equação. Este conjunto de equações também se repete para cada grupo de cadeira, divisão de cadeira, sala e dia da semana.

– Para as demais horas do dia

$$\begin{aligned} -X_{c,dc,s,d,h} + X_{c,dc,s,d,h+i} - X_{c,dc,s,d,h+j} \leq 0 \\ \forall c \in C, \forall dc \in DC, \forall s \in S, \forall d \in D, \\ \forall h \in H, h > 1, h < UHD - 1, \\ \forall i \geq 1, i < CH_{dc}, h + i < UHD, \\ \forall j \geq 2, j \leq CH_{dc}, j > i, h + j \leq UHD \end{aligned} \tag{3.13}$$

A equação (3.13) irá garantir que ao atribuir aula para uma determinada hora as demais horas posteriores também terão aulas atribuídas, até o limite da carga horária da divisão de cadeira. Considerando que o primeiro elemento, $X_{c,dc,s,d,h}$ é o elemento base e para cada elemento base das equações, será criado um conjunto de equações da seguinte forma:

1. Na primeira equação o segundo elemento, $X_{c,dc,s,d,h+i}$, terá i igual a 1 e o terceiro elemento, $X_{c,dc,s,d,h+j}$, j será 2. Na segunda equação, i continua com valor 1 e j será 3 e, assim por diante, até que j seja superior a carga horária da divisão da cadeira (CH_{dc}) ou $h+j$ seja superior a última hora do dia.
2. Em seguida, i passa a ter valor 2 e j será 3. O processo citado no passo 1 é repetido novamente. Uma vez concluído novamente o passo 1, i passará a ter valor 3 e j será 4, e o passo 1 é repetido novamente, O processo se encerra quando i atingir o valor da carga horária da divisão da cadeira ou que $h+i$ seja igual a última hora do dia.
3. Os passos 1 e 2 são executados para todos as horas H, excluindo a primeira hora do dia ($h=1$), a última hora do dia ($h=UHD$) e a penúltima hora do dia ($h=UHD-1$).
4. Os passos anteriores são repetidos para cada grupo de cadeira, divisão de cadeira, sala e dia da semana.

A tabela 3.2 demonstra exemplos da equação (3.13).

- **SC01** A soft constraint **SC01** tem como objetivo garantir a chamada mancha de aulas compacta. Em outras palavras, é gerar horários compactos para cada turma, ou seja, que todas as aulas atribuídas a uma determinada turma ocorram de forma consecutiva, sem vazios entre elas. Vazios são aquelas horas sem aula atribuída, mas que possui uma hora antes e outra depois com aula atribuída. A tabela 3.3 demonstra horas consideradas como vazios para uma determinada turma.

  Silva [6] tratou de uma necessidade semelhante a SC01 em seu trabalho para elaborar o horário da Universidade Federal Fluminense, porém com uma diferença relevante. No trabalho de Silva, o tamanho dos vazios eram importantes. Foi permitido estabelecer penalidades diferentes conforme o tamanho dos vazios. Portanto, Silva precisou criar variáveis binárias para cada tamanho possível

Equações considerando uma divisão de cadeira
com carga horária = 4 e hora base h = 2

i	j	Equação ($-X_{c,dc,s,d,h} + X_{c,dc,s,d,h+i} - X_{c,dc,s,d,h+j} \leq 0$)
1	2	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=3} - X_{c,dc,s,d,h=4} \leq 0$
1	3	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=3} - X_{c,dc,s,d,h=5} \leq 0$
1	4	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=3} - X_{c,dc,s,d,h=6} \leq 0$
2	3	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=4} - X_{c,dc,s,d,h=5} \leq 0$
2	4	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=4} - X_{c,dc,s,d,h=6} \leq 0$
3	4	$-X_{c,dc,s,d,h=2} + X_{c,dc,s,d,h=5} - X_{c,dc,s,d,h=6} \leq 0$

Tabela 3.2: Tabela com exemplos da equação (3.13)

existente. Para a Universidade Lusófona o tamanho dos vazios não são importantes e sim o total de vazios em um determinado dia.

Hora	Segunda	Terça	Quarta	Quinta	Sexta
1	C_01		C_04		C_02
2	C_02				C_02
3	C_01	C_01			
4	C_03	C_01	C_05		C_04
5	C_03	C_03	C_05		C_04
6	C_03				
7		C_02			
8					
N.º Vazios	0	1	2	0	1

Tabela 3.3: Tabela para demonstrar horas consideradas como vazios

SC01 é considerada uma soft constraint, ou seja, atender essa regra não é obrigatório. Ter vazios entre aulas não impede de obter um resultado válido, apenas obtém-se um resultado pior. Portanto, é incluído na função objetivo uma penalização para cada vazio encontrado.

Para resolver a soft constraint SC01 a solução encontrada foi obter para cada turma e dia da semana, a primeira e a última hora com aula atribuída, bem como, o total de aulas atribuídas neste dia. A diferença entre a última hora e a primeira hora deve ser igual ao total de aulas atribuídas no dia. O número de aulas inferior a diferença entre a última hora e o primeira indica o total de vazios.

Exemplificando, conforme a tabela 3.3, na terça a primeira e última hora com aula atribuída são, respectivamente, as horas 3 e 7 e foi atribuído o total de 4 aulas nesse dia. A partir do momento que foram atribuídas aulas na hora 3 e na hora 7, um calendário ideal seria que a turma tivesse aulas em todos as horas entre a hora 3 e 7, ou seja, 5 aulas no total, porém, só foram atribuídas 4 aulas, logo temos 1 vazio para esse dia.

Para resolver a soft contraint SC01 são necessários quatro conjuntos de equações.

- Obter a primeira hora com aula atribuída para uma determinada turma em um determinado

dia da semana:

$$-P_{t,d} + UHD + 1 - \sum_{c,dc \in DCT_t} \sum_{s \in S} (UHD - h) X_{c,dc,s,d,h} \geq 0 \quad \forall t \in T, \forall d \in D, \forall h \in H \tag{3.14}$$

– Obter a última hora com aula atribuída para uma determinada turma em um determinado dia da semana:

$$-U_{t,d} + \sum_{c,dc \in DCT_t} \sum_{s \in S} (h+1) * X_{c,dc,s,d,h} \leq 0 \quad \forall t \in T, \forall d \in D, \forall h \in H \tag{3.15}$$

– Obter o total de aulas atribuída para uma determinada turma em um determinado dia da semana:

$$-T_{t,d} + \sum_{c,dc \in DCT_t} \sum_{s \in S} \sum_{h \in H} X_{c,dc,s,d,h} \leq 0 \quad \forall t \in T, \forall d \in D \tag{3.16}$$

– Obter o total vazios para uma determinada turma em um determinado dia da semana:

$$U_{t,d} - P_{t,d} + 1 - T_{t,d} - V_{t,d} = 0 \quad \forall t \in T, \forall d \in D \tag{3.17}$$

A variável $V_{t,d}$ é incluída na função objetivo.

As equações (3.4) e (3.17) são lazy constraints. Lazy constraints ou restrições preguiçosas são restrições que provavelmente não serão violadas e, em consequência, deseja-se que sejam aplicadas preguiçosamente, ou seja, apenas conforme necessário ou não antes do necessário. As lazy constraints somente serão inseridas ao modelo depois que a primeira solução que a viola for encontrada.

Um amplo estudo sobre lazy constraints foi realizado por Pearce [36] em seu trabalho de doutoramento. Segundo ele, lazy constrainsts são uma capacidade que foi desenvolvida desde a década de 1990, mas só recentemente se tornou disponível nos solucionadores comerciais de alta potência, Gurobi, CPLEX e XPRESS.

Existem dois métodos de utilizar lazy constraints. Um método é tratar as restrições preguiçosas de forma manual. Para cada solução válida encontrada pelo solver, é analisado se essa solução viola uma ou mais restrição preguiçosa e, caso positivo, informar isso ao solver enviando-lhe as restrições que foram violada. O solver irá incluir essas restrições ao problema e descartar a solução apresentada.

O outro método é deixar essa responsabilidade ao solver. Nesse caso, é enviada uma especie de pool de restrições preguiçosas ao solver. que se encarrega a cada solução válida encontrada, de analisar dentro do pool de restrições preguiçosas se houve violação, rejeitando a solução e incluindo as restrições violada em caso de violação. Se uma solução não violar nenhuma restrição preguiçosa, então essa solução e considerada válida (podendo ou não ser a solução ótima).

A implementação de lazy constraints foi imprescindível para a resolução do problema em tempo útil. O método utilizado na solução foi lazy constraint de forma automática, ou seja, a responsabilidade de utilizar ou não restrições preguiçosas coube ao solver utilizado.

Função Objetivo

Função objetivo: a função objetivo é a minimização do somatório das aulas multiplicadas pelo peso das horas e o somatório dos vazios.

$$min : \sum_{c \in C} \sum_{dc \in DC} \sum_{s \in S} \sum_{d \in D} \sum_{h \in H} PH_{c,h} \cdot X_{c,dc,s,d,h} + \sum_{t \in T} \sum_{d \in D} V_{t,d} \tag{3.18}$$

Resultados

Foi utilizado o dataset do 1º semestre do ano letivo 2019/2020 dos cursos de licenciatura do Departamento de Engenharia Informática e Sistemas de Informação da Universidade Lusófona (DEISI). Os cursos do DEISI são divididos em três: Licenciatura em Engenharia Informática (LEI), Licenciatura em Engenharia Informática, Redes e Telecomunicações (LEIRT) e Licenciatura em Informática de Gestão (LIG). Todos os cursos são de 3 anos, divididos em semestres e podem ser cursados no período diurno (manhã ou tarde) ou no período noturno.

Em função do número de alunos, podem ser criadas mais de uma turma para cada curso. A tabela 3.4 demonstra quantidade de turmas por curso/ano.

	ANO 1	ANO 2	ANO 3	Total
LEI Diurno	4	3	2	9
LEIRT Diurno	1	1	1	3
LIG Diurno	1	1	1	3
LEI Noturno	1	1	1	3
LEIRT Noturno		1	1	2
LIG Noturno	1	1	1	3
Total	8	8	7	23

Tabela 3.4: Tabela para demonstrar a quantidade de turmas por curso e ano

Existem cadeiras que são lecionadas exclusivamente para uma determinada turma, enquanto outras cadeiras podem ser lecionadas para duas ou mais turmas ao mesmo tempo, inclusive turmas de cursos diferentes. Por exemplo, uma determinada cadeira é lecionada para as turmas 1 e 2 do ano 1 do curso LEI diurno e ao mesmo tempo é lecionada para a turma do curso LEIRT diurno ano 1. Enquanto uma

segunda cadeira é lecionada para as turmas 3 e 4 do ano 1 do curso LEI diurno e para a turma do LIG diurno.

Outra característica que é importante destacar, apesar das turmas serem diurnas ou noturnas, algumas cadeiras podem ser lecionadas no outro período. Por exemplo, as turmas diurnas do ano 3 do curso LEI possuem 1 cadeiras que são lecionadas no período noturno. Já as turmas noturnas dos cursos LEIRT e LIG possuem cadeiras que são lecionadas no período diurno. Como será visto mais a frente, essa característica em particular dificulta a formação das manchas de aulas.

Somadas, todas as turmas apresentam 152 cadeiras, com carga horária total de 262,5 horas. Considerando que as aulas são divididas em intervalos de 30 minutos, foram atribuídas 525 aulas. As cadeiras são lecionadas por um total de 31 professores e podem ocupar 26 salas disponibilizadas pela direção.

Todos os testes foram executados em um computador com Intel(R) Core(TM) i7-5500U (2.4 GHz) com 8.0 Gigabytes de memória RAM. A implementação foi utilizando Python-MIP [22] [23] e solver empregado foi o Gurobi (licença acadêmica), versão 9.1.1.

A tabela 3.5 demonstra as violações da nossa solução. Já a tabela 3.6 traz as violações do calendário que foi elaborado de forma manual pela direção o curso. O resultado obtido é melhor daquele produzido de forma manual.

A violação da SC01 ocorreu somente para as turmas cujo horário é diurno com algumas cadeiras lecionadas no período noturno ou turmas cujo horário é noturno, mas possuem cadeiras lecionadas no período diurno. A violação só ocorre para aquelas turmas diurnas com aulas obrigatoriamente no período matutino. Para as turmas diurnas com aulas no período vespertino, a solução conseguiu fazer a mancha compacta, sem vazios.

O horário elaborado de forma manual pela direção do curso, em algumas turmas, apresentam aulas atribuídas nas horas de 08:00 - 08:30, 23:00 - 23:30 e 23:30 - 00:00. A nossa solução penalizou essas horas e dessa forma não foram atribuídas aulas para as mesmas. Porém, não tivemos acesso às limitações dos professores. Não foi possível saber se essas horas eram as únicas horas nos quais o professores estavam disponíveis e não havia como a direção do curso atribuir aulas para outras horas. Se de fato há limitações nos horários dos professores, essa limitação também iria se refletir nos nossos resultados finais.

Os horários gerados podem ser consultados no apêndice E.

Turmas	Vazios (SC01)	Aula 08:00 08:30	Aula 23:00 23:30	Aula 23:30 00:00	Total
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	0	0	0	0	0
14	16	0	0	0	16
15	0	0	0	0	0
16	16	0	0	0	16
17	8	0	0	0	8
18	8	0	0	0	8
19	0	0	0	0	0
20	0	0	0	0	0
21	8	0	0	0	8
22	0	0	0	0	0
23	0	0	0	0	0
Total	48	0	0	0	48

Tabela 3.5: Tabela para demonstrar as violações obtidas pela solução apresentada. As turmas 14 e 16 são turmas noturnas com algumas cadeiras lecionadas no período diurno e as turmas 17 e 18 são turmas diurnas com algumas cadeiras lecionadas no período noturno

Turmas	Vazios (SC01)	Aula 08:00 08:30	Aula 23:00 23:30	Aula 23:30 00:00	Total
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	2	0	0	0	2
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	10	0	0	0	10
9	1	40	0	0	41
10	5	40	0	0	45
11	2	40	0	0	42
12	1	0	0	0	1
13	2	30	0	0	32
14	24	0	0	0	24
15	2	40	0	0	42
16	43	0	0	0	43
17	3	20	0	0	23
18	5	10	0	0	15
19	1	0	30	10	41
20	2	30	0	0	32
21	38	0	10	0	48
22	14	40	0	0	54
23	0	0	10	0	10
Total	155	290	50	10	505

Tabela 3.6: Tabela para demonstrar as violações do horário utilizado pela Universidade no semestre 1 do ano letivo 2019/2020 (elaborado de forma manual pela direção do curso)

4. Caso de estudo 2: ITC 2007

International Timetabling Competition 2007, ou ITC 2007, foi uma competição de UTCP realizada em 2007 com objetivo gerar novas abordagens para os problemas associados, atraindo usuários de todas as áreas de pesquisa e também fechar a lacuna existente entre a pesquisa e a prática nesta importante área da pesquisa operacional [1].

O ITC 2007 foi dividido em 3 categorias: horários de exames, horário universitário com base na pós-matricula e horário universitário com base no currículo. Foram disponibilizados datasets com dados reais da Universidade de Udine na Itália. A competição consistia em desenvolver soluções para resolver os datasets que foram disponibilizados.

Para cada categoria, foram estabelecidos hard constraints e soft constrains. Violar uma constraint significava uma penalização em pontos. As soluções vencedoras foram aquelas que apresentavam menor pontuação.

Ao implementar uma solução para o ITC 2007 permitiu comparar a nossa solução com outras soluções. Embora tenha algumas diferenças, as exigências da Universidade Lusófona equivalem a categoria *horário universitário com base no currículo* do ITC 2007.

Foram disponibilizados relatórios técnicos (technical report) para cada um das categorias. A categoria *horário universitário com base no currículo* foi detalhado no relatório técnico elaborado por Gaspero, McCollum e Schaerf [26]. Baseado nesse relatório, será apresentado um breve resumo da competição, em seguida a solução adotada e por fim, os resultados obtidos.

4.1 Resumo

O problema consiste no agendamento semanal das aulas de várias cadeiras dentro de um determinado número de salas e períodos de tempo, onde os conflitos entre as cadeiras são definidos de acordo com os currículos publicados pela universidade. Cada dataset era formado pelas seguintes entidades:

- Total de dias de aulas por semana (geralmente 5 ou 6 dias)
- Total de horários por dias de aula
- Cadeiras, cada cadeira é composta pelo professor, carga horária semanal, número mínimo de dias que a carga horária deve ser distribuída e a quantidade de alunos que irão frequentar a cadeira.
- Salas e a sua capacidade máxima de alunos. Todas as salas estão disponíveis para todas as cadeiras.

- Currículos. O currículo é um conjunto de cadeiras que tenha alunos em comum
- Indisponibilidade do professor. Conjunto de dias / horários que um determinado professor não está disponível para dar aula.

Lista das restrições

- Restrições rígidas (hard constraints)
  - **Lectures** Todas as aulas de uma cadeira devem ser programadas, e devem ser distribuídas em períodos distintos. Uma violação ocorre se uma palestra não for agendada.
  - **RoomOccupancy** Duas aulas não podem ocorrer na mesma sala no mesmo período. Duas ou mais aulas na mesma sala e no mesmo período representam uma violação.
  - **Conflicts** Aulas de cadeiras do mesmo currículo ou ministradas pelo mesmo professor devem ser todas programadas em períodos diferentes. Duas ou mais aulas conflitantes no mesmo período representam uma violação.
  - **Availabilities** Se o professor não estiver disponível para ministrar aula em um determinado período, nenhuma aula poderá ser agendada naquele período. Cada aula em um período indisponível para o professor é uma violação.
- Restrições flexíveis (soft constraints)
  - **RoomCapacity** Para cada aula agendada, o número de alunos da cadeira deve ser menor ou igual ao número de vagas da sala atribuída. Cada aluno acima da capacidade conta como 1 ponto de penalidade.
  - **MinimumWorkingDays** As aulas de cada cadeira devem ser distribuídas no número mínimo de dias indicado. Cada dia abaixo do mínimo conta como 5 pontos de penalização.
  - **CurriculumCompactness** As aulas pertencentes a um currículo devem ser adjacentes umas às outras (ou seja, em períodos consecutivos). Para um determinado currículo, é contabilizado uma violação toda vez que há uma aula não adjacente a qualquer outra aula no mesmo dia. Cada aula isolada em um currículo conta como 2 pontos de penalidade.
  - **RoomStability** Todas as aulas de uma cadeira devem ser ministradas na mesma sala. Cada sala distinta utilizada para as aulas de um curso conta como 1 ponto de penalização.

Foi disponibilizado aos participantes um validador desenvolvido em C++ para validarem seus resultados. O validador através de um arquivo TXT com a solução, avaliava e produzia uma descrição detalhada de todas as violações (hard e soft). Tivemos acesso a esse validador, o que nos ajudou validar nossos resultados.

4.2 Solução

Conjuntos

- C: Cadeiras, c ∈ C = {Cadeira 1, Cadeira 2, ... C}
- P: Professores, p ∈ P = {Professor 1, Professor 2, ... P}
- CP: Cadeiras do Professor, cp ∈ CP = {{Professor 1, {Cadeira 1, Cadeira, 2}}, ... {p, {c}} }
- CH: Carga horária das cadeiras, ch ∈ CH = {{Cadeira 1, 4}, ... {c, ch}}
- C_MIN_DIAS: Número mínimo de dias de aula por cadeira, c_min ∈ C_MIN = {{Cadeira 1, 2}, ... {c, c_min}}
- C_ALUNOS: Quantidade de alunos por cadeira, c_alunos ∈ C_ALUNOS = {{Cadeira 1, 50}, ... {c, c_alunos}}
- S: Sala de Aula, s ∈ S = {Sala 1, Sala 2, ... S}
- S_CAP: Capacidade da sala de Aula, s_cap ∈ S_CAP = {{Sala 1, 70}, ... {s, s_cap}}
- R: Currículos, r ∈ R = {Currículo 1, Currículo 2, ... R}
- R_C: Cadeiras do currículo, r_c ∈ R_C = {{Currículo 1, {Cadeira 1, Cadeira 2}}, ... {r, {c}} }
- UC: Períodos de indisponibilidades dos professores (unavailability constraints), uc ∈ uc = {{Professor 1, Dia 1, Horário 1}, ... {c, d, h}}

Parâmetros

- D = Total de dias por semana
- H = Total de horários por dia

Variáveis Auxiliares

- $Z_{r,d,h}$ = Variável binária que indica se um currículo tem aula atribuída em um determinado dia e horário.
- $I_{r,d,h}$ = Variável binária que indica se um currículo tem aula isolada em um determinado dia e horário.
- $Y_{c,d}$ = Variável binária que indica se uma cadeira tem aula atribuída em um determinado dia.
- $W_{c,d}$ = Variável inteira com o total de aulas atribuídas para uma cadeira em um determinado dia.
- M_c = Variável inteira que indica a quantidade de dias a menos que o exigido que uma determinada cadeira tem aulas atribuídas.

- $V_{c,s}$ = Variável binária que indica se uma cadeira tem aula atribuída em uma determinada sala.
- $E_{c,s}$ = Variável inteira que indica o total de salas com aulas atribuídas para uma determinada cadeira menos 1.
- $Q_{c,s}$ = Variável inteira com a diferença entre a quantidade de alunos de uma determinada cadeira com a capacidade de uma determinada sala. Se a capacidade da sala for igual ou superior a quantidade de alunos da cadeira, essa variável receberá zero.

Variáveis de decisão

As variáveis de decisão é uma matriz binária multidimensional de aulas, fruto da combinação entre Cadeiras, Salas, Dias de Semanas e Horários.

$$\begin{gathered} X_{c,s,d,h}\{0,1\} \\ \forall c \in C, \forall s \in S, \forall d \in D, \forall h \in H \end{gathered} \tag{4.1}$$

$X_{c,s,d,h}$ receberá 1 caso tenha uma aula atribuída e 0 caso contrário.

Restrições

- **Lectures** A hard constraint **Lectures** tem como objetivo garantir o cumprimento da carga horária das divisões de cadeiras.

  Para isto, a somatória de aulas atribuídas para uma determinada cadeira tem que ser igual a carga horária estabelecida.

$$\sum_{s \in S} \sum_{d \in D} \sum_{h \in H} X_{c,s,d,h} = CHc \qquad \forall c \in C \tag{4.2}$$

- **RoomOccupancy** A hard constraint **RoomOccupancy** tem como objetivo garantir que duas ou mais aulas não ocorram na mesma sala e no mesmo período.

  Para isto, a somatória das aulas atribuídas para uma determinada sala em um determinado dia da semana e em um determinado horário deve ser igual ou menor que 1.

$$\sum_{c \in C} X_{c,s,d,h} \leq 1 \qquad \forall s \in S, \forall d \in D, \forall h \in H \tag{4.3}$$

- **Conflicts** A hard constraint **Conflicts** tem como objetivo garantir que as aulas das cadeiras do mesmo currículo ou ministradas pelo mesmo professor devem ser todas programadas em períodos diferentes.

  Para isto, foi necessário duas equações. Para garantir que o professor não tenha duas aulas ou mais atribuídas em um mesmo período a somatória das aulas atribuídas a um professor em um determinado dia e em um determinado horário tem que ser igual ou menor que 1.

$$\sum_{p \in P} \sum_{c \in CP_p} \sum_{s \in S} X_{c,s,d,h} \leq 1 \qquad \forall d \in D, \forall h \in H \tag{4.4}$$

Para garantir que as cadeiras de um determinado currículo não tenham aulas atribuídas no mesmo período a somatória das aulas atribuídas das cadeiras de um determinado currículo em um determinado dia e em um determinado horário quem que ser igual ou menor a 1.

Porém, a equação foi adaptada para integrar a solução das aulas isoladas. Para as aulas isoladas, é necessário saber se um currículo tem aula atribuída em um determinado dia e horário, por isso foi criada a variável binária auxiliar $Z_{r,d,h}$. Para que essa variável seja preenchida corretamente, a somatória das aulas atribuídas das cadeiras de um determinado currículo em um determinado dia e em um determinado horário menos $Z_{r,d,h}$ tem que ser igual a zero.

Isso garante que $Z_{r,d,h}$ receberá 1 se foi atribuída alguma aula para o dia e horário ou 0 se não foi atribuído aula.

$$\sum_{r \in R} \sum_{c \in R\_C_r} \sum_{s \in S} X_{c,s,d,h} - Z_{r,d,h} = 0 \qquad \forall d \in D, \forall h \in H \tag{4.5}$$

- **Availabilities** A hard constraint **Availabilities** tem como objetivo garantir que os professores não tenham aulas nos períodos que não estão disponíveis.

  Para isto, as aulas dos professores nos períodos de indisponibilidade devem ser igual a zero.

$$X_{c,s,d,h} = 0 \qquad \forall p \in P, \forall c \in CP_p, \forall d, h \in UC_p, \forall s \in S \tag{4.6}$$

- **MinimumWorkingDays** A soft constraint **MinimumWorkingDays** tem como objetivo garantir que o número mínimo de dias de uma cadeira. Foi necessário incluir 4 equações.

  A primeira equação irá obter os dias que a cadeira tem aulas atribuídas. Para isso foi criado a variável binária $Y_{c,d}$ que indica se a cadeira tem ou não aula em um determinado dia.

$$X_{c,s,d,h} - Y_{c,d} \leq 0 \qquad \forall c \; inC, \forall s \in S, \forall d \in D, \forall h \in H \tag{4.7}$$

  A segunda equação irá obter o total de dias a menos que o mínimo exigidos para a cadeira. Para isso foi criado a variável M_c que irá para a função objetiva com peso 5.

$$C\_MIN\_DIAS_c - \sum_{d \in D} Y_{c,d} - M_c \leq 0 \qquad \forall c \in C \tag{4.8}$$

  As equações 4.7 e 4.8 não garantem que a variável $Y_{c,d}$ receba zero nos dias que a cadeira não tem aula. Foi necessário adicionar duas novas equações para garantir que a variável $Y_{c,d}$ só receba 1 nos dias que a cadeira tem aula atribuída.

  A primeira equação irá obter o total de aulas atribuídas no dia para uma determinada cadeira. Foi criado a variável inteira $W_{c,d}$.

$$\sum_{s \in S} \sum_{h \in H} X_{c,s,d,h} - W_{c,d} = 0 \qquad \forall c \in C, \forall d \in D \tag{4.9}$$

E finalmente a próxima equação vai garantir que só seja atribuído 1 para $W_{c,d}$ nos dias que a cadeira tiver aula.

$$Y_{c,d} - W_{c,d} \leq 0 \qquad \forall c \in C, \forall d \in D \tag{4.10}$$

- **CurriculumCompactness** A soft constraint **CurriculumCompactness** tem como objetivo garantir que as aulas pertencentes a um currículo devem ser adjacentes umas às outras, evitando aulas isoladas.

  Para resolver está soft constraint foi utilizada a solução apresentada por Lach [20]. São 3 conjuntos de equações que utilizam como base a variável $Z_{r,d,h}$ que tem valor 1 quando um determinado currículo tem aula atribuída em um determinado dia e horário e 0 caso contrário. Essa variável foi preenchida na equação 4.5.

  Foi criada a variável binária $I_{r,d,h}$ para apontar as aulas isoladas. $I_{r,d,h}$ irá receber 1 quando existir uma aula isolada. Essa variável fará parte da função objetiva com peso 2.

  Para o primeiro horário do dia:

$$Z_{r,d,h} - Z_{r,d,h+1} - I_{r,d,h} \leq 0 \qquad \forall r \in R, \forall d \in D, h = 1 \tag{4.11}$$

  Para o último horário do dia:

$$-Z_{r,d,h-1} + Z_{r,d,h} - I_{r,d,h} \leq 0 \qquad \forall r \in R, \forall d \in D, h = H \tag{4.12}$$

  Para os demais horários do dia:

$$-Z_{r,d,h-1} + Z_{r,d,h} - Z_{r,d,h+1} - I_{r,d,h} \leq 0 \qquad \forall r \in R, \forall d \in D, \forall h \in H, h > 1, h < H \tag{4.13}$$

- **RoomStability** A soft constraint **RoomStability** tem como objetivo garantir que as aulas de uma determinada cadeira sejam todas ministradas na mesma sala.

  Foi necessário a criação de duas variáveis auxiliares. A primeira $V_{c,s}$, variável binária que indica se uma cadeira tem aula atribuída em uma determinada sala e E_c, variável inteira que indica o total de salas excedentes com aulas atribuídas para uma determinada cadeira. A variável E_c é incluída na função objetiva com peso 1.

  Duas equações são necessárias. A primeira equação tem como objetivo obter quais as salas que a cadeira tem aulas atribuídas.

$$X_{c,s,d,h} - V_{c,s} \leq 0 \qquad \forall c \in C, \forall s \in S, \forall d \in D, \forall h \in H \tag{4.14}$$

A segunda equação irá preencher a variável E_c e obter as salas excedentes para uma cadeira.

$$\sum_{\forall s \in S} V_{c,s} - E_c - 1 = 0 \qquad \forall c \in C \tag{4.15}$$

Função Objetivo

Função objetivo: a função objetivo é a composta pelas penalizações das soft contraints **MinimumWorkingDays** (M_c), **CurriculumCompactness** ($I_{r,d,h}$) e **RoomStability** (E_c). O custo de cada penalização (5, 2, e 1 respectivamente) foram estabelecidos pela organização do ITC 2007.

Além disso, também está incluído na função objetivo o tratamento da soft constraint **RoomCapacity**. A soft constraint **RoomCapacity** penaliza atribuir uma cadeira a uma sala cuja capacidade é inferior a quantidade de alunos da cadeira. A variável $Q_{c,s}$ possui a diferença entre capacidade da sala versus quantidade de alunos

$$min : \sum_{c \in C} \sum_{s \in S} \sum_{d \in D} \sum_{h \in H} Q_{c,s} \cdot X_{c,s,d,h} + \sum_{c \in C} 5 \cdot M_c + \sum_{r \in R} \sum_{d \in D} \sum_{h \in H} 2 \cdot I_{r,d,h} + \sum_{c \in C} \sum_{s \in S} E_c \tag{4.16}$$

Execução

Para a obtenção de melhores resultados, a execução da solução ocorreu em 3 passos e está detalhado na figura 4.1.

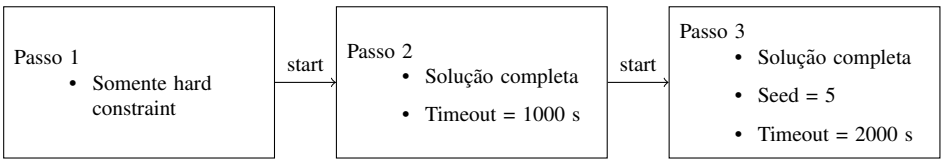

Figura 4.1: ITC 2007 - Execução da solução

O passo 1 consistiu em retirar da solução completa as soft constraints **MinimumWorkingDays**, **CurriculumCompactness** e **RoomStability**. Com isso a função objetiva foi alterada para a seguinte:

$$min : \sum_{c \in C} \sum_{s \in S} \sum_{d \in D} \sum_{h \in H} Q_{c,s} \cdot X_{c,s,d,h} \tag{4.17}$$

A execução do passo 1 ocorre em segundos e é obtido o resultado ótimo, obviamente quebrando diversas soft constraints. Portanto, é uma solução factível para o problema completa, mas não a solução ótima.

O passo 2 consiste em executar o problema completo, porém é realizado um *start* com o resultado obtido no passo 1. O *start* é um recurso da ferramenta Python-MIP que permite informar ao solver uma solução factível e dessa forma o solver faz a análise a partir dessa posição de *start*.

Foi estabelecido um timeout de 1000 segundos de execução para o passo 2. Para todas as instâncias do ITC 2007 não foi possível obter uma solução ótima em 1000 segundos. Mas é encontrada uma solução factível melhor do que a solução obtida passo 1.

A solução obtida no passo 2 é utilizada de *start* para o passo 3. O passo 3 é executado com timeout de 16200 segundos com alteração no *seed* para 5. *Seed* (ou semente) é parâmetro que, perturba aleatoriamente o solver com o objetivo de alterar a primeira solução ótima do nó raiz e, em seguida, todo o caminho de pesquisa do branch and cut. Ou seja, permite modificar a árvore de pesquisa utilizada pelo solver e analisar outras soluções possíveis. Seguir um caminho diferente na árvore pode levar a encontrar a solução ótima mais rapidamente.[28]

Alterar o *seed* permitiu melhor o desempenho da solução, obtendo melhores resultados com o mesmo tempo de execução. Como exemplo, para a instância 2, utilizando o *seed* default do solver, o resultado de violações foi de 113. Ao alterar o *seed* para 5, o resultado obtido foi de 33.

Resultados

Todos os testes foram executados em um computador com Intel(R) Core(TM) i7-5500U (2.4 GHz) com 8.0 Gigabytes de memória RAM. A implementação foi utilizando Python-MIP [22] [23] e solver empregado foi o Gurobi (licença acadêmica), versão 9.5.0.

A tabela 4.1 demonstra o total de violações que obtivemos por soft contraint. Já a tabela 4.2 traz os resultados dos finalistas do ITC 2007. Os valores em negritos são os melhores resultados e os valores em itálicos são aqueles resultados que foram superados pelo nosso trabalho. Os valores com asterisco indicam que o valor ótimo foi encontrado.

Instância	1	2	3	4	5	6	7
RoomCapacity	4	0	0	0	0	0	0
MinWorkingDays	0	5	35	5	180	20	0
IsolatedLectures	0	28	50	30	146	52	26
RoomStability	1	0	0	0	3	11	25
Total	5	33	85	35	329	83	51
Instância	8	9	10	11	12	13	14
RoomCapacity	0	0	0	0	0	0	0
MinWorkingDays	5	30	5	0	210	10	5
IsolatedLectures	32	68	38	0	230	58	54
RoomStability	0	0	12	0	0	0	0
Total	37	98	55	0	440	68	59
Instância	15	16	17	18	19	20	21
RoomCapacity	0	0	0	0	0	2	0
MinWorkingDays	15	15	20	10	20	5	15
IsolatedLectures	66	38	100	56	40	56	122
RoomStability	0	1	5	0	0	18	1
Total	81	54	125	66	60	81	138

Tabela 4.1: ITC 2007 - Resultados obtidos

Instância	1	2	3	4	5	6	7
T. Muller	**5**	*51*	*84*	*37*	*330*	**48**	**20**
Z. Lu et al.	**5**	*55*	**71**	*43*	**309**	53	28
M. Atsuta et al.	**5**	*50*	82	**35**	312	69	42
M Geiger	**5**	*111*	*128*	*72*	*410*	*100*	57
M. Clark et al.	10	*111*	*119*	*72*	*426*	*130*	*110*
Luciano Almeida	**5***	**33**	85	**35***	329	83	51
Instância	8	9	10	11	12	13	14
T. Muller	*41*	*109*	**16**	**0**	**333**	**66**	*59*
Z. Lu et al.	*49*	*105*	21	**0**	343	*73*	**57**
M. Atsuta et al.	*40*	*110*	27	**0**	351	68	*59*
M Geiger	*77*	*150*	*71*	**0**	*442*	*622*	*90*
M. Clark et al.	*83*	*139*	*85*	3	408	*113*	*84*
Luciano Almeida	**37***	**98**	55	**0***	440	68	59
Instância	15	16	17	18	19	20	21
T. Muller	*84*	**34**	**83**	*83*	*62*	**27**	**103**
Z. Lu et al.	**71**	39	91	69	*65*	47	106
M. Atsuta et al.	*82*	40	102	**68**	*75*	61	123
M Geiger	*128*	*81*	124	*116*	*107*	*88*	*174*
M. Clark et al.	*119*	*84*	*152*	*110*	*111*	*144*	*169*
Luciano Almeida	81	54	125	**66**	**60**	81	138

Tabela 4.2: ITC 2007 - Resultados dos finalistas * solução ótima

Conclusão

O trabalho desenvolvido para a Universidade Lusófona focou em melhorar o horário na ótica dos alunos. Evitar aulas nos primeiros horários e nos últimos horários do dia e obter a mancha de aulas sem vazios, melhora consideravelmente o conforto para os alunos.

O resultado foi um horário sem violar essas soft constraints melhorando consideravelmente o horário elaborado de forma manual pela direção dos cursos.

Com a ressalva de que não tivemos acesso às limitações dos professores e por consequência os horários que poderiam dar aulas, podemos afirmar que os resultados obtidos pela nossa solução é eficiente e atendeu aquilo que foi proposto a ser feito. A utilização de programação linear leva a solução apresentar resultados ótimos para a formulação utilizada.

Com o intuito de comparar nosso trabalho com outros trabalhos, adaptamos a nossa solução para o ITC 2007. O ITC 2007 foi uma competição de calendarização ocorrida em 2007 com 21 instâncias. A competição contou com os principais estudiosos na área de elaboração de horário. O nosso trabalho não nos daria a vitória na competição, mas estaríamos entre os cinco primeiros. Os resultados obtidos superou em quase todas as instâncias dois ou mais finalistas e em três instâncias conseguimos obter a melhor solução.

Trabalhos futuros

Melhorar o horário para o professor

O foco da solução foi melhorar o horário para os alunos. A solução não preocupou em melhorar o horário para os professores. Muito embora não ter aulas no inicio e fim do dia também é bom para os professores, em nenhum momento a solução se preocupa em tratar das manchas dos professores ou em diminuir os dias que o professores tem aulas.

Analisando o horário gerado, é possível notar que existem situações que é perfeitamente possível criar uma mancha. Como exemplo, um determinado professor que leciona cadeiras nos períodos vespertino e noturno, em uma segunda foram atribuídas aulas para o horário das 14:00 as 16:00 e das 18:00 as 20:00. O ideal para o professor, nesse caso, seria que o horário da cadeira do período vespertino fosse das 16:00 as 18:00, criando assim uma mancha de aulas.

Já para um segundo professor que leciona duas cadeiras, uma no período vespertino e outra no período noturno, as aulas foram atribuídas para dois dias diferentes. O ideal seria que as aulas fossem atribuídas para somente um dia e ainda que fossem consecutivas.

A proposta de trabalho futuro é incluir na solução duas novas soft constraints. Uma para tratar da

mancha do professor e a segunda para tratar na redução dos dias que o professor tem aulas atribuídas. Ao integrar o horário dos professores passa-se a ter um problema de otimização multiobjetivo, envolvendo mais do que uma função objetivo que devem ser otimizadas em simultâneo. Sugere-se a utilização de técnicas especificas para a solução problemas de otimização multiobjetivo.

Mancha horária por período

A mancha para os alunos foi tratada com sucesso pela solução apresentada. Mas existe o caso das turmas que tem aulas em um período diferente do seu período. É o caso das turmas noturnas com algumas cadeiras no período matutino ou turmas matutinas com algumas cadeiras noturnas.

Garantidamente essas turmas terão vazios entre as suas aulas. O sistema fica tentando eliminar ao máximo esses vazios que irão existir. Isso prejudica o desempenho da solução. Melhorar a forma como são tratadas as manchas dos alunos, pode vir a melhorar o desempenho.

A proposta de trabalho futuro aqui é tratar a mancha por períodos e não por todos os horários do dia. Então para as turmas que tem aulas em mais de um período, dividir o tratamento da mancha por períodos.

Pode-se também implementar uma solução que penalize atribuir para uma mesma turma aulas em períodos distintos em um mesmo dia. Isso significa evitar que a turma tenha aulas em dois períodos no mesmo dia. Irá ajudar a evitar vazios no horário dos alunos. Cabe a ressalva que implementar essa sugestão pode prejudicar o desempenho da solução e, em função da carga horária, nem sempre é possível evitar aulas em períodos distintos.

Formação de turmas e disponibilidade das cadeiras

Não entrou no escopo da nossa solução para a Universidade Lusófona a formação de turmas. O dataset utilizado já tinha as turmas formadas, seguindo aquilo que foi utilizado pela universidade no período 2019/2020. Por exemplo, o curso LEI, em função do número de alunos matriculados, teve 4 turmas no primeiro ano. A definição desse número (4 turmas) foi responsabilidade da direção do curso. No entanto, com variações no número de alunos de ano para ano é possível que o número de turmas de cada curso também varie.

Da mesma forma que as quantidades de turmas já estavam no dataset utilizado, a disponibilidade das cadeiras também. Por exemplo, um determinada cadeira tem aulas teóricas e práticas. Os alunos do primeiro ano dos cursos LEI e LEIRT devem cursar essa cadeira. Como já foi citado, no primeiro ano, o curso LEI teve 4 turmas. O curso LEIRT, 1 turma. Para as aulas práticas, a direção do curso disponibilizou uma cadeira para cada turma. Ou seja, 5 turmas, a cadeira foi lecionada 5 vezes. Já para a as aulas teóricas, só foi disponibilizado duas cadeiras. Uma cadeira de aula teórica para as turmas LEI 1 e LEI 2 e outra cadeira para as turmas LEI 3, LEI 4 e para a turma de LEIRT. Novamente não é possível saber se o número de cadeiras disponibilizadas é o número certo.

Propõe-se incluir parâmetros que permita ao sistema estabelecer o número de turmas necessárias e quantidade de cadeiras que devem ser disponibilizadas.

Aplicar técnicas de geração de colunas ou decomposição de Dantzig-Wolfe

Geração de colunas e decomposição de Dantzig-Wolfe são ambas técnicas para resolver problemas de programação linear de elevada dimensão. Foi observado que a matriz de restrições apresenta as características necessárias para a aplicação das técnicas geração de colunas ou de decomposição de Dantzig-Wolfe, entre elas grupos restrições independentes entre si e um grupo de restrições de ligação entre os grupos independentes.

Acredita-se que a utilização dessas técnicas representem ganhos em questões de performances, principalmente aplicando ao problema do ITC 2007.

ITC 2007: Entender o motivo da variabilidade de performance nos resultados obtidos

Uma característica apresentada na solução para o ITC 2007 foi a variabilidade de performance no resultados entre as diversas instâncias. Em algumas delas os resultados são melhores do que a média e muito piores do que a média em outras instâncias. A mesma situação acontece para os resultados finalistas. Não está claro o motivo disso acontecer. É interessante realizar um estudo para entender o motivo disso acontecer.

Bibliografia

[1] ITC 2007. International timetabling competition 2007. http://www.cs.qub.ac.uk/itc2007/index.htm, acessado em 07/04/2021.

[2] Adriane Beatriz de Souza Serapião. Fundamentos de otimização por inteligência de enxames: uma visão geral. *Revista Controle & Automação*, 20(3):271–304, 2009.

[3] A.J. Hoffman. Total unimodularity and combinatorial theorems. *Linear Algebra and its Applications*, 13(1):104, 1976.

[4] Alfian Akbar Gozali, Jimmy Tirtawangsa e Thomas Anung Basuki. Asynchronous Island Model Genetic Algorithm for University Course Timetabling. *10th International Conference of the Practice and Theory of Automated Timetabling*, 1(1):26–29, 2014.

[5] Anderson Roges Teixeira Góes. *Otimização na distribuição da carga horária de professores–método exato, método heurístico, método misto e interface*. Universidade Federal do Paraná, Curitiba, Brasil, 2005.

[6] André Renato Villela da Silva. Uma formulação matemática para o problema da alocação de horários em um curso universitário: um estudo de caso. *XLVI Simpósio Brasileiro de Pesquisa Operacional*, 1(1):2704–2715, 2014.

[7] Graham Kendall e Nasser R. Sabar Anmar Abuhamdah, Masri Ayob. Population based local search for university course timetabling problems. *Applied Intelligence*, 40:44–53, 2013.

[8] Ayla Gülcü e Can Akkan. Bi-Criteria Simulated Annealing Algorithms for the Robust University Course Timetabling Problem. *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018), Vienna, Austria*, 1(1):129–136, 2018.

[9] Rainer E. Burkard. Selected topics on assignment problems. *Discrete Applied Mathematics*, 123(1):257–302, 2002.

[10] Clemens Nothegger, Alfred Mayer, Andreas Chwatal e Gunther R. Raidl. Solving the Post Enrolment Course Timetabling Problem by Ant Colony Optimization. *Annals of Operations Research*, 194(1):325–339, 2012.

[11] Dirk G. Cattrysse; Marc Salomon; Luk N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, 72(1):164–174, 1994.

[12] Lina Pupeikienė e Eugenijus Kurilovas. Optimal School Scheduling Problem. *INFORMACIJOS MOKSLAI*, 50(1):69–73, 2009.

[13] T. Stutzle e H. H. Hoos. Max -min ant system. *Future Generation Computer Systems*, 16:889–914, 2000.

[14] Krzysztof Socha e Joshua Knowles e Michael Sampels. A max-min ant system for the university course timetabling problem. *Ant Algorithms*, 1-13, 10 2002.

[15] Nurul Liyana Abdul Aziz e Nur Aidya Hanum Aizam. A brief review on the features of university course timetabling problem. *AIP Conference Proceedings*, 2016.

[16] Guilherme Brandelli Bucco e outros. Desenvolvimento de um modelo de programação linear para o Problema da Construção de Grades Horárias em Universidades. *Gest. Prod., São Carlos, Brasil*, 24(1):40–49, 2017.

[17] Yoshiko Wakabayashi e outros. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Universidade de São Paulo, São Paulo, Brasil, 2001.

[18] Simon Kristiansen e Thomas Riis Stidsen. A Comprehensive Study of Educational Timetabling - a Survey. *DTU Management Engineering Report*, 8.2013, 2013.

[19] Fabrício Bueno. *Métodos Heurísticos - Teoria e Implementações*. Instituto Federal de Santa Catarina, Araranguá, Santa Catarina, 2009, 2009.

[20] Gerald Lach e Marco E. Lubbecke. *Curriculum Based Course Timetabling: Optimal Solutions to the Udine Benchmark Instances*. Technische Universit at Berlin, Institut fur Mathematik, Berlim, Alemanha, 2008.

[21] Haroldo G. Santos e Marcone Jamilson Freitas Souza. Programação de Horários em Instituições Educacionais: Formulações e Algoritmos. *XXXIX SBPO - A Pesquisa Operacional e o Desenvolvimento Sustentável, Fortaleza, Brasil*, 29(1):2827–2882, 2007.

[22] Haroldo G. Santos e Túlio A. M. Toffolo. Tutorial de Desenvolvimento de Métodos de Programação Linear Inteira Mista em Python usando o pacote python-mip. *Pesquisa Operacional para o Desenvolvimento*, 11(3):127–138, 2019.

[23] Haroldo G. Santos e Túlio A. M. Toffolo. *Mixed Integer Linear Programming with Python*. COIN-OR Computational Infrastructure for Operations Research, 2020.

[24] John E. Mitchell. *Branch-and-Cut Algorithms for Combinatorial Optimization Problems*. Mathematical Sciences Rensselaer Polytechnic Institute Troy, NY, USA, 1999.

[25] R.M. Karp. Reducibility among combinatorial problems. *Proceedings of the Complexity of Computer Computations*, pages 85–103, 1972.

[26] Luca Di Gaspero, Barry McCollum E Andrea Schaerf. *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*. ITC 2007, 2007.

[27] M Akif Bakir e Cihan Aksop. A 0-1 integer programming approach to a university timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, 37(1):41–55, 2008.

[28] M. Monaci Fischetti. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.

[29] Marco Dorigo, Mauro Birattari e Thomas Stützle. Exploiting erraticism in search. *Ant Colony Optimization*, November 2006(1):28–39, 2006.

[30] Marcone Jamilson Freitas Souza. *Programação de horários em escolas: uma aproximação por metaheurísticas*. Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2000.

[31] Marcos Thomaz da Silva. *Hibridização de métodos exatos e heurísticos para a minimização do atraso ponderado no escalonamento de tarefas em máquinas paralelas*. Universidade Federal do Amazonas, Manaus, Brasil, 2018.

[32] Martin Ledermann e Nivia Maria Kinalsk. *Pesquisa Operacional*. Universidade Regional do Noroeste do Estado do Rio Grande do Sul - Unijuí, Ijuí, Brasil, 2012.

[33] Mokhtar S. Bazaraa, John J. Jarvis e HanifD.Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2010.

[34] NP-completeness. Np-completeness. https://en.wikipedia.org/wiki/np-completeness, acessado em 29/08/2021.

[35] PATAT International Conference on the Practice and Theory of Automated Timetabling. Patat - international conference on the practice and theory of automated timetabling. https://patatconference.org/, acessado em 09/09/2021.

[36] Robin H. Pearce. *Towards a general formulation of lazy constraints*. The University of Queensland, Austrália, 2019.

[37] Rui Alves e Catarina Delgado. *Programação Linear Inteira*. Faculdade de Economia da Universidade do Porto, 1997.

[38] A. Schaerf. *A Survey of Automated Timetabling*. Università di Roma "La Sapienza", Itália, 1999.

[39] Robert Fabrício Subtil. *Problema generalizado de atribuição: contribuições ao estudo de algoritmos mono e multiobjetivos*. Centro Federal de Educação Tecnológica de Minas Geral, Belo Horizonte, Brasil, 2012.

[40] Toha Ardi Nugraha, Karisma Trinanda Putra e Nur Hayati. University Course Timetabling with Genetic Algorithm: A Case Study. *Journal of Electrical Technology UMY (JET-UMY)*, 1(2):100–105, 2017.

[41] Tomas Muller and Roman Bartak Hana Rudova. Minimal perturbation problem in course timetabling. *Practice and Theory of Automated Timetabling*, V(1):126–146, 2005.

A. Problema de Otimização

Problemas de otimização têm como objetivo encontrar um ponto ótimo (mínimo ou máximo) de uma função definida sobre um certo domínio [17]. Em outras palavras, são algoritmos que procuram encontrar um conjunto de soluções factíveis, partindo de um conjunto de dados e uma função objetiva que atribui um valor a cada solução factível encontrada. Quando o algoritmo não encontra nenhuma solução factível, considera-se o problema como não solucionável.

O objetivo final é obter soluções de valor mínimo (para problemas de minimização) ou soluções de valor máximo (para problemas de maximização). Ao encontrar soluções mínimas ou máximas, diz-se que foi encontrado o valor ótimo e a solução que possui o valor ótimo designa-se por solução ótima.

É perfeitamente possível aplicar técnicas de otimização para problemas de diversas áreas, tais como engenharia, de administração, de logística, de transporte, de economia, de biologia, etc, desde que se consiga construir modelos matemáticos para os problemas. Este trabalho foi resolvido aplicando técnicas de otimização.

A.1 Programação linear

Para Bazaraa [33], programação linear abrange problemas de otimização (minimização ou maximização) de uma função linear enquanto satisfaz um conjunto de restrições ou restrições de igualdade e / ou desigualdade linear. Foi concebido por George B. Dantzig, em 1947, no período em que ele trabalhava como consultor matemático do Controlador da Força Aérea dos Estados Unidos no desenvolvimento de uma ferramenta de planejamento para um programa de distribuição, treinamento e abastecimento logístico. É importante ressalvar que o termo *programação* tem o sentido de planejamento e não de programação de computadores.

Ledermann [32], didaticamente, define um roteiro simples para a construção de um modelo matemático para a programação linear: determinação das variáveis de decisão, determinação da função objetivo e determinação das restrições.

Determinar as variáveis de decisão consiste em identificar as decisões que devem ser tomadas e representa-las em variáveis (de decisão). A função objetivo irá identificar o objetivo da tomada de decisão. A função objetivo irá minimizar ou maximizar o objetivo (custo ou lucro, por exemplo). E finalmente, as restrições são os elementos ou condições que limitam o processo decisório.

As equações (A.1)-(A.5) consistem em um exemplo de programação linear na forma *standard*. A função objetivo é representada na equação (A.1) e as restrições nas equações (A.2)-(A.5). Na função objectivo o vetor c irá identificar o objetivo da tomada de decisão. Cada elemento c_i contem o custo

associado à ativação da variável de decisão x_i. Já a matriz a são os coeficientes das restrições e o vetor b é chamado de termo independente da restrições.

$$min : c_1 * x_1 + c_2 * x_2 + ... + c_n * x_n \tag{A.1}$$

$$a_{11} * x_1 + a_{12} * x_2 + ... + a_{1n} * x_n = b_1 \tag{A.2}$$

$$a_{21} * x_1 + a_{22} * x_2 + ... + a_{2n} * x_n = b_2 \tag{A.3}$$

$$a_{m1} * x_1 + a_{m2} * x_2 + ... + a_{mn} * x_n = b_m \tag{A.4}$$

$$x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0 \tag{A.5}$$

A.1.1 Programação Inteira

Programação inteira é uma classe de problemas de programação linear em que todas ou algumas das suas variáveis têm de assumir valores inteiros [37]. Se todas as variáveis do problema são inteiras, o problema é considerado um problema de Programação Inteira Pura. Se apenas parte das variáveis são inteiras, o problema é classificado como sendo de Programação Inteira Mista.

Existe ainda um caso especial de variáveis inteiras, as variáveis binárias. Uma variável binária são aquelas que podem receber somente valor zero ou um. Nesse caso, o problema é considerado como um problema de Programação Inteira Binária.

Segundo Alves [37], as variáveis binárias são muito úteis para representar situações dicotômicas, podendo desempenhar dois papéis distintos: como variáveis principais ou de decisão (decisões do tipo fazer ou não fazer, construir ou não construir, etc.); como variáveis auxiliares, sendo utilizadas para exprimir certas condições.

Os problemas do horário universitário são considerados um problema de Programação Inteira Binária. A variáveis de decisão que indicam se uma aula foi atribuída é uma variável binária, com valor um indicando que há aula atribuída e zero indicando que não há aula atribuída.

Problemas de programação inteira são geralmente mais difíceis de serem resolvidos quando comparados com os problemas de programação linear *standard* (com variáveis não inteiras). Ao indicar que variáveis são inteiras, na verdade o que está sendo feito é a inclusão de novas restrições ao problema. Novas restrições tornam o problema mais difícil de ser resolvido.

A literatura científica traz algumas soluções para conseguir contornar a dificuldade de resolução dos problemas de programação inteira entre as quais serão destacadas os métodos dos *Planos de Corte*, *Branch and Bound* e *Branch and Cut*.

A.1.2 Problema de Transporte

Problemas de transporte é uma classe de programação linear inteira. É originado da necessidade de transportar mercadorias. A sua formulação básica consiste em atender as demandas dos destinos (*demand*) através das capacidades das origens (*supply*) ao menor custo possível.

A.1.3 Problema de Atribuição

Um caso especial do problema de transporte são os problemas de atribuição.

Burkad [9] define os problemas de atribuição como aqueles problemas que lidam com a questão de como atribuir n itens (jobs, alunos) a n outros itens (máquinas, tarefas). Ou seja, atribuir um conjunto de tarefas a um conjunto de agentes com um custo de atribuição associado. É considerado como o Problema Generalizado de Atribuição (PGA). Nesse problema cada tarefa obrigatoriamente deve ser atribuída a um agente e cada agente só pode receber tarefas até o limite de sua capacidade. A associação entre tarefas e agentes deve ser de tal forma, que o custo total de atribuição seja minimizado (ou o lucro maximizado).

O PGA pode ser formulado matematicamente da seguinte forma [11]:

$$min : \sum_{i \in A} \sum_{j \in T} c_{i,j} * x_{i,j} \tag{A.6}$$

$$\sum_{j \in T} x_{i,j} \leq b_i \qquad \forall i \in A \tag{A.7}$$

$$\sum_{i \in A} x_{i,j} = 1 \qquad \forall j \in T \tag{A.8}$$

$$x_{i,j} \in \{0,1\} \qquad \forall i \in A, \forall j \in T \tag{A.9}$$

Nessa formulação, considerando que existe um conjunto A de agentes e um conjunto T de tarefas, $c_{i,j}$ representa o custo de atribuir a tarefa j ao agente i, b_i a capacidade do agente i e $x_{i,j}$ assume o valor de 1 se a tarefa j foi atribuída ao agente i ou 0, caso contrário.

A equação (A.6) é a função objetivo que procura minimizar os custos de atribuição da tarefa j ao agente i. A restrição (A.7) irá assegurar que a capacidade b_i do agente i não seja violada, a restrição (A.8) garante que toda tarefa j seja atribuída e a somente um agente i e a restrição (A.9) define as variáveis $x_{i,j}$ como binária.

Os problemas de atribuição ganharam bastante importância além da simples atribuição de tarefas entre agentes. Subtil [39] em seu trabalho cita outros contextos de utilização, contextos estes algumas vezes mais complexos e difíceis de resolver: atribuição de tarefas em redes de computadores, localização de facilidades, carregamento de contêineres, escalonamento de tarefas em máquinas paralelas, roteamento de veículos, comparação de estoque/documentos, escalonamento de recursos, escalonamento de multiprocessadores, etc.

O número de variáveis e restrições do PGA pode aumentar consideravelmente. Basta adicionar um novo agente ou uma nova tarefa que o número de variáveis e restrições aumentam, tornando a sua resolução mais difícil.

Uma extensão do PGA é o Problema de Atribuição Multidimensional (Multi-dimensional assignment problems MAP). Enquanto o PGA trata somente de duas dimensões (agente x tarefa), os problemas de atribuição multidimensional possuem três ou mais dimensões. Adicionar dimensões ao problema torna a sua resolução ainda mais difícil. Pode-se exemplificar adicionando a dimensão slot de tempo à formulação PGA apresentada anteriormente.

$$min : \sum_{i \in A} \sum_{j \in T} \sum_{z \in S} c_{i,j,z} * x_{i,j,z} \tag{A.10}$$

$$\sum_{j \in T} \sum_{z \in S} x_{i,j,z} \leq b_i \qquad \forall i \in A \tag{A.11}$$

$$\sum_{i \in A} \sum_{z \in S} x_{i,j,z} = 1 \qquad \forall j \in T \tag{A.12}$$

$$\sum_{j \in T} x_{i,j,z} \leq 1 \qquad \forall i \in A, \forall z \in S \tag{A.13}$$

$$x_{i,j,z} \in \{0,1\} \qquad \forall i \in A, \forall j \in T, \forall z \in S \tag{A.14}$$

Nessa formulação foi adicionado à formulação do PGA o conjunto S de slots de tempo. Dessa forma, $c_{i,j,z}$ representa o custo de atribuir a tarefa j ao agente i no horário z, b_i a capacidade do agente i e $x_{i,j,z}$ assume o valor de 1 se a tarefa j foi atribuída ao agente i no horário z ou 0, caso contrário. Para cada agente existe a limitação de executar uma tarefa a cada slot de tempo.

A equação (A.10) é a função objetivo que procura minimizar os custos de atribuição da tarefa j ao agente i no horário h. A restrição (A.11) irá assegurar que a capacidade $b_{i,z}$ do agente i no horário z não seja violada, a restrição (A.12) garante que toda tarefa j seja atribuída a somente um agente i e em um determinado horário, a restrição (A.13) garante a atribuição de somente uma tarefa i ao agente i em um determinado slot de tempo z e a restrição (A.14) define as variáveis $x_{i,j,z}$ como binária.

A restrição (A.13) foi incluída forma propositada para demonstrar que, ao adicionar novas dimensões ao problemas, podem surgir novas restrições. Ao mesmo tempo que adicionar novas dimensões aumenta o número de variáveis do problema. Enquanto que na formulação do PGA as variáveis de decisão estão contidas uma matriz bidimensional de agentes x tarefas, na solução do MAP a variável de decisão passa a ser uma matriz multidimensional de agentes x tarefas x slots de tempo. Como já foi referido anteriormente, adicionar novas variáveis e restrições torna o problema ainda mais difícil de resolução.

B. Tabela Simplex

A seguir é demonstrado passo a passo como resolver o método Simplex através da resolução matricial ou tabela Simplex. Considerando o seguinte problema extraído de Alves [37]:

$$\begin{aligned}
&\max z = 2400 * x_1 + 1500 * x_2 \\
&x_1 + x_2 \leq 6 \\
&9 * x_1 + 5 * x_2 \leq 45 \\
&x_1, x_2 \geq 0
\end{aligned}$$

Para resolver o Método Simplex é necessário converter o problema para a forma canônica. Isso significa, para o problema apresentado, transformar inequações ($Ax_1 \leq B$) em equações ($Ax_1 + s_1 = B$). Para cada inequação do tipo $\leq$ deve ser adicionado uma *variável de folga (slack)*. Também é realizada uma adaptação a função objetiva para que a mesma seja igual a zero. Após a conversão, o problema fica da seguinte forma:

$$\begin{aligned}
&\max z - 2400 * x_1 - 1500 * x_2 = 0 \\
&x_1 + x_2 + s_1 = 6 \\
&9 * x_1 + 5 * x_2 + s_2 = 45 \\
&x_1, x_2 \geq 0
\end{aligned}$$

Uma vez na forma canônica, o problema é então inserido na tabela Simplex (tabela B.1), onde cada coluna da tabela equivale a uma variável do problema e cada linha equivale a função objetiva e as restrições do problema. Os coeficientes de cada variável/restrição são os valores que ficam dentro da tabela. E finalmente a coluna B estão os valores dos termos independentes da função objetivo e das restrições. O termo independente é o valor que fica após o sinal de igual nas restrições.

	Base	z	x1	x2	s1	s2	B
1	z	1	-2400	-1500	0	0	0
2	s1	0	1	1	1	0	6
3	s2	0	9	5	0	1	45

Tabela B.1: Tabela Simplex - solução inicial

A linha 1 da tabela corresponde a função objetiva e as demais, as restrições do problema. Outra função das linhas da tabela é determinar as *variáveis básicas* do problema. As variáveis básicas são

aquelas que assumem valor para fornecer uma solução ao problema.

Inicialmente é atribuído valor zero para as variáveis x_1 e x_2. Então, s_1 e s_2, recebem valores 6 e 45, respectivamente, atendendo as restrições (que na forma canônica, são restrições de igualdade). As variáveis s_1 e s_2 são as primeiras variáveis básicas do problema. Na tabela Simplex, cada linha equivale a uma variável básica, indicado na primeira coluna e a coluna B os seus valores. As demais variáveis, que não estão na linha, tem o seu valor igual a zero e são chamadas de *variáveis não básicas*.

Portanto, na tabela B.1, é possível identificar que a solução inicial tem como variáveis básicas s_1 e s_2 e seus valores são 6 e 45. Ainda existe, na primeira linha, a variável z com valor zero. O valor de z é o valor da função objetiva.

O objetivo é fazer que as variáveis de decisão do problema (x_1 e x_2) passem a ser variáveis básicas. Para que uma variável não básica entre na base, ou seja, passe a ser uma variável básica, é necessário que uma variável básica saia da base. O método Simplex é responsável por realizar a troca.

Para determinar qual variável não básica deverá entrar na base deve-se escolher entre as variáveis com valores negativo na linha 1. Geralmente escolhe aquela variável que possui o maior valor absoluto. A variável com maior valor absoluto é a variável x_1, com valor -2400. Portanto, x_1 será a variável que irá entrar na base.

Para escolher a variável que irá sair da base, deve-se, para cada linha, dividir o valor de B pelo coeficiente de x_1. Aquela variável que possuir o menor valor positivo, será que a variável que irá sair da base (tabela B.2).

			Entra na base							
	Base	z	x1	x2	s1	s2	B	B'	B/B'	
1	z	1	-2400	-1500	0	0	0			
2	s1	0	1	1	1	0	6	1	6	
3	s2	0	9	5	0	1	45	9	5	Sai da base

Tabela B.2: Tabela Simplex - determinação das variáveis que entra e que sai da base - passo 1

A linha que sai da base é chamada de linha pivot (LP) e a sua intersecção com a coluna que entra na base é chamado de elemento pivot. A linha pivot será utilizada para determinar novos valores para as demais linhas. Mas antes é necessário que o elemento pivot tenha valor 1. No exemplo, o elemento pivot está com valor 9. Portanto é necessário dividir toda a linha pivot pelo valor do pivot e, assim o elemento pivot terá valor 1. A tabela B.3 demonstra a obtenção da nova linha pivot (NLP).

LP	0	9	5	0	1	45	/ 9
NLP	0	1	0,56	0	0,11	5	Nova Linha 3

Tabela B.3: Tabela Simplex - determinação da NLP - passo 1

Ao determinar a NLP, também é determinada os valores da nova linha 3. O próximo passo é determinar os valores para as demais linhas da tabela. Para isso é necessário multiplicar cada elemento da NLP pelo valor do elemento da coluna que sai de cada linha, multiplicado por -1. A esse valor, soma-se os valores da linha. A tabela B.4 demonstra a obtenção dos novos valores das linhas.

Uma vez calculado os novos valores das linhas, uma nova tabela Simplex é gerada (tabela B.5).

NLP	0	1	0,56	0	0,11	5
* 2400	0	2400	1333,33	0	266,67	12000
+ Linha 1	1	-2400	-1500	0	0	0
Nova linha 1	1	0	-166,67	0	266,67	12000

NLP	0	1	0,56	0	0,11	5
* -1	0	-1	-0,56	0	-0,11	-5
+ Linha 2	0	1	1	1	0	6
Nova linha 2	0	0	0,44	2	-0,11	1

Tabela B.4: Tabela Simplex - determinação das novas linhas - passo 1

	Base	z	x1	x2	s1	s2	B
1	z	1	0	-166,67	0	266,67	12000
2	s1	0	0	0,44	1	-0,11	1
3	x1	0	1	0,56	0	0,11	5

Tabela B.5: Tabela Simplex - após passo 1

Uma nova solução viável é encontrada com as variáveis $x_1 = 5$ e $s_1 = 1$ o valor da função objetiva com valor de 12000. Como na primeira linha ainda existe um valor negativo ($x_2 = -166,67$), uma nova troca deve ser realizada (tabela B.6). Nesse segundo passo, a linha 2 será a linha pivot.

	Base	z	x1	x2 (Entra na base)	s1	s2	B	B'	B/B'	
1	z	1	0	-166,67	0	0	12000			
2	s1	0	0	0,44	1	-0,11	1	0,44	2,25	Sai da base
3	x1	0	1	0,56	0	0,11	5	0,56	9	

Tabela B.6: Tabela Simplex - determinação das variáveis que entra e que sai da base - passo 2

A tabela B.7 demonstra o cálculo na NLP do passo 2 e a tabela B.8 demonstra o cálculo das novas linhas do passo 2. E, finalmente, uma nova tabela Simplex é gerada (tabela B.9), com os novos valores de cada linha. Uma nova solução viável é encontrada, $x_1 = 3,75$ e $x_2 = 2,25$ e o valor da função objetiva é de 12375. Como na primeira linha não existem mais valores negativos, essa solução é considerada a solução ótima e o Simplex é encerrado.

LP	0	0	0,44	1	-0,11	1	/ 0,44
NLP	0	0	1	2,25	-0,25	2,25	Nova linha 2

Tabela B.7: Tabela Simplex - determinação da NLP - passo 2

NLP	0	0	1	2,25	-0,25	2,25
* 166,67	0	0	166,67	375	-41,67	375
+ Linha 1	1	0	-166,67	0	266,67	12000
Nova linha 1	1	0	0	375	225	12375

NLP	0	0	1	2,25	-0,25	2,25
* -0,56	0	0	-0,56	-1,25	0,14	-1,25
+ Linha 3	0	1	0,56	0	0,11	5
Nova linha 3	0	1	0	-1,25	0,25	3,75

Tabela B.8: Tabela Simplex - determinação das novas linhas - passo 2

	Base	z	x1	x2	s1	s2	B
1	z	1	0	0	375	225	12375
2	x2	0	0	1	2,25	-0,25	2,25
3	x1	0	1	0	-1,25	0,25	3,75

Tabela B.9: Tabela Simplex - após passo 2

C. Branch and Bound

O método *Branch and Bound* foi criado com o objetivo de resolver problema de programação inteira. Segundo Alves [37], consiste na ramificação sucessiva do conjunto de soluções possíveis do problema em subconjuntos e na limitação do valor ótimo da função objetiva de modo a excluir os subconjuntos que não contenham a solução ótima.

O problema é executado como um problema de programação linear *standard*, ou seja, sem as restrições de variáveis inteiras. Esse procedimento é considerado uma relaxação linear. Uma premissa deve ser considerada: **se, na solução ótima da relaxação linear, os valores das variáveis forem todos inteiros, então a solução encontrada é a solução ótima do problema de programação inteira**. Então, ao relaxar, se o resultado apresentar somente valores inteiros, então é encontrada a solução ótima, caso contrário, o problema é dividido em dois subproblemas, adicionando restrições adicionais a cada um dos subproblemas.

Cada subproblema é executado com relaxação linear e sempre que os resultados não forem valores inteiros, cada subproblema é novamente dividido em dois novos subproblemas até que todos subproblemas tenham soluções inteiras ou que não tenha soluções factíveis.

A seguir o método é demonstrado em detalhes. Considerando o seguinte problema de programação inteira extraído de Alves [37]:

$$\begin{aligned}
&\max F(x) = 2400 * x_1 + 1500 * x_2 \\
&x_1 + x_2 \leq 6 \\
&9 * x_1 + 5 * x_2 \leq 45 \\
&x_1, x_2 \geq 0 \text{ e inteiros}
\end{aligned}$$

O primeiro passo consiste em resolver o relaxamento linear do problema. A solução ótima obtida é: F(3.75, 2.25) = 12375. O valor da função já define o limite máximo que o problema pode resultar, $0 \leq F(x) \leq 12375$. E como há variáveis com valor não inteiro, é necessário realizar a divisão em dois novos subproblemas, A e B.

Para realizar a divisão, deve-se escolher uma variável com valor não inteira. No primeiro relaxamento linear executado, ambas as variáveis do problema resultaram em valor não inteiro. Então escolhe uma delas para ser utilizada nos subproblemas. Optando pela variável $x_1 = 3,75$, são introduzidas novas restrições de eliminação de soluções não inteiras, $x_1 \leq 3$ e $x_1 \geq 4$ e são criados os subproblemas A e B.

$$\text{A: } \max F(x) = 2400 * x_1 + 1500 * x_2$$

$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$
$$x_1 \leq 3$$
$$x_1, x_2 \geq 0$$

B: max $F(x) = 2400 * x_1 + 1500 * x_2$
$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$
$$x_1 \geq 4$$
$$x_1, x_2 \geq 0$$

O subproblema A tem como solução ótima F(3, 3) = 11700 e o subproblema B, F(4, 1.8) = 12300. Por ter uma solução ótima inteira, o subproblema A não é mais dividido e estabelece um novo valor limite mínimo para o problema, $11700 \leq F(x) \leq 12375$.

Já o subproblema B, cuja solução ótima é não inteiro e o seu valor está compreendido entre os limites mínimo e máximo de F(x) ($11700 \leq 12300 \leq 12375$), será dividido em dois novos subproblemas, B1 e B2, com a inclusão das seguintes restrições: $x_2 \geq 2$ e $x_2 \leq 1$. O subproblema B também estabelece um novo máximo para a função objetiva $11700 \leq F(x) \leq 12300$.

B1: max $F(x) = 2400 * x_1 + 1500 * x_2$
$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$
$$x_1 \geq 4$$
$$x_2 \geq 2$$
$$x_1, x_2 \geq 0$$

B2: max $F(x) = 2400 * x_1 + 1500 * x_2$
$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$
$$x_1 \geq 4$$
$$x_2 \leq 1$$
$$x_1, x_2 \geq 0$$

O subproblema B1 não é solucionável (não possui solução factível), portanto é descartado. Já o subproblema B2 tem com solução ótima F(4.4, 1) = 12167. Por não possuir uma solução inteira e o seu resultado estar compreendido entre os limites mínimo e o máximo de F(x) ($11700 \leq 12167 \leq 12300$) o mesmo deve ser dividido em dois novos subproblemas, B21 e B22, adicionando as restrições $x_1 \leq 4$ e $x_1 \geq 5$

B21: max $F(x) = 2400 * x_1 + 1500 * x_2$
$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$

$$x_1 \geq 4$$
$$x_2 \leq 1$$
$$x_1 \leq 4$$
$$x_1, x_2 \geq 0$$

B21: max $F(x) = 2400 * x_1 + 1500 * x_2$
$$x_1 + x_2 \leq 6$$
$$9 * x_1 + 5 * x_2 \leq 45$$
$$x_1 \geq 4$$
$$x_2 \leq 1$$
$$x_1 \geq 5$$
$$x_1, x_2 \geq 0$$

O subproblema B21 tem como solução ótima F(4,1) = 11100. Apesar de ser uma solução inteira, o valor do subproblema B21 (11000) é inferior ao limite mínimo (11700), portanto é descartado.

Já o subproblema B22 tem como solução ótima F(5,0) = 12000. Uma solução inteira e o seu valor está compreendido entre os limites mínimo e máximo ($11700 \leq 12000 \leq 12300$) então é uma solução válida para o problema. Pelo fato do resultado de B22 ser uma solução inteira, o mesmo não é mais subdivido e como não há mais subproblemas a serem divididos, concluí-se que o resultado de B22 é a solução ótima para o problema original.

A árvore Branch and Bound completa do exemplo é demonstrado na figura C.1.

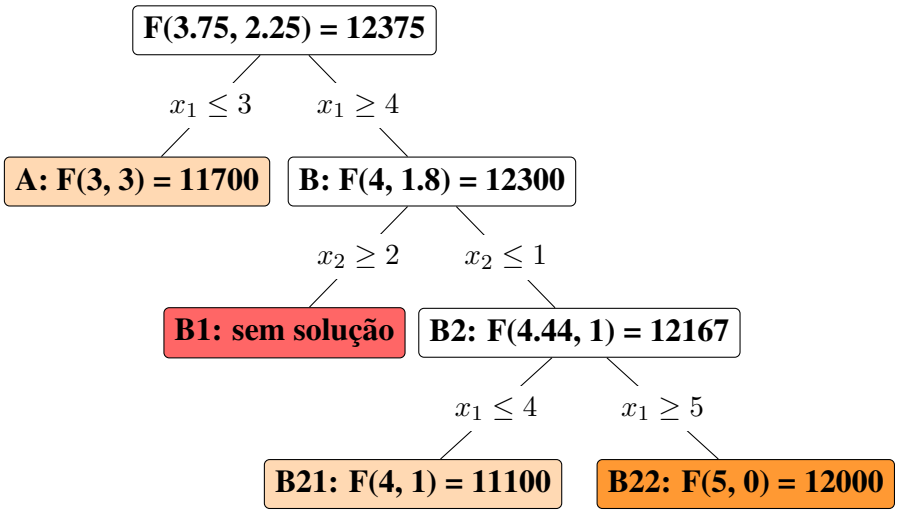

Figura C.1: Árvore Branch and Bound do exemplo apresentado

D. Código fonte solução básica

```
from mip import *

#Valores de entrada
cadeira = ['cadeira_1','cadeira_2','cadeira_3','cadeira_4','cadeira_5',
           'cadeira_6']
sala_aula = ['sala_1','sala_2','sala_3','sala_4']
dia_semana = ['segunda', 'terca', 'quarta', 'quinta', 'sexta']
hora_aula = ['horario_1','horario_2','horario_3','horario_4','horario_5',
           'horario_6']
turma = ['turma_1','turma_2']
cadeira_turma = [['cadeira_1','cadeira_2','cadeira_3'],
                 ['cadeira_4','cadeira_5','cadeira_6']]
professor = ['professor_1','professor_2','professor_3','professor_4']
cadeira_professor = [['cadeira_1'],
                     ['cadeira_2'],
                     ['cadeira_3','cadeira_4'],
                     ['cadeira_5','cadeira_6']]
ch = [3,2,4,4,3,2]

#custo de atribuição
ca = [[[[1 for h in hora_aula]
           for d in dia_semana]
           for s in sala_aula]
           for c in cadeira]

#obtem os totais dos conjuntos de entrada
C = range(len(cadeira))
S = range(len(sala_aula))
D = range(len(dia_semana))
H = range(len(hora_aula))
T = range(len(turma))
P = range(len(professor))

#cria um model do MIP
m = Model()

# Variável binária decisória que indica as aulas atribuídas
```

```
x = [[[[m.add_var(name='x_{}_{}_{}_{}'.format(c, s, d, h), var_type=BINARY)
        for h in hora_aula]
        for d in dia_semana]
        for s in sala_aula]
        for c in cadeira]

# HC01 Garantir o cumprimento da carga horária das cadeiras.
for c in C:
        m.add_constr(xsum(x[c][s][d][h] for s in S
                                        for d in D
                                        for h in H)
                     == ch[c])

# HC02 Garantir apenas uma aula seja atribuída para uma determinada sala,
# dia da semana e horário (conflito das salas).
for s in S:
        for d in D:
                for h in H:
                        m.add_constr(xsum(x[c][s][d][h] for c in C) <= 1)

# HC03 Garantir que cada professor tenha apenas uma aula atribuída para um
# determinado dia de semana e horário (conflito dos professores).
for cp in cadeira_professor:
        for d in D:
                for h in H:
                        m.add_constr(xsum(x[cadeira.index(c)][s][d][h]
                                          for c in cp
                                          for s in S)
                                     <= 1)

# HC04 Garantir que cada turma tenha apenas uma aula atribuída para um
# determinado dia de semana e horário (conflito dos alunos).
for ct in cadeira_turma:
        for d in D:
                for h in H:
                        m.add_constr(xsum(x[cadeira.index(c)][s][d][h]
                                          for c in ct
                                          for s in S) <= 1)

#define a função objetivo
m.objective = minimize(ca[c][s][d][h] * xsum(x[c][s][d][h]
                       for c in C
                       for s in S
                       for d in D
                       for h in H))

#executa o modelo
```

```
85 m.optimize()
86
87 #exibe o resultado
88 for v in m.vars:
89         if abs(v.x) > 1e-6: # only printing non-zeros
90                 print("{};{}".format(v.name, v.x))
```

E. Resultados - Caso de estudo UHLT

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00	Matemática I	Mat Discreta	Mat Discreta	Fund Física	Sist Digitais
14 30	Matemática I	Mat Discreta	Mat Discreta	Fund Física	Sist Digitais
15 00	Matemática I	Mat Discreta	Mat Discreta	Fund Física	Sist Digitais
15 30	Matemática I	Mat Discreta	Mat Discreta	Fund Física	Fund Física
16 00	Sist Digitais	Fund Program	Fund Program	Matemática I	Fund Física
16 30	Sist Digitais	Fund Program	Fund Program	Matemática I	Fund Física
17 00	Sist Digitais	Fund Program	Fund Program	Matemática I	Fund Física
17 30	Sist Digitais	Fund Program	Fund Program	Matemática I	

Tabela E.1: LEI ANO 1 Turma 1

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00	Matemática I	Mat Discreta	Matemática I	Sist Digitais	Sist Digitais
14 30	Matemática I	Mat Discreta	Matemática I	Sist Digitais	Sist Digitais
15 00	Matemática I	Mat Discreta	Matemática I	Sist Digitais	Sist Digitais
15 30	Matemática I	Mat Discreta	Matemática I	Sist Digitais	Fund Física
16 00	Fund Program	Fund Program	Fund Física	Mat Discreta	Fund Física
16 30	Fund Program	Fund Program	Fund Física	Mat Discreta	Fund Física
17 00	Fund Program	Fund Program	Fund Física	Mat Discreta	Fund Física
17 30	Fund Program	Fund Program	Fund Física	Mat Discreta	

Tabela E.2: LEI ANO 1 Turma 2

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00	Fund Program	Fund Program	Fund Física	Mat Discreta	Sist Digitais
14 30	Fund Program	Fund Program	Fund Física	Mat Discreta	Sist Digitais
15 00	Fund Program	Fund Program	Fund Física	Mat Discreta	Sist Digitais
15 30	Fund Program	Fund Program	Fund Física	Mat Discreta	Sist Digitais
16 00	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Matemática I
16 30	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Matemática I
17 00	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Matemática I
17 30		Matemática I	Mat Discreta	Fund Física	Matemática I

Tabela E.3: LEI ANO 1 Turma 3

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00	Matemática I	Fund Program	Sist Digitais	Fund Program	Fund Física
14 30	Matemática I	Fund Program	Sist Digitais	Fund Program	Fund Física
15 00	Matemática I	Fund Program	Sist Digitais	Fund Program	Fund Física
15 30	Matemática I	Fund Program	Sist Digitais	Fund Program	Fund Física
16 00	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Mat Discreta
16 30	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Mat Discreta
17 00	Sist Digitais	Matemática I	Mat Discreta	Fund Física	Mat Discreta
17 30		Matemática I	Mat Discreta	Fund Física	Mat Discreta

Tabela E.4: LEI ANO 1 Turma 4

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00	Matemática I	Mat Discreta	Fund Program	Fund Física	Sist Digitais
14 30	Matemática I	Mat Discreta	Fund Program	Fund Física	Sist Digitais
15 00	Matemática I	Mat Discreta	Fund Program	Fund Física	Sist Digitais
15 30	Matemática I	Mat Discreta	Fund Program	Fund Física	Fund Física
16 00	Mat Discreta	Fund Program	Matemática I	Sist Digitais	Fund Física
16 30	Mat Discreta	Fund Program	Matemática I	Sist Digitais	Fund Física
17 00	Mat Discreta	Fund Program	Matemática I	Sist Digitais	Fund Física
17 30	Mat Discreta	Fund Program	Matemática I	Sist Digitais	

Tabela E.5: LEIRT ANO 1

Horário	Segunda	Terça	Quarta	Quinta	Sexta
14 00		Fund Program		Matemática I	
14 30		Fund Program		Matemática I	
15 00	Contabilidade	Fund Program	Marketing	Matemática I	Contabilidade
15 30	Contabilidade	Fund Program	Marketing	Matemática I	Contabilidade
16 00	Contabilidade	Matemática I	Marketing	Fund Program	Contabilidade
16 30	Marketing	Matemática I	Sist de Inf	Fund Program	Sist de Inf
17 00	Marketing	Matemática I	Sist de Inf	Fund Program	Sist de Inf
17 30	Marketing	Matemática I	Sist de Inf	Fund Program	Sist de Inf

Tabela E.6: LIG ANO 1

Horário	Segunda	Terça	Quarta	Quinta	Sexta
18 00	Matemática I	Sist Digitais	Mat Discreta	Fund Program	Fund Program
18 30	Matemática I	Sist Digitais	Mat Discreta	Fund Program	Fund Program
19 00	Matemática I	Sist Digitais	Mat Discreta	Fund Program	Fund Program
19 30	Matemática I	Sist Digitais	Mat Discreta	Fund Program	Fund Program
20 00	Fund Física	Fund Física	Sist Digitais	Mat Discreta	Matemática I
20 30	Fund Física	Fund Física	Sist Digitais	Mat Discreta	Matemática I
21 00	Fund Física	Fund Física	Sist Digitais	Mat Discreta	Matemática I
21 30	Fund Física	Fund Física		Mat Discreta	Matemática I

Tabela E.7: LEI/LEIRT ANO 1 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
15 00	Contabilidade		Marketing		Contabilidade
15 30	Contabilidade		Marketing		Contabilidade
16 00	Contabilidade		Marketing		Contabilidade
16 30	Marketing		Sist de Inf		Sist de Inf
17 00	Marketing		Sist de Inf		Sist de Inf
17 30	Marketing		Sist de Inf		Sist de Inf
18 00	Matemática I			Fund Program	Fund Program
18 30	Matemática I			Fund Program	Fund Program
19 00	Matemática I			Fund Program	Fund Program
19 30	Matemática I			Fund Program	Fund Program
20 00					Matemática I
20 30					Matemática I
21 00					Matemática I
21 30					Matemática I

Tabela E.8: LIG ANO 1 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30		Sist Operativos			Arq Avanç Comp
9 00		Sist Operativos	Sist Operativos		Arq Avanç Comp
9 30	Bases Dados	Sist Operativos	Sist Operativos		Arq Avanç Comp
10 00	Bases Dados	Sist Operativos	Sist Operativos		Arq Avanç Comp
10 30	Bases Dados	Sinais e Sist	Sist Operativos	Sinais e Sist	Bases Dados
11 00	Bases Dados	Sinais e Sist	LP II	Sinais e Sist	Bases Dados
11 30	LP II	Sinais e Sist	LP II	Sinais e Sist	Bases Dados
12 00	LP II	Sinais e Sist	LP II	Arq Avanç Comp	Bases Dados
12 30	LP II		LP II	Arq Avanç Comp	
13 00	LP II			Arq Avanç Comp	
13 30				Arq Avanç Comp	

Tabela E.9: LEI ANO 2 Turma 1

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30				Bases Dados	Sist Operativos
9 00			Sist Operativos	Bases Dados	Sist Operativos
9 30	LP II		Sist Operativos	Bases Dados	Sist Operativos
10 00	LP II		Sist Operativos	Bases Dados	Sist Operativos
10 30	LP II		Sist Operativos	Sinais e Sist	Bases Dados
11 00	LP II	Arq Avanç Comp	LP II	Sinais e Sist	Bases Dados
11 30	Sinais e Sist	Arq Avanç Comp	LP II	Sinais e Sist	Bases Dados
12 00	Sinais e Sist	Arq Avanç Comp	LP II	Arq Avanç Comp	Bases Dados
12 30	Sinais e Sist	Arq Avanç Comp	LP II	Arq Avanç Comp	
13 00	Sinais e Sist			Arq Avanç Comp	
13 30				Arq Avanç Comp	

Tabela E.10: LEI ANO 2 Turma 2

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 00					
8 30				Sist Operativos	
9 00			Sinais e Sist	Sist Operativos	
9 30			Sinais e Sist	Sist Operativos	
10 00	Sist Operativos	LP II	Sinais e Sist	Sist Operativos	
10 30	Sist Operativos	LP II	Sinais e Sist	Sinais e Sist	Bases Dados
11 00	Sist Operativos	LP II	LP II	Sinais e Sist	Bases Dados
11 30	Sist Operativos	LP II	LP II	Sinais e Sist	Bases Dados
12 00	Arq Avanç Comp	Bases Dados	LP II	Arq Avanç Comp	Bases Dados
12 30	Arq Avanç Comp	Bases Dados	LP II	Arq Avanç Comp	
13 00	Arq Avanç Comp	Bases Dados		Arq Avanç Comp	
13 30	Arq Avanç Comp	Bases Dados		Arq Avanç Comp	

Tabela E.11: LEI ANO 2 Turma 3

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30		Bases Dados			LP II
9 00		Bases Dados	Sist Operativos		LP II
9 30		Bases Dados	Sist Operativos		LP II
10 00	Sist Operativos	Bases Dados	Sist Operativos		LP II
10 30	Sist Operativos	Sinais e Sist	Sist Operativos	Sinais e Sist	Bases Dados
11 00	Sist Operativos	Sinais e Sist	LP II	Sinais e Sist	Bases Dados
11 30	Sist Operativos	Sinais e Sist	LP II	Sinais e Sist	Bases Dados
12 00	Fund Telecom	Sinais e Sist	LP II	Fund Telecom	Bases Dados
12 30	Fund Telecom	Fund Telecom	LP II	Fund Telecom	
13 00	Fund Telecom	Fund Telecom		Fund Telecom	
13 30	Fund Telecom			Fund Telecom	

Tabela E.12: LEIRT ANO 2

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30		Bases Dados			LP II
9 00		Bases Dados	Sist Operativos		LP II
9 30		Bases Dados	Sist Operativos		LP II
10 00	Sist Operativos	Bases Dados	Sist Operativos		LP II
10 30	Sist Operativos	Gestão	Sist Operativos		Bases Dados
11 00	Sist Operativos	Gestão	LP II	Invest Operac	Bases Dados
11 30	Sist Operativos	Gestão	LP II	Invest Operac	Bases Dados
12 00		Invest Operac	LP II	Invest Operac	Bases Dados
12 30		Invest Operac	LP II	Gestão	
13 00		Invest Operac		Gestão	
13 30				Gestão	

Tabela E.13: LIG ANO 2

Horário	Segunda	Terça	Quarta	Quinta	Sexta
18 00	Sist Operativos				LP II
18 30	Sist Operativos	Bases Dados		Arq Avanç Comp	LP II
19 00	Sist Operativos	Bases Dados	Sinais e Sist	Arq Avanç Comp	LP II
19 30	Sist Operativos	Bases Dados	Sinais e Sist	Arq Avanç Comp	LP II
20 00	Sinais e Sist	Bases Dados	Sinais e Sist	Arq Avanç Comp	Bases Dados
20 30	Sinais e Sist	Arq Avanç Comp	Sist Operativos	LP II	Bases Dados
21 00	Sinais e Sist	Arq Avanç Comp	Sist Operativos	LP II	Bases Dados
21 30	Sinais e Sist	Arq Avanç Comp	Sist Operativos	LP II	Bases Dados
22 00			Sist Operativos	LP II	

Tabela E.14: LEI ANO 2 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
12 00	Fund Telecom			Fund Telecom	
12 30	Fund Telecom	Fund Telecom		Fund Telecom	
13 00	Fund Telecom	Fund Telecom		Fund Telecom	
13 30	Fund Telecom			Fund Telecom	
14 00					
14 30					
15 00					
15 30					
16 00					
16 30					
17 00					
17 30					
18 00	Sist Operativos			LP II	LP II
18 30	Sist Operativos			LP II	LP II
19 00	Sist Operativos		Sinais e Sist	LP II	LP II
19 30	Sist Operativos		Sinais e Sist	LP II	LP II
20 00	Sinais e Sist		Sinais e Sist	Bases Dados	Bases Dados
20 30	Sinais e Sist		Sist Operativos	Bases Dados	Bases Dados
21 00	Sinais e Sist		Sist Operativos	Bases Dados	Bases Dados
21 30	Sinais e Sist		Sist Operativos	Bases Dados	Bases Dados
22 00			Sist Operativos		

Tabela E.15: LEIRT ANO 2 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
10 30		Gestão			
11 00		Gestão		Invest Operac	
11 30		Gestão		Invest Operac	
12 00		Invest Operac		Invest Operac	
12 30		Invest Operac		Gestão	
13 00		Invest Operac		Gestão	
13 30				Gestão	
14 00					
14 30					
15 00					
15 30					
16 00					
16 30					
17 00					
17 30					
18 00	Sist Operativos			LP II	LP II
18 30	Sist Operativos			LP II	LP II
19 00	Sist Operativos			LP II	LP II
19 30	Sist Operativos			LP II	LP II
20 00				Bases Dados	Bases Dados
20 30			Sist Operativos	Bases Dados	Bases Dados
21 00			Sist Operativos	Bases Dados	Bases Dados
21 30			Sist Operativos	Bases Dados	Bases Dados
22 00			Sist Operativos		

Tabela E.16: LIG ANO 2 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 00					
8 30	Comp Dist	TFC 07	Compiladores		
9 00	Comp Dist	Eng Software	Compiladores		
9 30	Comp Dist	Eng Software	Compiladores		
10 00	Comp Dist	Eng Software	Compiladores		
10 30	TFC 03	Eng Software	Comp Dist	IHM	SIM
11 00	TFC 04	SIM	Comp Dist	IHM	SIM
11 30	TFC 02	SIM	Comp Dist	IHM	SIM
12 00	TFC 06	SIM	TFC 05	Eng Software	SIM
12 30	TFC 01	IHM		Eng Software	Compiladores
13 00		IHM		Eng Software	Compiladores
13 30		IHM		Eng Software	Compiladores
14 00					
14 30					
15 00					
15 30					
16 00					
16 30					
17 00					
17 30					
18 00					Seminários
18 30					Seminários
19 00					Seminários
19 30					Seminários

Tabela E.17: LEI ANO 3 Turma 1

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 00					
8 30		TFC 07	Compiladores	Eng Software	SIM
9 00		Eng Software	Compiladores	Eng Software	SIM
9 30		Eng Software	Compiladores	Eng Software	SIM
10 00		Eng Software	Compiladores	Eng Software	SIM
10 30	TFC 03	Eng Software	Comp Dist		Comp Dist
11 00	TFC 04	SIM	Comp Dist		Comp Dist
11 30	TFC 02	SIM	Comp Dist		Comp Dist
12 00	TFC 06	SIM	TFC 05		Comp Dist
12 30	TFC 01	IHM	IHM		Compiladores
13 00		IHM	IHM		Compiladores
13 30		IHM	IHM		Compiladores
14 00					
14 30					
15 00					
15 30					
16 00					
16 30					
17 00					
17 30					
18 00					Seminários
18 30					Seminários
19 00					Seminários
19 30					Seminários

Tabela E.18: LEI ANO 3 Turma 2

Horário	Segunda	Terça	Quarta	Quinta	Sexta
18 00	SIM	Eng Software		SIM	Seminários
18 30	SIM	Eng Software		SIM	Seminários
19 00	SIM	Eng Software	TFC 10	SIM	Seminários
19 30	SIM	Eng Software	TFC 09	Eng Software	Seminários
20 00	Comp Dist	Comp Dist	Compiladores	Eng Software	IHM
20 30	Comp Dist	Comp Dist	Compiladores	Eng Software	IHM
21 00	Comp Dist	Comp Dist	Compiladores	Eng Software	IHM
21 30		Comp Dist	Compiladores	IHM	Compiladores
22 00		TFC 11	TFC 08	IHM	Compiladores
22 30				IHM	Compiladores

Tabela E.19: LEI ANO 3 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30	Comp Dist				Proj Telec
9 00	Comp Dist	Eng Software			Proj Telec
9 30	Comp Dist	Eng Software			Proj Telec
10 00	Comp Dist	Eng Software			Proj Telec
10 30	Complem Redes	Eng Software	Comp Dist		SIM
11 00	Complem Redes	SIM	Comp Dist		SIM
11 30	Complem Redes	SIM	Comp Dist		SIM
12 00	Proj Telec	SIM	Complem Redes	Eng Software	SIM
12 30	Proj Telec		Complem Redes	Eng Software	
13 00	Proj Telec		Complem Redes	Eng Software	
13 30	Proj Telec		Complem Redes	Eng Software	

Tabela E.20: LEIRT ANO 3

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30			Sistemas Móveis	Eng Software	
9 00	TFC LIG 02	Eng Software	Sistemas Móveis	Eng Software	
9 30	Data Mining	Eng Software	Sistemas Móveis	Eng Software	
10 00	Data Mining	Eng Software	Sistemas Móveis	Eng Software	
10 30	Data Mining	Eng Software	Data Mining	SIM	
11 00	Data Mining	SIM	Data Mining	SIM	
11 30	TFC LIG 01	SIM	Data Mining	SIM	
12 00	Sistemas Móveis	SIM	Data Mining	SIM	
12 30	Sistemas Móveis	IHM	IHM	TFC LIG 03	
13 00	Sistemas Móveis	IHM	IHM	TFC LIG 04	
13 30	Sistemas Móveis	IHM	IHM	TFC LIG 05	
14 00					
14 30					
15 00					
15 30					
16 00					
16 30					
17 00					
17 30					
18 00					Seminários
18 30					Seminários
19 00					Seminários
19 30					Seminários

Tabela E.21: LIG ANO 3

Horário	Segunda	Terça	Quarta	Quinta	Sexta
8 30 9 00					Proj Telec
9 00 9 30					Proj Telec
9 30 10 00					Proj Telec
10 00 10 30					Proj Telec
10 30 11 00	Complem Redes				
11 00 11 30	Complem Redes				
11 30 12 00	Complem Redes				
12 00 12 30	Proj Telec		Complem Redes		
12 30 13 00	Proj Telec		Complem Redes		
13 00 13 30	Proj Telec		Complem Redes		
13 30 14 00	Proj Telec		Complem Redes		
14 00 14 30					
14 30 15 00					
15 00 15 30					
15 30 16 00					
16 00 16 30					
16 30 17 00					
17 00 17 30					
17 30 18 00					
18 00 18 30	SIM	Eng Software		SIM	
18 30 19 00	SIM	Eng Software		SIM	
19 00 19 30	SIM	Eng Software		SIM	
19 30 20 00	SIM	Eng Software		Eng Software	
20 00 20 30	Comp Dist	Comp Dist		Eng Software	
20 30 21 00	Comp Dist	Comp Dist		Eng Software	
21 00 21 30	Comp Dist	Comp Dist		Eng Software	
21 30 22 00		Comp Dist			

Tabela E.22: LEIRT ANO 3 Noturno

Horário	Segunda	Terça	Quarta	Quinta	Sexta
18 00	Sistemas Móveis	Eng Software		SIM	Seminários
18 30	Sistemas Móveis	Eng Software	Data Mining	SIM	Seminários
19 00	Sistemas Móveis	Eng Software	Data Mining	SIM	Seminários
19 30	Sistemas Móveis	Eng Software	Data Mining	Eng Software	Seminários
20 00	TFC LIG 05	Data Mining	Data Mining	Eng Software	IHM
20 30	TFC LIG 04	Data Mining	SIM	Eng Software	IHM
21 00	TFC LIG 06	Data Mining	SIM	Eng Software	IHM
21 30		Data Mining	SIM	IHM	
22 00			SIM	IHM	
22 30				IHM	

Tabela E.23: LIG ANO 3 Noturno