UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS - FANAT
DEPARTAMENTO DE INFORMÁTICA - DI

JOÃO LUCAS DA SILVA SANTOS

APLICAÇÃO DA METAHEURÍSTICA GRASP NA SOLUÇÃO DO PROBLEMA DE AGENDAMENTO DE HORÁRIO DE AULAS UNIVERSITÁRIAS

MOSSORÓ – RN

2024

JOÃO LUCAS DA SILVA SANTOS

APLICAÇÃO DA METAHEURÍSTICA GRASP NA SOLUÇÃO DO PROBLEMA DE AGENDAMENTO DE HORÁRIO DE AULAS UNIVERSITÁRIAS

Monografia apresentada à Universidade do Estado do Rio Grande do Norte, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação, sob orientação do Prof. Dr. Francisco Chagas de Lima Júnior.

MOSSORÓ – RN

2024

JOÃO LUCAS DA SILVA SANTOS

APLICAÇÃO DA METAHEURÍSTICA GRASP NA SOLUÇÃO DO PROBLEMA DE AGENDAMENTO DE HORÁRIO DE AULAS UNIVERSITÁRIAS

Monografia apresentada à Universidade do Estado do Rio Grande do Norte, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Dr. Francisco Chagas de Lima Júnior.

Aprovado em:_____/_____/_____.

BANCA EXAMINADORA

_____
Dr. Francisco Chagas de Lima Júnior
Universidade do Estado do Rio Grande do Norte - UERN

_____
Prof. Dr. Carlos Heitor Pereira Liberalino
Universidade do Estado do Rio Grande do Norte - UERN

_____
Prof. Dr. Sebastião Emidio Alves Filho
Universidade do Estado do Rio Grande do Norte - UERN

Dedico este trabalho à minha mãe, Lidiane, ao meu pai, Lindaci, por toda compreensão e incentivo ao longo dessa jornada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me guiou e me sustentou para que não desistisse.

Agradeço aos meus pais, Lidiane e Lindaci, por todo apoio, paciência e dedicação.

Agradeço a minha namorada, Yáscara, por todo apoio, por ter tido paciência de me escutar falando dos trabalhos acadêmicos.

Agradeço a todos os professores do DI, pela dedicação, paciência e disposição para com os alunos do curso Ciência da Computação.

Agradeço ao meu orientador, Prof. Dr. Francisco Chagas de Lima Júnior, por ter aceitado o convite para me orientar neste trabalho, por compartilhar o seu conhecimento e disponibilizar o seu tempo.

Agradeço a todos que contribuíram com esse trabalho diretamente ou indiretamente, muito obrigado

“Tudo Posso Naquele que me fortalece”

Filipenses 4:13

RESUMO

O Problema do Agendamento de Horários de Aulas, também conhecido como "Timetabling Problem" ou “University Course Timetabling Problem” (UCTP), é um dos frequentes desafios de otimização combinatória em instituições de ensino superior e requer alocação eficiente de horários para disciplinas, tendo em vista diversas restrições. O presente problema é classificado como NP-Completo e NP-Difícil, sendo, tradicionalmente, abordado por métodos heurísticos, GRASP, Simulated annealing, com o objetivo de obter soluções de alta qualidade em um curto período de tempo computacional. O presente estudo tem como objetivo a automatização do agendamento de horários de aulas no contexto universitário, utilizando o Algoritmo GRASP e considerando as restrições da Competição Internacional de Horários do ano 2007 (ITC-2007), de forma a lidar com limitações como a disponibilidade de professores, a carga horária das disciplinas e a capacidade das salas de aula, de forma a aprimorar a eficiência e equidade do serviço oferecido à comunidade acadêmica. A solução apresentada foi aplicada às instâncias fornecidas pelo ITC-2007.

Palavras-chave: Problema do Agendamento de Horários de Aulas, GRASP , Otimização Combinatória, ITC-2007

ABSTRACT

The Class Schedule Scheduling Problem, also known as the "Timetabling Problem" or “University Course Timetabling Problem” (UCTP), is one of the frequent challenges of combinatorial optimization in higher education institutions and requires efficient allocation of timetables for subjects, having in view of several restrictions. The present problem is classified as NP-Complete and NP-Hard, and is traditionally approached by heuristic methods, GRASP, Simulated annealing, with the objective of obtaining high-quality solutions in a short period of computational time. The present study aims to automate the scheduling of class times in the university context, using the GRASP Algorithm and considering the restrictions of the International Timetable Competition of the year 2007 (ITC-2007), in order to deal with limitations such as the availability of teachers, the workload of the subjects and the capacity of the classrooms, in order to improve the efficiency and equity of the service offered to the academic community. The presented solution was applied to the instances provided by ITC-2007.

Keywords: Class Schedule Scheduling Problem, GRASP, Combinatorial Optimization, ITC-2007

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABC	*Artificial Bee Colony*
ACO	*Ant Colony Optimization*
CB-CTP	*Curriculum Based Course Timetabling Problem*
CP	*Constraint Programming*
DI	*Departamento de Informática*
FP	*First Period*
GA	*Genetic Algorithm*
GD	*Great Deluge*
GRASP	*Greedy Randomized Adaptive Search Procedures*
HC	*Hill Climbing*
ILP	*Integer Linear Programming*
ITC	*International Timetabling Competition*
LE	*Largest Enrolment*
LCD	*Largest Colour Degree*
LD	*Largest Degree*
LWD	*Largest Weighted Degree*
MA	*Memetic Algorithm*
MCP	*Minimum Cost Period*
MIP	*Mixed Integer Programming*
PEB-CTP	*Post Enrolment Based Course Timetabling Problem*
PSO	*Particle Swarm Optimization*

RP	*Random Period*
SA	*Simulated Annealing*
TS	*Tabu Search*
UCTP	*University Course Timetabling Problem*
UERN	*Universidade do Estado do Rio Grande do Norte*
VNS	*Variable Neighborhood Search*

SUMÁRIO

1. INTRODUÇÃO

O Problema de Agendamento de Aula, também conhecido como "Timetabling Problem" ou UCTP, é um desafio de otimização no qual se deve alocar o cronograma de disciplinas de um curso levando em consideração muitas restrições, como disponibilidade do professor, interdependências entre disciplinas e interesses dos alunos.

Este problema é um clássico de otimização combinatória e é um tópico frequentemente discutido por muitos pesquisadores e estudiosos da área. Além disso, este é um problema frequente nas instituições de ensino superior, deve-se ter o cuidado de implementar um novo horário de aulas a cada semestre levando em consideração uma série de variáveis como cursos, disciplinas, salas disponíveis, etc. Ou seja, devemos propor soluções cada vez mais eficazes utilizando diferentes abordagens, que podem variar dependendo de como o problema é modelado.

Computacionalmente, o UCTP (do inglês University Course Timetabling Problem) pode ser considerado do tipo NP-Completo e NP-Difícil. Em outras palavras, encontrar a solução ótima leva um tempo exponencial,mas algoritmos heurísticos podem ser usados para encontrar uma solução satisfatória em um período de tempo razoável.

Um dos métodos heurísticos mais utilizados para solucionar o Problema do Agendamento de Horários de Aulas é a metaheurística GRASP. A metaheurística GRASP tem como etapas:

- Inicialização: A solução inicial é obtida através de um método guloso e/ou randômico;
- Iterações: O algoritmo realiza uma busca local na vizinhança da solução atual para explorar o conjunto de soluções vizinhas. Avaliando a qualidade da solução com base na função objetivo
- Randomização: O GRASP introduz um elemento de aleatoriedade para permitir que o algoritmo faça escolhas aleatórias, assim, ajudando a sair do ótimo local e explorar o espaço de soluções
- Atualização: A melhor solução encontrada durante as iterações é repetida

como solução atual se ela for melhor que a solução anterior.

- Finalização: O algoritmo repete as iterações até que um critério de parada seja atendido, como o número máximo de iterações ou um determinado nível de qualidade da solução

O agendamento de aulas é extremamente importante para as instituições de ensino superior, pois um cronograma eficaz pode influenciar positivamente a qualidade do ensino, a satisfação dos alunos e professores e a produtividade dos alunos. No entanto, criar cronogramas manualmente é uma tarefa complexa e demorada, muitas vezes levando a soluções improvisadas que podem ser ineficazes, deixando todos os envolvidos insatisfeitos.

1.1. Objetivos

Este trabalho propõe projetar um sistema automatizado de agendamento de horários de aulas no contexto universitário. O sistema usará a metaheurística GRASP para determinar soluções aproximadas ou ótimas para o problema.

Automatizar a criação desses cronogramas ajuda a resolver algumas limitações problemáticas, especificamente:

- Disponibilidade dos professores;
- Carga horária das disciplinas;
- Capacidade das salas de aula.

Acreditamos que esta iniciativa será uma valiosa contribuição para a melhoria do serviço de agendamento de horários de aulas oferecido à comunidade universitária. O sistema proposto pode criar horários mais equitativos e eficazes, proporcionando benefícios claros aos alunos, professores e colaboradores do departamento.

1.2. Organização do trabalho

Este trabalho foi estruturado em capítulos, que foram organizados da seguinte forma:

O Capítulo 2 é dedicado à fundamentação teórica. Apresentará, portanto, os conceitos essenciais que sustentam o estudo, abordando tanto o problema do agendamento dos horários das aulas como os princípios da metaheurística GRASP. Ainda neste capítulo será formulado o problema, apresentadas suas limitações, definida a função objetivo e casos utilizados para testar e validar as soluções.

O Capítulo 3 concentra-se no método proposto. Aqui será detalhada a estratégia utilizada para solucionar o problema e discutidos os métodos utilizados em nossa abordagem, justificando suas escolhas em bases teóricas. Adicionalmente, serão apresentados detalhes do algoritmo GRASP customizado, explicando como ele foi adaptado para resolver o problema de agendamento de horários de aulas

O Capítulo 4 centra-se nos resultados e na avaliação. Aqui serão apresentados os resultados obtidos ao longo do estudo, incluindo a etapa final do algoritmo. Realizamos uma análise crítica dos resultados e sua interpretação de acordo com os objetivos do estudo.

No Capítulo 5, concluímos o nosso trabalho resumindo as principais conclusões retiradas dos resultados. Além disso, destacamos perspectivas futuras, descrevemos possíveis direções para futuras pesquisas e melhorias do sistema proposto.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. O Problema do Agendamento de Horários

O problema de agendamento de horários é um problema complexo de otimização que envolve a alocação de recursos em períodos de tempo específicos para cumprir certas tarefas ou compromissos visando atingir objetivos específicos. Este desafio é de grande importância em diversos contextos, tais como: gestão de recursos humanos, programação de produção, logística e transporte, planejamento de eventos, saúde, entre outros.

Dada a sua complexidade, o agendamento de horários coloca uma série de desafios. Entre eles, destacam-se questões relacionadas à complexidade computacional, restrições específicas, objetivos conflitantes, situações dinâmicas, incerteza dos dados, proporção do problema, recursos apropriados e satisfação das partes interessadas.

Para superar esses desafios, várias abordagens foram desenvolvidas, incluindo algoritmos de otimização, métodos de programação linear e técnicas de inteligência artificial, como a metaheurística GRASP, algoritmos de busca local e aprendizado de máquina. A escolha da abordagem apropriada depende da natureza específica do problema e das restrições envolvidas.

O problema de agendamento de horários educacionais é um problema complexo relacionado à organização de horários em instituições de ensino como escolas e universidades. Esta questão tem sido objeto de pesquisas acadêmicas desde a década de 1960, e há riqueza de literatura sobre o assunto.

Os primeiros trabalhos sobre este assunto foram apresentados por Gotlieb (1963) e Csima e Gotlieb (1964), mas estudos relacionados incluem trabalhos como "Introduction to the Theory of Scheduling" e "A Survey of Automated Timetabling", enquanto questões educacionais são abordadas em publicações como "Time and School Learning". Em 1999, Schaerf estabeleceu três principais classificações para o problema de programação de horários educacionais, que ajudam a organizar e compreender a diversidade de desafios na área. São eles:

- **Problema de Horário de Escolar (*School Timetabling Problem*):** diz a

respeito à organização do tempo de aprendizagem semanal dos horários das turmas de uma escola, em que professores e as turmas devem ser distribuídos de forma que os professores não estejam presentes em mais de uma turma ao mesmo tempo e as turmas não tenham aulas com mais de um professor simultaneamente.

- **Problema de Horários de Cursos Universitários (University Course Timetabling Problem):** diz respeito à repartição dos cursos por disciplinas dentro de uma instituição de ensino superior, tendo em conta o número de salas disponíveis e a capacidade das salas de aula. Os alunos têm a oportunidade de escolher cursos de disciplinas de seu programa, e as aulas podem incluir alunos de diferentes programas.
- **Problema de Horário de Exames (Examination Timetabling Problem):** Resolver a alocação de exames entre cursos universitários, evitando conflitos de horários de exames entre exames relacionados a alunos regulares.

Neste trabalho, focamos nossa atenção na questão do Problema de Horários de Cursos Universitários (*University Course Timetabling Problem*), por ser o que mais se assemelha à nossa realidade e contexto. Deve-se notar que o UCTP pode ser subdividido em dois tipos. São eles:

- **Grade horária do Curso Baseada na Pós-Matrícula (PEB-CTP ou *Post Enrolment Based Course Timetabling Problem*):** Neste caso, a programação dos horários ocorre após a matrícula dos alunos. Isso envolve atribuir eventos letivos (aulas) a horários e salas de aula específicas, levando em consideração as escolhas e as disponibilidades dos alunos. Os alunos têm a flexibilidade para escolher as turmas das disciplinas com base nos seus interesses, e a instituição deve acomodar essas escolhas dentro dos recursos disponíveis.
- **Grade Horária do Curso Baseada no Currículo (CB-CTP ou Curriculum Based Course Timetabling Problem):** Aqui o horário é elaborado tendo em conta as necessidades de outras disciplinas que fazem parte do currículo de diferentes cursos universitários. Trata-se de atribuir uma série de aulas semanais a diferentes disciplinas, atribuindo-as a horários e salas específicas, tendo em conta a capacidade das salas e as necessidades das diferentes disciplinas nos diferentes currículos.

Resolver com eficiência o problema de agendamento de horários educacionais é importante para garantir que os alunos recebam uma educação de qualidade, evitem conflitos de horário e utilizem efetivamente os recursos institucionais.

2.2. O Problema de Horários de Cursos baseado em Currículo segundo o ITC-2007

O ITC-2007, sigla em inglês para International Timetabling Competition, foi um evento realizado em 2007 que desempenhou um papel importante na promoção da investigação e desenvolvimento de novos algoritmos para a resolver desafios de programação de horários no mundo real. Esta competição inclui três componentes distintas:

- **Componente 1:** Problema de Agendamento de Exames;
- **Componente 2:** Problema de Agendamento de Cursos Pós-Matrícula (PEB-CTP);
- **Componente:** Problema de Agendamento de Cursos Baseado no Currículo (CB-CTP).

Cada um desses componentes possuía seu próprio conjunto de instâncias de problemas, e os participantes eram convidados a submeter suas soluções para avaliação. Os vencedores de cada componente foram determinados com base na qualidade das suas soluções, medida por um conjunto de métricas de desempenho pré-determinadas.

Os algoritmos de programação de horários desenvolvidos durante o ITC-2007 continuam a ser usados hoje em dia por diversas instituições, como universidades e escolas, para resolver problemas reais de programação de horários. Neste trabalho, abordaremos especificamente o desafio da Programação de Cursos Baseada no Currículo (CB-CTP).

A principal motivação para este trabalho reside no fato de que, atualmente, o agendamento do horário de aulas para as turmas de Ciência da Computação do Departamento de Informática da UERN - Campus Central é feito de forma manual. Assim, a modelagem do problema ora em estudo será realizada considerando o contexto dos dados do referido departamento.

Para entender o processo atual, foi realizada uma entrevista com o técnico responsável pela definição do horário, que descreveu detalhadamente como ocorre esse processo. Nesse sentido, será apresentada a seguir uma descrição detalhada deste procedimento, as restrições que devem ser levadas em consideração e a solução proposta para o problema.

A implementação do algoritmo GRASP para o problema de otimização de horários será validada neste trabalho, com casos gerais obtidos na literatura, mas a utilização do método com casos reais pode ser realizada sem grandes dificuldades e fáceis adaptações. Experimentos com instâncias reais, geradas a partir de dados do Departamento de Informática da UERN serão realizados para futuras publicações.

2.2.1. Formulação do Problema

Levando em conta a montagem do horário a cada semestre, é fundamental entender que ele é a base para o próximo semestre. O horário segue uma estrutura que inclui vários aspectos:

- **Divisão por Turmas**: As turmas representam períodos (ou semestres) de um curso. Nos períodos pares as turmas são distribuídas em semestres pares, e o mesmo acontece nos períodos ímpares. Por exemplo, no semestre 2023.1 foram realizadas as turmas dos períodos I, III, V e VII do curso de Ciência da Computação. No semestre 2023.2 são oferecidas turmas dos períodos II, IV, VI e VIII. É importante ressaltar que o curso de Ciência da Computação é composto por um total de 8 períodos, sendo que cada um com disciplinas ministradas por professores diferentes.

- **Divisão por Dias da Semana:** As aulas podem ser agendadas de segunda a sábado para atender diferentes turmas. Neste contexto, vamos nos referir apenas aos dias úteis, com exceção dos sábados letivos, que são reservados à recuperação das aulas e não prejudicam a elaboração do horário.

- **Divisão por Horários:** As aulas acontecem em intervalos pré-determinados e não variam a cada semestre. A única diferença é a distribuição das combinações de disciplinas/professores nestes horários. Portanto, a quantidade de horários em uma semana pode ser determinada multiplicando o número de horários em um dia pelo número de dias da semana.

- **Identificação das Aulas:** Cada aula é definida por uma combinação de disciplina e professor(es) responsável(is). As disciplinas podem ser divididas como obrigatórias ou optativas. Disciplinas obrigatórias são as que os alunos devem cursar para concluir o curso, no entanto as disciplinas optativas oferecem aos alunos a oportunidade de escolher entre diversas disciplinas.

A estrutura de horário do CB-CTP (Curso Baseado no Currículo) desempenha um papel essencial na organização e planejamento das atividades acadêmicas, assegurando que os alunos tenham acesso às disciplinas essenciais para sua formação, ao mesmo tempo em que proporciona flexibilidade por meio das disciplinas optativas. A otimização deste processo é de fundamental importância para garantir uma experiência de aprendizagem eficaz e satisfatória.

Tabela 1 - Horário da turma do IV Período de 2023.2,utilizado como contexto no modelo do problema

HORÁRIO	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA
07:00 – 07:50	---------	Computação Gráfica - 90h (Heitor)	**1**. Tópicos Esp. em Eng. de Software – 30h – **optativa** (Max) **2**. Tópicos Esp. em Redes de Computadores-30h-**optativa** (Alysson)	Computação Gráfica - 90h (Heitor)	Computação Gráfica - 90h (Heitor)
07:50 – 08:40	Banco de Dados I - 60h (Alysson)	Computação Gráfica (Heitor)	**1**. Tópicos Esp. em Eng. de Software – 30h – **optativa** (Max) **2**. Tópicos Esp. em Redes de Computadores-30h-**optativa** (Alysson)	Computação Gráfica - 90h (Heitor)	Computação Gráfica - 90h (Heitor)
08:55 – 09:45	Banco de Dados I - 60h (Alysson)	Banco de Dados I - 60h (Alysson)	Linguagens Formais e Autômatos – 60h *(Isaac)*	Hardware III – 60h (André)	Hardware III – 60h (André)
09:45 – 10:35	UCE - 0107 (Lima / Max)	Banco de Dados I - 60h (Alysson)	Linguagens Formais e Autômatos – 60h (Isaac)	Hardware III – 60h (André)	Hardware III – 60h (André)
10:50 – 11:40	UCE - 0107 (Alysson/Sebastião)	Linguagens Formais e Autômatos – 60h (Isaac)	Tópicos Esp. em Modelagem de Problemas de Otimização e IA – 30h – **optativa** (Dario)	Fundamentos de Redes de Computadores (Rommel)	Fundamentos de Redes de Computadores (Rommel)
11:40 – 12:30	UCE - 0107 (Heitor)	Linguagens Formais e Autômatos – 60h (Isaac)	Tópicos Esp. em Modelagem de Problemas de Otimização e IA – 30h – **optativ**a (Dario)	Fundamentos de Redes de Computadores (Rommel)	Fundamentos de Redes de Computadores (Rommel)
VESPERTINO 13h50			Seminários em Ciência da Computação IV – 15h - (Henrique)		

Fonte: Departamento de Informática da UERN (Campus Central)

A Tabela 1 apresenta a grade horária de aulas de uma turma de Ciência da Computação da Universidade do Estado do Rio Grande do Norte (UERN). Nela, é visível observar que a turma pode ter aulas em até sete horários distintos, conforme o dia da semana. Os cursos oferecidos e os professores responsáveis estão listados nos respectivos horários.

Uma observação de relevância é que durante as duas primeiras aulas da quarta-feira contou com duas aulas de disciplinas distintas, lecionadas por professores distintos. Isso é possível porque essas disciplinas são optativas. Ou

seja, os alunos podem escolher qual delas desejam cursar, ou até mesmo não cursar nenhuma delas.

Utilizando a lógica de que as turmas são organizadas por número de semestre (par ou ímpar), a montagem do horário do semestre seguinte é inicialmente criado copiando o horário do semestre anterior. Por exemplo, a grade horária do semestre 2023.2 é baseada na grade horária do semestre 2022.2.

O técnico contactará então cada professor individualmente para verificar se eles podem continuar a lecionar as mesmas disciplinas nos mesmos dias e horários da grade anterior. Caso os professores não estejam disponíveis, deverão declarar disponibilidade, permitindo ao técnico ajustar manualmente o horário das aulas.

Este ajuste inclui examinar os horários disponíveis para acomodar os professores para alocar as aulas. Se não houver horários livres, o técnico precisa verificar se outro professor está disposto a efetuar a permuta. O horário será considerado eficaz quando for viável, isto é, se os professores e alunos que o utilizam estiverem pelo menos moderadamente satisfeitos.

O Departamento de Informática (DI) conta atualmente com 19 professores, embora outros departamentos possam ser convidados para lecionar se necessário. Além disso, a extensa grade de disciplinas do curso permite o ensino de até 28 disciplinas por semestre. Portanto, definir a grade horária manualmente é um processo trabalhoso e sujeito a erros que, de acordo com o técnico responsável, pode levar cerca de 30 dias.

Para solucionar esse desafio, é essencial otimizar o processo. Para isso, será empregado um algoritmo GRASP, modelado para atender à divisão em turmas, dias da semana e alocação das combinações professor/disciplina em horários, com o objetivo de satisfazer todas as restrições estabelecidas pelo técnico. Essas restrições serão detalhadamente exploradas no tópico subsequente.

2.2.2. Restrições

Segundo Pillay (2013), o *timetabling problem* inclui a alocação de professores, turmas e salas nos espaços de tempo para satisfazer as restrições fortes e fracas de um problema. Restrições são simplesmente condições que devem

ser atendidas no contexto de um problema. Pode ainda ser classificado como rígido ou flexível.

Segundo Mikuska (2015), restrições fortes (ou rígidas) são consideradas as condições que inviabilizam uma solução. No entanto, soluções fracas (ou flexíveis) são as condições que afetam a qualidade da solução proposta. Idealmente, nenhuma restrição rígida deverá ser violada, pois isso poderia tornar a solução inviável. As restrições flexíveis, que são apenas desejáveis, podem ser violadas, porque mesmo que isto conduza a uma solução de baixa qualidade, ainda é aplicável.

No contexto desse trabalho, as restrições foram classificadas da seguinte forma:

- **Restrições Rígidas**
    - **Dias de aula:** Só podem ser alocadas aulas nos dias pré-estabelecidos pela instituição. Ela é violada quando é alocado fora dos dias estabelecidos;
    - **Horário da aula:** Só podem ser alocadas aulas dentro dos horários pré-estabelecidos pela instituição. Essa restrição é violada caso seja alocada aula fora de um horário estabelecido;
    - **Alocação da disciplina:** Todas as aulas de uma disciplina devem ser alocadas e atribuídas a horários diferentes. Essa restrição é violada quando uma disciplina não tem suas aulas agendadas ou quando duas disciplinas distintas de uma mesma turma são alocadas no mesmo horário;
    - **Ocupação da sala:** Duas aulas não podem ser realizadas na mesma sala no mesmo horário. Essa restrição é violada quando duas aulas forem alocadas na mesma sala e no mesmo horário;
    - **Disponibilidade do professor:** Um professor só pode ministrar uma aula por horário, e essa alocação deve considerar a sua disponibilidade pessoal. Essa restrição é violada caso seja atribuída aula em um período que o professor esteja indisponível (seja por

escolha pessoal ou por já estar alocado com alguma disciplina no mesmo horário);

- **Conflitos**: Aulas de disciplinas de uma mesma turma ou ministradas pelo mesmo professor devem ser agendadas em horários diferentes. Essa restrição é violada quando aulas da mesma turma ou de um mesmo professor não alocadas no mesmo horário;
- **Combinação professor/disciplina:** Algumas disciplinas só podem ser ministradas por professores especialistas. Essa restrição é violada quando um professor é atribuído a uma disciplina em que ele não possui formação;

- **Restrições Flexíveis**
  - **Capacidade da sala:** Uma sala não deve receber aulas de uma disciplina que possua mais alunos matriculados do que sua capacidade permite. Essa restrição é violada quando uma disciplina com 30 alunos é alocada em uma sala cuja capacidade é de apenas 25 alunos, por exemplo. Cada aluno acima da capacidade receberá uma penalidade de 1 ponto;
  - **Dias mínimos de trabalho:** As disciplinas possuem cargas horárias que podem variar entre 15, 30, 45, 60 e 90 horas. Devido a isso, a distribuição dessa carga horária deve obedecer a um número mínimo de dias de aula por semana, conforme indicado abaixo:
    - As disciplinas com 15h devem ser alocadas em 1 dia, sendo 1 aula por semana;
    - As disciplinas com 30h devem ser alocadas em 1 dia, sendo 2 aulas por semana;
    - As disciplinas com 45h devem ser alocadas em 1 dia, sendo 3 aulas por semana;
    - As disciplinas com 60h devem ser alocadas em 2 dias, sendo 4

aulas por semana;

- As disciplinas com 90h devem ser alocadas em 2 dias, sendo 6 aulas na semana.

Cada dia abaixo do mínimo receberá uma penalidade de 5 pontos;

- **Ociosidade da turma:** As aulas de uma turma devem ocorrer sempre em horários consecutivos. O objetivo é evitar que uma turma fique com um período muito grande sem aulas entre os horários finais e iniciais. Cada horário de aula isolado receberá uma penalidade de 2 pontos;
- **Estabilidade da sala:** Todas as aulas de uma disciplina devem ser ministradas na mesma sala. O objetivo é evitar que os alunos e professores precisem se deslocar para salas diferentes para assistirem a aulas da mesma disciplina. Cada sala distinta usada para as aulas de uma mesmo disciplina receberá uma penalidade de 1 ponto;

Não são consideradas restrições, mas podem ocorrer as seguintes situações:

- Um professor ministrar mais de uma disciplina para a mesma turma ou em um mesmo semestre;
- Uma disciplina ser ministrada por mais de um professor em um mesmo semestre.

Dado que uma solução precisa atender a todas as restrições rígidas para ser considerada viável e que a qualidade da solução é determinada pela satisfação das restrições flexíveis, logo, a tabela de horário ótima é a que atende a todas as restrições rígidas e maximiza a satisfação das restrições flexíveis (JAENGCHUEA; LOHPETCH, 2015; TEOH; ABDULLAH; HARON, 2015).

2.2.3. Função Objetivo

Uma função objetivo é uma equação matemática usada na otimização para avaliar a qualidade de uma solução de acordo com determinados critérios. isto é importante em problemas de otimização porque ajuda a encontrar a melhor solução. Alcançar a função objetivo requer a compreensão das restrições que devem ser atendidas para que a solução seja considerada viável.

Neste trabalho, as restrições afetam a qualidade e a viabilidade da atribuição de horários. A violação de restrições rígidas, como a limitação dos dias de aula, horários de aula, disponibilidade dos professores, combinação professor/disciplina e alocação das disciplinas, pode inviabilizar a solução. Existem restrições flexíveis, como a ociosidade da turma e a estabilidade da turma, que afetam a qualidade da solução, mas podem ser violadas se necessário.

Considerando que as restrições rígidas já serão verificadas na etapa de construção e as soluções que violam essas restrições serão rejeitadas, a função objetivo proposta pretende satisfazer as restrições flexíveis. Ou seja, devemos reduzir os intervalos (intervalos em que não há aulas no dia), distribuir adequadamente a carga horária de cada disciplina e garantir que as aulas da mesma disciplina sejam sempre ministradas na mesma sala de aula. Como tal, o cronograma ideal é aquele que satisfaz todas essas restrições e minimiza a lacuna para garantir a eficiência do processo de alocação.

Abaixo, segue a função que será utilizada neste trabalho.

$$\textit{Minimizar } F(S) = \sum (P_{capacidade\_sala}) + \sum (P_{dias\_mínimos}) + \sum (P_{compacidade\_turma}) + \sum (P_{estabilidade\_sala})$$

Onde:

- *F(O)* é a função objetivo;
- ∑ é o somatório das penalidades aplicadas a cada restrição flexível violada;
- $P_{capacidade\_sala}$ há uma penalidade de 1 ponto para cada aluno excedente da capacidade da sala;
- $P_{dias\_mínimos}$ há uma penalidade de 5 pontos para cada aula abaixo do número mínimo de dias de aula por semana;
- $P_{compacidade\_turma}$ há uma penalidade de 2 pontos para cada aula alocada em horários não subsequente;

- $P_{estabilidade\_sala}$ há uma penalidade de 1 ponto para cada aula de uma disciplina alocada em salas distintas.

Esta função calcula o custo total da solução para o problema de atribuição de aulas. O custo total é a soma de todas as penalidades, ou seja, cada restrição violada aumentará o valor da função objetivo pelo seu peso e tornará a solução menos viável.

2.2.4. Instâncias

Para auxiliar os competidores, o ITC disponibilizou conjuntos de instâncias que replicam as características de universidades do mundo real, possibilitando a comparação com diversos métodos de resolução.

Para este trabalho, utilizaremos as mesmas instâncias que os competidores do ITC-2007 utilizaram para o problema do CB-CTP, totalizando 21 instâncias com vários níveis de dificuldade. A organização da competição assegura a existência de soluções viáveis para todas as instâncias, confirmadas por meio de testes. No entanto, não foram fornecidas informações sobre violações de restrições flexíveis.

A Tabela 2 apresenta os dados relevantes de cada uma das 21 instâncias, com informações sobre o número de currículos, salas, disciplinas, horários por dia, dias, conflitos e disponibilidade. Os valores referentes a currículos, salas e disciplinas variam consideravelmente de uma instância para outra, enquanto os dados relacionados a horários por dia e dias são praticamente uniformes, exceto para a instância comp11.

É importante notar que o número de conflitos e a disponibilidade têm um impacto significativo no tempo de execução do algoritmo e na complexidade da busca por uma solução viável. Quanto mais conflitos e menos disponibilidade, mais desafiador se torna encontrar uma solução que atenda às restrições rígidas. Isso pode afetar tanto o processo de geração da solução inicial quanto a busca por soluções melhores na vizinhança durante o processo de otimização.

Tabela 2 - Informações sobre cada Instância do ITC-2007

Instância	Currículos	Salas	Disciplinas	Horários por dia	Dias	Conflitos	Disponibi-lidade
comp01	14	6	30	6	5	13.2	93.1
comp02	70	16	82	5	5	7.97	76.9
comp03	68	16	72	5	5	8.17	78.4
comp04	57	18	79	5	5	5.42	81.9
comp05	139	9	54	6	6	21.7	59.6
comp06	70	18	108	5	5	5.24	78.3
comp07	77	20	131	5	5	4.48	80.8
comp08	61	18	86	5	5	4.52	81.7
comp09	75	18	76	5	5	6.64	81
comp10	67	18	115	5	5	5.3	77.4
comp11	13	5	30	9	5	13.8	94.2
comp12	150	11	88	6	6	13.9	57
comp13	66	19	82	5	5	5.16	79.6
comp14	60	17	85	5	5	6.87	75
comp15	68	16	72	5	5	8.17	78.4
comp16	71	20	108	5	5	5.12	81.5
comp17	70	17	99	5	5	5.49	79.2
comp18	52	9	47	6	6	13.3	64.6
comp19	66	16	74	5	5	7.45	76.4
comp20	78	19	121	5	5	5.06	78.7
comp21	78	18	94	5	5	6.09	82.4

Fonte: Website do ITC-2007

Inicialmente, vamos desenvolver um algoritmo e testá-lo nas instâncias do ITC-2007. Quando alcançarmos resultados semelhantes aos obtidos pelos participantes da competição, aplicaremos o nosso algoritmo na instância construída com os dados reais do curso de Ciência da Computação da UERN.

A Figura 1 apresenta uma representação detalhada de cada instância, que é dividida nas seguintes seções:

Figura 1 - Arquivo de Entrada da Instância

```
Name: Toy
Courses: 4
Rooms: 3
Days: 5
Periods_per_day: 4
Curricula: 2
Constraints: 8

COURSES:
SceCosC Ocra 3 3 30
ArcTec Indaco 3 2 42
TecCos Rosa 5 4 40
Geotec Scarlatti 5 4 18

ROOMS:
rA 32
rB 50
rC 40

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec

UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

END.
```

Fonte: Dados da Instância Toy do ITC-2007 obtidas no site https://github.com/walacesrocha/Timetabling/blob/master/instancias/comp01.ctt

- **Cabeçalho:** Esta seção fornece informações gerais e descreve os recursos essenciais da instância antes de entrar em detalhes específicos sobre cursos (disciplinas), salas, currículos (turmas) e restrições;
- **Cursos (Courses):** Neste trecho fornece o número de cursos (ou disciplinas)

envolvidos, que são 4 na Figura 1. Os dados correspondentes indicam o nome da disciplina, o nome do professor, o número de aulas, o número mínimo de dias da semana em que devem ser ministradas as aulas e o número de alunos inscritos na disciplina;

- **Salas (Rooms):** Neste trecho fornece informação sobre número de salas disponíveis, que são 3 na Figura 1. Os dados indicam respectivamente o nome da sala e sua capacidade;

- **Currículos (Curricula):** Neste trecho são listados os currículos (ou turmas) dos quais existem apenas 2 na Figura 1. Os dados indicam respectivamente o nome do currículo (turma), a quantidade e a lista de disciplinas da turma;

- **Restrições (Unavailability_Constraints):** Neste trecho indica o número total de restrições ou limitações que precisam ser consideradas no problema. No caso da Figura 1, existem 8 restrições, e para cada uma delas são apresentados respectivamente o nome da disciplina, o dia e o período (horário) do dia em que ela não deve ocorrer. É importante ressaltar que os dias e os períodos começam em 0. Ou seja, a restrição "ArcTec 4 0" especifica que a disciplina "ArcTec" não pode ser agendada durante o primeiro período da quinta-feira.

2.2.5. Arquivo de Saída

O arquivo de saída nada mais é do que um arquivo único gerado a partir da aplicação do algoritmo construído para solucionar o problema do agendamento de horários educacionais. Ele segue o modelo do arquivo de saída da ITC-2007 e contém a solução viável, representada por linhas que indicam alocações de aulas. Essas alocações são determinadas usando a função objetivo definida na seção 2.2.4 deste trabalho. As linhas de alocação de aulas seguem o seguinte formato:

<Nome da disciplina> <Nome da Sala> <Dia> <Período do Dia>

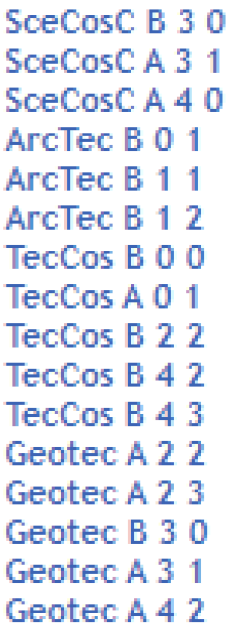
Lembrando que, como os dias e períodos começam com 0, por exemplo, "TecCos A 2 3" indica que a disciplina "TecCos" será ministrada na quarta-feira (2) no quarto período (3) na sala A.

2.2.6. Validação das Soluções

Da mesma forma que o ITC-2007 disponibilizou as instâncias modelo, ele também disponibilizou um validador com código fonte em C++ para validar as soluções geradas pelas abordagens desenvolvidas pelos competidores.

O validador funciona de forma simples: Primeiro ele lê o arquivo da instância (arquivo de entrada) e o arquivo com a solução (arquivo de saída). Em seguida, ele gera uma saída padrão com a avaliação da solução e a descrição detalhada de todas as violações de restrições rígidas e flexíveis.

Figura 2 - Arquivo de saída gerado a partir de uma instância



Fonte: Website do ITC-2007

A Figura 2 ilustra um exemplo de solução gerada a partir de uma instância, enquanto a Figura 3 ilustra um exemplo da saída padrão do validador.

Figura 3 - Saída do validador

```
[H] Courses ArcTec and TecCos have both a lecture at period 1 (day 0, timeslot 1)
[H] Courses TecCos and Geotec have both a lecture at period 10 (day 2, timeslot 2)
[H] Courses TecCos and Geotec have both a lecture at period 18 (day 4, timeslot 2)
[H] 2 lectures in room B the period 12 (day 3, timeslot 0)
[H] 2 lectures in room A the period 13 (day 3, timeslot 1)
[S(8)] Room A too small for course TecCos the period 1 (day 0, timeslot 1)
[S(5)] The course SceCosC has only 2 days of lecture
[S(5)] The course TecCos has only 3 days of lecture
[S(5)] The course Geotec has only 3 days of lecture
[S(2)] Curriculum Cur1 has an isolated lecture at period 10 (day 2, timeslot 2)
[S(2)] Curriculum Cur1 has an isolated lecture at period 16 (day 4, timeslot 0)
[S(1)] Course SceCosC uses 2 different rooms
[S(1)] Course TecCos uses 2 different rooms
[S(1)] Course Geotec uses 2 different rooms

Violations of Lectures (hard) : 0
Violations of Conflicts (hard) : 3
Violations of Availability (hard) : 0
Violations of RoomOccupation (hard) : 2
Cost of RoomCapacity (soft) : 8
Cost of MinWorkingDays (soft) : 15
Cost of CurriculumCompactness (soft) : 4
Cost of RoomStability (soft) : 3

Summary: Violations = 5, Total Cost = 30
```

Fonte: Website do ITC-2007

2.3. Abordagens para resolver Problemas de Programação de Horários Educacionais

Tal como mencionado no capítulo 1, vários pesquisadores estão a trabalhar na área de otimização, tentando encontrar as soluções mais eficazes para resolver os desafios associados à programação de horários educacionais. Através deste esforço, a literatura apresenta uma variedade de métodos computacionais para resolver este problema, incluindo heurísticas, meta-heurísticas, métodos híbridos, hiper-heurísticas, entre outros.

Nas subseções seguintes discutiremos o uso de Heurísticas Construtivas, com foco na construção gradual de soluções iniciais e métodos de otimização, utilizando técnicas diversas para melhorar a qualidade das soluções encontradas. Isso inclui a otimização de funções objetivas e a aplicação de algoritmos de busca local para obter horários mais eficientes e adequados.

2.3.1. Heurísticas Construtivas

As heurísticas construtivas são métodos de resolução de problemas que

dependem de uma abordagem humana para resolver desafios, construindo gradualmente a solução final a partir de um estado inicial. Isso envolve adicionar elementos gradualmente, com base em critérios específicos, até que uma solução completa seja alcançada ou critério de parada predeterminado seja satisfeito.

Existem várias abordagens para resolver o problema da programação de horários educacionais. Uma delas é o uso da metaheurística GRASP para automatizar o processo de alocação de horários. Outra alternativa é utilizar algoritmos híbridos, que combinam diferentes métodos metaheurísticos para buscar a melhor solução possível. Como exemplo desta segunda abordagem, podemos citar o trabalho de SENA (2021), que utilizou o Biased Random-Key Genetic Algorithm como metaheurística principal, o Simulated Annealing como estratégia de busca local, e a Cadeia de Kempe como método de perturbação para resolver o problema. Contudo, é importante estabelecer um ponto de partida: a solução inicial será ajustada iterativamente até que a solução final ou mais viável seja alcançada.

As heurísticas construtivas podem ser classificadas como aleatórias, gulosa ou parcialmente gulosa. Na construção aleatória, a solução é construída passo a passo, adicionando elementos selecionados aleatoriamente até ser concluída. Essa aleatoriedade pode nos ajudar a explorar diferentes regiões do espaço de busca, mas não garante a qualidade da solução. Na construção gulosa, também construímos a solução passo a passo, mas em vez de escolher aleatoriamente, a heurística escolhe o elemento que parece ser o melhor no momento, com base na função de avaliação local. Isso muitas vezes leva a soluções rápidas, no entanto não garante a melhor solução global. Finalmente, a heurística parcialmente gulosa é uma junção das estratégias aleatória e gulosa. Ela usa uma heurística gulosa para a maioria das decisões, mas ocasionalmente faz escolhas aleatórias para diversificar a busca. Isso pode auxiliar a evitar ficar preso em mínimos locais e a explorar um espaço de soluções mais amplo. Por esta razão, heurística parcialmente gulosa muitas vezes encontram soluções melhores do que as heurísticas puramente gulosas, sem aumentar significativamente o tempo de execução. O fim do método acontece quando todos os elementos são adicionados ou quando um critério de parada é atingido.

No contexto dos horários educacionais, essas técnicas podem ser usadas

para preencher gradativamente o horário, inserindo uma aula de cada vez até que todas as aulas sejam atribuídas ou até que sejam identificados conflitos (SCHAERF, 1999).

Dentro dessas abordagens, ainda pode haver heurísticas no agendamento de eventos, onde cada um tem sua própria lógica de prioridade. São eles:

- **Maior Matrícula (LE):** Os eventos são listados em ordem decrescente de acordo com a quantidade de alunos matriculados. Tem prioridade eventos com mais alunos .
- **Maior Grau (LD):** Os eventos tem prioridade com base na quantidade de conflitos que apresentam com outros eventos. Quanto mais conflitos, maior será a prioridade.
- **Maior Grau Ponderado (LWD):** Semelhante ao LD, mas leva em consideração a quantidade de alunos envolvidos em ambos os eventos, ao invés de apenas contabilizar a quantidade de eventos com conflitos.
- **Maior Grau de Cor (LCD):** Variação da LD que leva em consideração a quantidade de conflitos com eventos agendados.
- **Grau de Saturação (SD):** Prioriza eventos com a menor quantidade de horários disponíveis, tendo em conta restrições rígidas do cronograma.

Essas heurísticas podem ainda ser consideradas como estáticas ou dinâmicas. Nas estáticas (LE, LD e LWD), os valores heurísticos são determinados antes da construção do cronograma e permanecem os mesmos para cada evento durante todo o processo de construção. Todos os eventos são classificados com esses valores e são atribuídos em ordem e períodos viáveis. Se não houver um período viável, o evento pode não ser atribuído ou ser alocado aleatoriamente, aumentando o custo da restrição rígida. Já nas dinâmicas (LCD e SD), os valores heurísticos de cada evento não atribuído alteram à medida que os eventos são atribuídos no cronograma. No início do processo, todos os eventos possuem o valor heurístico igual. À medida que o cronograma é construído, o valor heurístico de um evento que compartilha um aluno com um evento recém-atribuído é ajustado. Tal como, no caso da heurística SD, o nível de saturação de um evento diminui à

medida que o cronograma é construído, já na heurística LCD, o valor heurístico de um evento não alocado aumenta à medida que os eventos são alocados (PILLAY; ÖZCAN, 2019; PILLAY, 2016a).

Nas heurísticas estáticas, existem três maneiras de selecionar a duração de um evento:

- **Primeiro Período (FP):** O evento é inserido no primeiro horário disponível onde não viola as regras rígidas;
- **Período Aleatório (RP):** O horário é selecionado aleatoriamente entre todos os horários disponíveis;
- **Período de Custo Mínimo (MCP):** Cada horário disponível é avaliado quanto aos custos associados à violação das restrições flexíveis no cronograma atual. O evento é atribuído no horário onde o custo da violação é menor.

Em geral, cada heurística tem sua própria abordagem para determinar a ordem em que os eventos são agendados, com base em diferentes critérios como a quantidade de alunos, conflitos ou períodos disponíveis. A abordagem escolhida para construir a solução inicial deste trabalho será discutida na subseção 3.2 do capítulo 3.

2.3.2. Abordagens de Otimização

Na literatura, encontramos diversas estratégias para resolver problemas de otimização. Esses métodos de resolução podem ser divididos em dois grupos principais: métodos exatos e métodos aproximados.

Os métodos exatos visam encontrar a solução ideal para o problema, garantindo resposta correta e livre de erros. No contexto da programação de horários educacionais, isto significa criar um horário que satisfaça de forma ideal todas as restrições. Vários métodos exatos, incluindo programação linear inteira (ILP), programação inteira-mista (MIP) e programação de restrições (CP), que podem ser aplicados para garantir uma solução exata, embora possam exigir muito tempo computacional.

Embora consigam encontrar a solução ideal, os métodos exatos apresentam

certas dificuldades que os tornam menos adequados para a resolução de problemas complexos. Uma dessas dificuldades é o alto consumo computacional, principalmente no caso de grandes espaços de busca (BEHESHTI; SHAMSUDDIN, 2013). Para problemas mais complexos da classe NP-Completo, como é o caso do UCTP, a utilização exclusiva destes métodos só é possível para instâncias pequenas. Classificar a versão de otimização do problema como NP-Difícil implica em maior complexidade computacional, impossibilitando a aplicação exclusiva de métodos exatos.

Alguns dos trabalhos que podem ser identificados na literatura utilizando os métodos exatos são: Programação Linear Inteira (MENDES; CONCATTO; SANTIAGO, 2018; SILVA; CAMPOS, 2017; QUEIROZ; NEPOMUCENO, 2017; BUCCO; BORNIA-POULSEN; BANDEIRA, 2017); Programação Inteira Mista (NEUKIRCHEN et al., 2014; SILVA, 2014; POULSEN; BUCCO; BANDEIRA, 2014) e Programação Linear Binária (ANDRADE; SCARPIN; STEINER, 2012; BORGES et al., 2015).

Por outro lado, os métodos de aproximação concentram-se em encontrar soluções que estejam próximas da solução ideal, mas que possam ser computacionalmente mais eficientes. Isso envolve a utilização de técnicas que equilibram a qualidade da solução e velocidade computacional, tornando o método mais adequado para resolução de problemas com uma alta complexidade computacional.

Os métodos de aproximação podem ser divididos em três classes: métodos heurísticos, metaheurísticos e métodos especiais.

Os métodos heurísticos são estratégias aproximadas de resolução de problemas que procuram encontrar soluções aceitáveis dentro de um período de tempo razoável, embora não haja garantia de encontrar a solução ótima. Frequentemente, são desenvolvidos com base na intuição, experiência ou em regras práticas específicas para basear a busca por soluções. Exemplos comuns incluem Busca Local e Busca Tabu. Estas abordagens são muitas vezes mais simples e são aplicadas a tipos específicos de problemas, aproveitando características específicas para encontrar soluções eficazes. Em um contexto da programação de horários, um método heurístico pode envolver a criação de horários iniciais e iterações para

aprimorar a solução, ajustando a alocação de aulas e recursos.

As metaheurísticas são estratégias mais amplas que coordenam a aplicação de métodos heurísticos em espaços de busca complexos e fornecem soluções satisfatórias e de boa qualidade que superam os métodos manuais com pouco esforço. Contudo, não garantem a solução ótima (JARDIM; SEMAAN; PENNA, 2015). Elas podem ainda ser categorizadas com base em sua abordagem, que pode ser baseada na solução ou população.

Exemplos de metaheurísticas baseadas em população: Otimização de Colônia de Formigas (ACO), Algoritmo Genético (GA, do inglês Genetic Algorithm), do inglês Ant Colony Optimization),Busca Tabu (TS, do inglês Tabu Search), Recozimento Simulado (SA, do inglês Simulated Annealing), Otimização de Enxame de Partículas (PSO, do inglês Particle Swarm Optimization), Subida da Encosta (HC, do inglês Hill Climbing), Colônia de Abelhas Artificiais (ABC, do inglês Artificial Bee Colony) e Algoritmo Memético (MA, do inglês Memetic Algorithm), Grande Dilúvio (GD, do inglês Great Deluge) e Busca de Vizinhança Variável (VNS, do inglês Variable Neighborhood Search) (VRIELINK et al.,2019).

Embora as metaheurísticas populacionais tenham potencial de explorar tanto global quanto localmente (BEHESHTI; SHAMSUDDIN, 2013), pesquisas mostram que os algoritmos de busca local podem ser mais eficiente em termos de tempo de execução, mesmo que não garantam a obtenção de soluções ótimas. Por outro lado, os algoritmos populacionais se destacam pela capacidade de explorar novas áreas e se aproximar do ótimo (HABASHI et al., 2018). Em suma, estes métodos, por serem mais abstratos e adaptáveis são particularmente de grande vantagem para a exploração eficiente de grandes espaços de busca e complexos, como é o caso da programação de horários educacionais.

Os métodos especiais são abordagens individualizadas e adaptadas para resolução de problemas específicos. Eles são desenvolvidos levando em conta as características únicas de um problema específico, como restrições específicas, estruturas de dados específicas ou características exclusivas. Esses métodos buscam otimizar a eficiência na busca de soluções, aproveitando perspectivas específicas do problema. Os métodos especiais podem ser mais específicos e visar um determinado tipo de problema, fornecendo soluções precisas para contextos

específicos. No caso da programação de horários educacionais, isso pode incluir levar em consideração as preferências de professores, as necessidades das salas de aula e as necessidades dos alunos, adaptando-se às especificidades da área.

Existem também os métodos híbridos e hiper-heurísticos. Os métodos híbridos, que combinam algoritmos baseados em população com técnicas de busca local, e hiper-heurísticas, que operam em um nível mais abstrato combinando e adaptando várias heurísticas, surgiram como abordagens mais adequadas para enfrentar os desafios complexos associados à programação de horários educacionais (SUSAN; BHUTANI, 2019; HABASHI et al., 2018; MATIAS; FAJARDO; MEDINA, 2018). Estas abordagens avançadas visam superar as fraquezas individuais através da combinação de estratégias, aproveitando as vantagens específicas de cada abordagem. Os métodos híbridos procuram integrar a eficiência dos algoritmos baseados em população com a precisão das técnicas de busca local, enquanto as hiper-heurísticas oferecem a flexibilidade para explorar o grande espaço heurístico. A sua complexidade e adaptabilidade tornam estas abordagens mais eficazes na abordagem das várias restrições e prioridades específicas aos problemas de programação de horários educacionais, trazendo assim soluções mais eficazes e qualidade alta.

A decisão sobre qual método a ser adotado depende de vários fatores, incluindo a natureza do problema, o tamanho da instância, a disponibilidade de recursos computacionais e as preferências das partes interessadas. Muitas vezes, uma combinação de métodos exatos e aproximados pode ser usada para equilibrar a precisão e a eficiência na resolução de problemas de programação de horários educacionais.

3. ABORDAGEM PROPOSTA

A solução para o CB-CTP é desenvolvida em duas fases: construção e melhoria. Durante a fase de construção, uma tabela de horários é criada iterativamente sem violar restrições rígidas, mas com a capacidade de violar restrições flexíveis. Durante a fase de melhoria, a solução inicial é gradualmente melhorada para que sejam violadas o mínimo de restrições flexíveis. Essas etapas são importantes porque afetam a taxa de convergência do algoritmo e a qualidade da solução final (WAHID; HUSSIN, 2016).

Nas seções a seguintes, serão apresentados o metaheurística GRASP, a construção da solução inicial, a fase de melhoria (busca local) e a avaliação da solução.

3.1. A Metaheurística GRASP

A meta-heurística GRASP (Procedimentos de Busca Aleatória Adaptativa e Gulosa, do inglês Greedy Randomized Adaptive Search Procedure) foi introduzida por Feo e Resende (1989), com o objetivo de lidar com o problema de cobertura de conjuntos. O GRASP já foi aplicado com êxito em diversos problemas de otimização, como o conjunto máximo (Feo, Resende e Smith 1994), o roteamento de circuitos virtuais (Resende e Ribeiro 2003), dentre outros. No problema de agendamento de horários o GRASP foi abordado por ROCHA em 2013 utilizando na busca local Hill Climbing e Simulated Annealing, onde obteve resultados competitivos para o ITC-2007.

O GRASP é um algoritmo metaheurístico, onde cada iteração é construída uma solução inicial e aplicado busca local para melhora-lá, que se destaca na resolução de problemas de otimização combinatória (POC). O GRASP se baseia em princípios probabilísticos e construtivos para explorar o espaço de busca de forma eficiente e eficaz.

Um exemplo do Algoritmo GRASP proposto, em forma de pseudocódigo, é apresentado na Figura 4

Figura 4 - Pseudocódigo do Algoritmo GRASP

```
função GRASP(DisciplinasOrd, Cursos, Salas, Restricoes, Curriculo, TamLRC, MaxIter)
 // inicializa os valores para a melhor (star) solução e FO
 fo_solucao_star = 99999999
 solucao_star = Nulo

 // Repetição baseado no critério de parada MaxIter
 para i de 1 até MaxIter faça
  solucao = Nulo
  SolucaoEncontrada = Falso

  // Geração da solução Inicial
  solucao = MtGulosoAleatorio2(DisciplinasOrd, Cursos, Salas, Restricoes, TamLRC)

  // Verifica se todas as aulas foram alocadas
  se verificaNumAulas(solucao, Cursos) então
   SolucaoEncontrada = Verdadeiro
  fim se

  // Todas as aulas alocadas
  se SolucaoEncontrada então
   // Inicia o processo de busca local
   solucao, fo_solucao = BuscaLocal(solucao, Cursos, Salas, Restricoes, Curriculo)

   // Escolhe o melhor valor e atribui a solução star
   se fo_solucao < fo_solucao_star então
    fo_solucao_star = fo_solucao
    solucao_star = solucao
   fim se
  fim se
 fim para

 // Retorna a solução
 retorne solucao_star
fim função
```

Fonte: Autoria própria (2024)

Os parâmetros utilizados na implementação do GRASP são MaxIter e TamLRC, onde MaxIter que determina o número máximo de iterações do GRASP e TamLRC será o tamanho da lista restrita de candidatos usado na criação da solução inicial. Inicialmente o MaxIter foi inicializado com o valor de 100 e TamLRC com o valor de 5, caso não seja passado o parâmetro, o valor padrão é 3.

3.2. Solução Inicial

O algoritmo GRASP inicia com a criação de uma solução inicial. Essa solução pode ser vista como uma construção gulosa, que gera uma solução parcial que será refinada ao longo das iterações do algoritmo. A fase de construção do GRASP utiliza da abordagem gulosa com aleatoriedade, onde o objetivo é gerar soluções iniciais diversificadas para explorar diferentes regiões no espaço de busca. A aleatoriedade é realizada com base na criação de uma LRC (Lista Restrita de Candidatos), onde é definido um tamanho padrão ou passado por parâmetro para essa lista.

O objetivo principal deste procedimento é produzir uma solução viável, ou seja, sem violar as restrições rígidas. Para alcançar esse objetivo, é utilizada uma estratégia de alocar as aulas que tenham o maior número de conflitos (número de restrições somado com o número de currículos).

Inicialmente para a construção da solução inicial foi utilizado o método puramente guloso com 3 critérios de ordenação, sendo eles:

- Critério 1: Somatório do número de restrições;
- Critério 2: Número de restrições somado com o número de currículos em que a disciplina pertence;
- Critério 3: Somatório do número de currículos em que a disciplina pertence.

Devido em algumas instâncias não ter sido possível alocar todas as aulas como mostra a Tabela 3 onde os melhores valores estão em negrito. Portanto, para solucionar o problema de todas as aulas não serem alocadas foi utilizado o Método Guloso Aleatório com o critério 2 de ordenação por ter sido o com melhor desempenho nos testes.

Tabela 3 - Resultado do Método Puramente Guloso

Instâncias	Critério 1	Critério 2	Critério 3
Comp01	1	**0**	**0**
Comp02	5	**3**	**3**
Comp03	4	**1**	**1**
Comp04	0	**0**	**0**
Comp05	**2**	**2**	**2**
Comp06	3	**0**	**0**
Comp07	3	**2**	**2**
Comp08	1	**0**	**0**
Comp09	**0**	1	1
Comp10	4	**0**	**0**
Comp11	**0**	**0**	**0**
Comp12	**2**	**2**	**2**
Comp13	**0**	**0**	**0**
Comp14	**0**	**0**	**0**
Comp15	**0**	**0**	**0**
Comp16	1	**0**	**0**
Comp17	**0**	2	2
Comp18	**0**	**0**	**0**

Comp19	**2**	**2**	**2**
Comp20	**0**	1	1
Comp21	**2**	5	5

Fonte: Autoria própria (2024)

O método guloso aleatório foi estruturado da seguinte forma:

- Gera a LRC com base na estratégia de conflitos utilizada;
- Sorteia-se uma disciplina da LRC;
- Percorre o array ordenado (menor para a maior) com base na capacidade das salas. O array é composto por dias, horários e salas. Verifica-se se a sala comporta todos os alunos;
- Realiza as verificações necessárias para que as restrições rígidas não sejam violadas;
- Caso não viole, aloca-se a aulas e exclui do array de horários disponíveis e acrescenta o contador;
- Caso não comporte todos os alunos;
- Realiza as verificações necessárias para que as restrições rígidas não sejam violadas;
- Caso não viole, aloca-se a aulas e exclui do array de horários disponíveis e acrescenta o contador;
- Repete os passos acima até que todas as aulas sejam alocadas.

3.3. Busca Local

O objetivo da fase de busca local é melhorar a solução inicial, para isso foi implementado o movimento de SWAP, onde duas aulas trocam de posição na tabela de horário.

O SWAP está sendo aplicado da seguinte forma:

- São escolhidos de forma pseudo-aleatória a aula, dia e horário;
- Caso não haja aula no dia e horário escolhidos a aula é alocada e a busca encerra;
- Caso haja aula alocada vai iterar sobre tabela de horários a fim de encontrar a aula que está dia e horário escolhidos;
- O SWAP só ocorrerá quando as restrições rígidas não forem violadas, caso sejam violadas a troca não acontecerá e será retornando a solução inicial;

- O processo se repete até em 20 iterações.

Após o procedimento de SWAP é verificado o valor da função objetivo, caso o valor seja melhor é adotado como nova solução, assim, retornando a melhor solução e o melhor valor.

3.4. Avaliação da Solução

A solução é avaliada utilizando a função objetivo, que é a soma das penalidades atribuídas às restrições flexíveis violadas. A avaliação permite medir a qualidade de cada solução em relação ao objetivo do problema. Neste caso, procuramos a solução com a menor pontuação, pois isso indica um menor número de violações flexíveis. Caso seja identificada uma violação, penalidades são contabilizadas e aplicadas, conforme explicado na subseção 2.2.3 deste documento.

Foram criadas 4 funções para verificar as violações das restrições flexíveis em cada solução. São elas:

- **Calcula_Violacao_Salas:** Esta função verifica a restrição flexível "Estabilidade da sala". Para isso, cada disciplina alocada é verificada se é alocada em mais de uma sala. Se houver a função conta o número de salas extras e soma a penalidade, retornando a soma total da penalidade;
- **Calcula_Violacao_CapacidadeSala:** Esta função verifica a restrição flexível “Capacidade da sala”. Para isso, verifica-se a quantidade de alunos excedentes em cada disciplina que excede a capacidade da sala onde ela foi alocada. Se houver violação é feita a diferença entre a quantidade de alunos e a capacidade da sala, retornando essa diferença total ao final da função;
- **Calcula_Violacao_AulaIsolada:** Esta função verifica a restrição flexível "Ociosidade da turma". Para cada aula alocada, ela separa por currículo e em seguida por dias, fazendo a verificação para ver se há intervalos entre aulas do mesmo currículo e dia. Se houver adiciona 1 violação, retornando a soma total de violações;
- **Calcula_Violacao_MinSemanal:** Esta função verifica a restrição flexível "Dias mínimos de trabalho". Para isso, é verificado em quantos dias diferentes da semana a disciplina precisa estar alocada, cada dia abaixo do mínimo é contado como uma violação, retornando o número total de violações.

Na função objetivo é atribuído peso a cada violação, São os pesos na respectiva ordem de apresentação das funções: 1, 1, 2 ,5.

3.5. Condição de parada

A execução da metaheurística GRASP finaliza quando uma condição de parada específica é atingida. Isso pode acontecer após um número máximo de iterações, por limite de tempo, após um certo número de iterações sem melhoria.

Neste trabalho, será utilizado o número máximo de iterações. Dado que estão sendo utilizadas 21 instâncias diferentes e que não se conhece o melhor valor da função objetivo dessas instâncias.

A Tabela 4 abaixo apresenta o cronograma de horários, obtidos a partir dos dados fornecidos pela melhor solução encontrada na execução da instância de teste Toy. O melhor valor alcançado da função objetivo foi 4.

Tabela 4 - Resultado da Instância Toy

Curriculo 1					
	dia 0	dia 1	dia 2	dia 3	dia 4
aula 0		TecCos - rC		TecCos - rC	TecCos - rC
aula 1	ArcTec - rB	SecCosC - rA	TecCos - rC	TecCos - rC	
aula 2	ArcTec - rB	ArcTec - rB			
aula 3	SecCosC - rA				SecCosC - rA

Curriculo 2					
	dia 0	dia 1	dia 2	dia 3	dia 4
aula 0	GeoTec - rA	TecCos - rC		TecCos - rC	TecCos - rC
aula 1	GeoTec - rA		TecCos - rC	TecCos - rC	GeoTec - rA
aula 2		GeoTec - rA	GeoTec - rA		
aula 3					

Fonte: Autoria própria (2024)

4. RESULTADOS

Os experimentos computacionais foram realizados no Google Colaboratory ou Colab, onde é um ambiente de notebook interativo baseado em nuvem que permite aos usuários escrever e executar código em Python nos navegadores, sem a necessidade de instalação ou configuração de software. Algumas vantagens do Colab:

- **Acessibilidade:** O Colab pode ser acessado em qualquer lugar com uma conexão à internet, através do link https://colab.research.google.com/;
- **Facilidade de uso:** A interface do Colab é amigável e intuitiva, facilitando o uso para iniciantes e experientes em Python;
- **Recursos:** O Colab oferece acesso a GPUs e TPUs gratuitas, além de uma ampla variedade de bibliotecas Python pré-instaladas, como TensorFlow, PyTorch, NumPy, Pandas e Matplotlib;
- **Compartilhamento e colaboração:** O Colab permite que os usuários compartilhem seus notebooks com outros usuários e trabalhem em tempo real em um mesmo notebook.

A implementação do algoritmo GRASP foi realizada na linguagem Python. Para esse estudo foram utilizadas as 21 instâncias do problema de horários de curso, baseadas no currículo disponibilizado na competição ITC-2007.

4.1. Execução de Experimentos

A Tabela 5 apresenta os resultados finais obtidos pelo algoritmo de construção para as 21 instâncias do ITC-2007. As instâncias do ITC são divididas em três níveis de dificuldade: iniciais, atrasadas e ocultas. Para avaliar o algoritmo foram realizados 30 testes de execução com cada instância e para cada execução foi delimitado 100 iterações para o GRASP. Logo, o número máximo de soluções que podem ser geradas para cada instância é igual a 3.000.

Os dados apresentados na tabela são:

- **Melhor F(O):** melhor valor da função objetivo obtido com o algoritmo proposto;
- **Pior F(O):** pior valor da função objetivo, sendo melhor em cada execução;

- **Tempo Médio por Execução:** tempo médio em segundos de cada execução;
- **Máx. Soluções:** número máximo de soluções geradas;
- **Média Soluções:** número médio de soluções geradas por execução.

Vale salientar que todos os valores apresentados na tabela correspondem apenas a soluções que não violaram restrições rígidas.

Tabela 5 - Resultados do Algoritmo GRASP para as Instâncias do ITC-2007

Níveis	Instâncias	Melhor F(O)	Pior F(O)	Tempo Médio por Execução	Máximo Soluções	Média Soluções
Iniciais	Comp01	197	303	5,6746	3000	100
	Comp02	631	664	98,6002	3000	100
	Comp03	460	532	72,1154	3000	100
	Comp04	472	534	81,2690	3000	100
	Comp05	833	977	24,0773	3000	100
	Comp06	683	750	170,5551	3000	100
	Comp07	785	846	242,1582	3000	100
Atrasadas	Comp08	545	610	93,3769	3000	100
	Comp09	569	615	93,3238	3000	100
	Comp10	736	777	157,4129	3000	100
	Comp11	224	268	5,2122	3000	100
	Comp12	768	850	253,5427	3000	100
	Comp13	558	606	59,5259	3000	100
	Comp14	542	593	96,8408	3000	100
Ocultas	Comp15	498	531	72,8138	3000	100
	Comp16	723	773	143,3927	3000	100
	Comp17	682	727	71,9480	3000	100
	Comp18	358	391	23,5055	3000	100
	Comp19	573	616	84,7746	3000	100
	Comp20	764	835	215,4269	3000	100
	Comp21	668	728	123,1469	3000	100

Fonte: Autoria própria (2024)

Analisando os dados da Tabela 5, observa-se que, no geral, o tempo médio de execução das instâncias vai aumentando conforme o tamanho das instâncias.

Por exemplo, na Comp12 teve um tempo de 253,5427 segundos em relação a Comp1 que obteve 5,6746 segundos. Mesmo com a diferença de tempo foi possível encontrar soluções viáveis para todas as instâncias.

Embora não existam provas de que a heurística sempre encontre uma solução viável para qualquer instância, ela apresentou sucesso em todas as instâncias testadas neste trabalho, uma solução viável é facilmente encontrada. Portanto, é verdade que o algoritmo proposto é capaz de gerar soluções viáveis para o CB-CTP, somente violando restrições flexíveis.

4.2. Comparação e Análise dos resultados

A Tabela 6 traz a comparação entre os resultados do ITC-2007 gerados pelos participantes da competição na ordem de classificação: Muller, Lu, Atzunz, Clark, Geiger e a proposta deste trabalho. Os valores em negrito indicam os melhores resultados da competição

Tabela 6 - Resultados do ITC-2007

Instâncias	Muller	Lu	Atzunz	Clark	Geiger	Proposta
Comp01	**5**	**5**	**5**	10	**5**	197
Comp02	43	**34**	55	83	108	631
Comp03	72	**70**	91	106	115	460
Comp04	**35**	38	38	59	67	472
Comp05	**298**	**298**	325	362	408	833
Comp06	**41**	47	69	113	94	683
Comp07	**14**	19	42	73	75	785
Comp08	**39**	43	42	73	75	545
Comp09	103	**102**	109	130	153	569
Comp10	**9**	16	32	67	66	736
Comp11	**0**	**0**	**0**	1	**0**	224
Comp12	331	**320**	344	383	430	768
Comp13	66	**65**	75	105	101	558
Comp14	53	**52**	61	82	88	542
Comp15	84	**71**	82	128	119	498
Comp16	**34**	39	40	81	84	723
Comp17	**83**	91	102	124	152	682

Comp18	83	69	**68**	116	111	358
Comp19	**62**	65	75	107	111	573
Comp20	**27**	47	61	88	144	764
Comp21	**103**	106	123	174	169	668

Fonte: Autoria própria (2024)

Para avaliar o desempenho do algoritmo GRASP proposto neste estudo, foram conduzidas 30 execuções independentes para cada instância.

Analisando os resultados obtidos pode-se notar que o algoritmo proposto não gerou resultados satisfatórios em relação aos da competição, mas pode ter ocorrido devido está utilizando a função objetivo de um forma diferente sendo somente o somatório das penalidades das restrições flexíveis e utilizando mais restrições que a competição. Por exemplo, na instância Comp1 onde o melhor resultado da competição foi 5 e na proposta obteve 197. Portanto, verifica-se que o algoritmo proposto precisa de melhorias, seja na construção inicial usando outros critérios de ordenação, quanto na busca local para a melhora dos resultados obtidos usando outros algoritmos de busca.

5. CONCLUSÃO

Este trabalho apresentou um algoritmo GRASP usando SWAP na busca local para resolver o Problema de Horários de Cursos baseado em Currículo. A eficácia do algoritmo foi avaliada nas 21 instâncias do ITC-2007. Os resultados obtidos com a utilização da meta-heurística GRASP provam que o algoritmo é capaz de fornecer soluções viáveis, mas o algoritmo ainda precisa de ajustes para conseguir fornecer soluções de qualidade.

Observando a comparação e análise dos resultados obtidos pelo algoritmo proposto mostram que apesar de não gerar soluções melhores que os da competição, ainda gera soluções viáveis para o problema proposto levando em conta não violar as restrições rígidas, assim, o algoritmo precisa de melhorias para obter respostas e desempenho melhores sendo na busca local, quanto criando estrutura de dados mais eficientes.

As perspectivas para trabalhos futuros, entre eles:

- Investigar outros algoritmos de busca local;
- Melhorar o desempenho de alocação na fase da construção da solução;
- Executar o algoritmo por mais tempo ou mais iterações e usar critérios de parada diferentes;
- Criar estruturas de dados para que as alocações de aulas e o cálculo da função objetivo seja calculada de forma mais eficiente;
- Refatorar o código para otimizar o desempenho do algoritmo;
- Considerar mais restrições flexíveis, como alocação de aulas por turno e divisão de turmas com múltiplos professores, atendendo às necessidades do curso de Ciência da Computação da UERN.

REFERÊNCIAS

ANDRADE, P. R. d. L.; SCARPIN, C. T.; STEINER, M. T. A. Geração da grade horária do curso de engenharia de produção da UFPR através de programação linear binária. Anais do XLV Simpósio Brasileiro de Pesquisa Operacional, p. 1052–1063, 2012. Citado na página 34.

BEHESHTI, Z.; SHAMSUDDIN, S. M. H. A review of population-based meta-heuristic algorithms. Int. J. Adv. Soft Comput. Appl, v. 5, n. 1, p. 1–35, 2013. Citado nas páginas 34 e 35.

BORGES, A.; OSPINA, R.; CRISTINA, G.; LEITE, A. Binary integer programming model for university courses timetabling: a case study. XLVII Simpósio Brasileiro de Pesquisa Operacional, p. 2338–2345, 2015. Citado na página 34.

BUCCO, G. B.; BORNIA-POULSEN, C. J.; BANDEIRA, D. L. Desenvolvimento de um modelo de programação linear para o problema da construção de grades horárias em universidades. Gestão & Produção, v. 24, n. 1, p. 40–49, 2017. Citado na página 34.

CSIMA, J.; GOTLIEB, C. Tests on a computer method for constructing school timetables. Communications of the ACM, ACM New York, NY, USA, v. 7, n. 3, p. 160–163, 1964. Citado na página 15.

FEO, T.; RESENDE, M. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, v. 8, 1989. Citado na página 37.

FEO, T.; RESENDE, M.; SMITH, S. A greedy randomized adaptive search procedure for maximum independent set. Operations Research, v. 42, p. 860–878, 1994. Citado na página 37.

GOTLIEB, C. C. The construction of class-teacher timetables. In: Proceedings IFIP Congress. [S.l.: s.n.], 1963. v. 62, p. 73–77. Citado na página 15.

HABASHI, S. S.; SALAMA, C.; YOUSEF, A. H.; FAHMY, H. M. A. Adaptive diversifying hyper-heuristic based approach for timetabling problems. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). [S.l.: s.n.], 2018. p. 259–266. ISSN null. Citado nas páginas 35 e 36. Citado nas páginas 35 e 36.

JAENGCHUEA, S.; LOHPETCH, D. A hybrid genetic algorithm with local search and tabu search approaches for solving the post enrolment based course timetabling problem: Outperforming guided search genetic algorithm. In: 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE). [S.l.: s.n.], 2015. p. 29–34. ISSN null. Citado na página 23.

JARDIM, A. M.; SEMAAN, G. S.; PENNA, P. H. V. Um algoritmo para o problema de programação de horários: Um estudo de caso. XXII Simpósio de Engenharia de Produção (SIMPEP), 2015. Citado na página 35.

MATIAS, J. B.; FAJARDO, A. C.; MEDINA, R. M. Examining genetic algorithm with guided search and self-adaptive neighborhood strategies for curriculum-based course timetable problem. In: 2018 Fourth International Conference on Advances in Computing, Communication Automation (ICACCA). [S.l.: s.n.], 2018. p. 1–6. ISSN 2641-8134. Citado na página 36.

MENDES, R. F. d. S.; CONCATTO, F.; SANTIAGO, R. Desenvolvimento de um modelo exato de alocação de disciplinas no contexto de um curso de graduação. L Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro, 2018. Citado na página 34.

MIKUSKA, M. I. S. Uma proposta baseada em Algoritmo Genético para o Problema Timetable Escolar Compacto. Dissertação – Programa de Pós-Graduação em Métodos Numéricos em Engenharia UFPR, Curitiba, 103p, 2015. Citado na página 21.

NEUKIRCHEN, F. V. P.; DORNELES, Á. P.; WEBER, R. F.; BURIOL, L. S. Um estudo de caso sobre a geração de quadros de horários no departamento de ciência da computação da ufrgs. Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional, p. 3272–3279, 2014. Citado na página 34.

POULSEN, C. J. B.; BUCCO, G. B.; BANDEIRA, D. L. Uma proposta de programação matemática para o university course timetabling problem. Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional, p. 979–990, 2014. Citado na página 34.

PILLAY, N.; ÖZCAN, E. Automated generation of constructive ordering heuristics for educational timetabling. Annals of Operations Research, Springer, v. 275, n. 1, p. 181–208, Apr 2019. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/s10479-017-2625-x>. Citado na página 33.

QUEIROZ, D. L. d.; NEPOMUCENO, N. V. Um modelo em programação linear inteira para alocação de disciplinas: Um estudo de caso no curso de ciência da computação da universidade de Fortaleza. XLIX Simpósio Brasileiro de Pesquisa Operacional, p. 2914–2925, 2017. Citado na página 34.

RESENDE, M.; RIBEIRO, C. A GRASP with path-relinking for private virtual circuit routing. Networks, v. 41, n. 1, p. 104–114, 2003. Citado na página 37.

ROCHA, W. d. S. Algoritmo GRASP para o Problema de Tabela-horário de Universidades. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - ES, 2013. Citado nas páginas 27 e 37.

SCHAERF, A. A survey of automated timetabling. Artif. Intell. Rev., Kluwer Academic Publishers, Norwell, MA, USA, v. 13, n. 2, p. 87–127, abr. 1999. ISSN 0269-2821. Citado nas páginas 15 e 32.

SENA, Daniela Costa de. Uma abordagem híbrida para o problema de programação de horários de cursos universitários. Dissertação - Programa de Pós-Graduação em

Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, Mossoró, 72p, 2021. Citado na página 31.

SILVA, A. R. V. d. Uma formulação matemática para o problema da alocação de horários em um curso universitário: um estudo de caso. Anais do XLVI Simpósio Brasileiro de Pesquisa Operacional, p. 2704–2715, 2014. Citado na página 34.

SILVA, I. L. d.; CAMPOS, S. C. Programação inteira para timetabling em uma universidade privada. XLIX Simpósio Brasileiro de Pesquisa Operacional, p. 798–809, 2017. Citado na página 34.

SUSAN, S.; BHUTANI, A. A novel memetic algorithm incorporating greedy stochastic local search mutation for course scheduling. In: 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). [S.l.: s.n.], 2019. p. 254–259. ISSN null. Citado na página 36.

TEOH, C.; ABDULLAH, M. Y. C.; HARON, H. Effect of pre-processors on solution quality of university course timetabling problem. In: 2015 IEEE Student Conference on Research and Development (SCOReD). [S.l.: s.n.], 2015. p. 472–477. ISSN null. Citado na página 23.

VRIELINK, R. A. O.; JANSEN, E. A.; HANS, E. W.; HILLEGERSBERG, J. van. Practices in timetabling in higher education institutions: a systematic review. Annals of operations research, Springer, v. 275, n. 1, p. 145–160, 2019. Citado na página 35.

WAHID, J.; HUSSIN, N. M. Construction of initial solution population for curriculum-based course timetabling using combination of graph heuristics. Journal of Telecommunication, Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, v. 8, n. 8, p. 92–95, 2016. Citado na página 37.