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

LIZA ALEXANDRA VIEIRA MAGALHÃES DE AZEVEDO

APLICAÇÃO DO ALGORITMO GENÉTICO NA SOLUÇÃO DO PROBLEMA DE AGENDAMENTO DE HORÁRIO DE AULAS UNIVERSITÁRIAS

MOSSORÓ – RN

2024

LIZA ALEXANDRA VIEIRA MAGALHÃES DE AZEVEDO

APLICAÇÃO DO ALGORITMO GENÉTICO 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

LIZA ALEXANDRA VIEIRA MAGALHÃES DE AZEVEDO

APLICAÇÃO DO ALGORITMO GENÉTICO 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.

Aprovada 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, Dona Antonia, por sempre acreditar em mim e compreender minhas ausências. Dedico também a meu esposo, André Viana, por todo apoio e companheirismo ao longo dessa jornada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que até aqui me sustentou e me inspirou. A Ti dedico sempre todas as minhas conquistas.

Agradeço a mim, por ter tido a coragem de abrir mão de um emprego fixo e de outra graduação para recomeçar do zero em uma área totalmente diferente. Transição de carreira não é algo fácil, mas preferi encarar meus medos e correr atrás de algo onde eu me sentisse mais feliz profissionalmente.

Agradeço a meu esposo, André, por ter sido meu maior incentivador desde sempre. Foi observando sua jornada durante a graduação e sua rotina como profissional já formado, que eu criei coragem para trilhar o mesmo caminho. Nos primeiros períodos do curso eu estava desempregada, mas ele me deu todo o suporte psicológico e financeiro para que eu seguisse em frente. Essa conquista não é só minha, é dele também.

Agradeço a minha mãe, Dona Antonia, pela paciência e pela confiança em mim. Em diversos momentos precisei me distanciar de tudo e todos para focar nos meus sonhos e alcançar meus objetivos, e ela sempre foi compreensiva e me apoiou.

Agradeço ao meu orientador, Prof. Dr. Francisco Chagas de Lima Júnior, por ter aceitado o convite para me orientar neste trabalho, por ter disponibilizado seu tempo e por ter compartilhado tanto do seu conhecimento comigo.

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

E por fim, mais não menos importante, gostaria de agradecer a cada um dos amigos que fiz ao longo da graduação. Graças a eles, os dias foram mais leves, as cargas divididas e o curso de computação, que para muitos é um “bicho papão”, se tornou menos assustador. Entre eles, destaco: Carolayne Barreto, Fagner Silva, Paulo Fanin, Márcio Vinicius, Pâmela Santos, Alber Neto, Igor Figueiredo e o principal, que desde as aulas remotas foi meu maior amigo e parceiro, Isaú Lucas.

A todos que contribuíram diretamente ou indiretamente, meu mais sincero Obrigado.

“Não creias impossível o que apenas improvável parece.”

William Shakespeare

RESUMO

O Problema do Agendamento de Horários de Aulas, conhecido como "Timetabling Problem" ou UCTP, é um desafio de otimização combinatória recorrente em instituições de ensino superior e que requer a alocação eficiente de horários para disciplinas considerando diversas restrições. Este problema é caracterizado como NP-Completo e NP-Difícil, sendo tradicionalmente abordado por métodos heurísticos como o Algoritmo Genético, buscando soluções de alta qualidade em tempo computacional razoável. Este estudo propõe a automatização do agendamento de horários de aulas no contexto universitário, utilizando o Algoritmo Genético e considerando as peculiaridades do ITC-2007 para lidar com restrições como disponibilidade de professores, carga horária das disciplinas e capacidade das salas de aula, com o objetivo de aprimorar a eficiência e equidade do serviço oferecido à comunidade acadêmica. A solução proposta foi aplicada às instâncias fornecidas pelo ITC-2007.

Palavras-chave: Otimização Combinatória, Problema do Agendamento de Horários de Aulas, Algoritmo Genético, ITC.

ABSTRACT

The Class Scheduling Problem, also known as "Timetabling Problem" or UCTP, is a recurring combinatorial optimization challenge in higher education institutions, requiring efficient scheduling of class times considering various constraints. This problem is characterized as NP-Complete and NP-Hard, traditionally addressed by heuristic methods such as Genetic Algorithm, aiming to find high-quality solutions within reasonable computational time. This study proposes the automation of class scheduling for the universitary context, using the Genetic Algorithm and considering the peculiarities of ITC-2007 to handle constraints such as teacher availability, course workload, and classroom capacity, with the goal of enhancing the efficiency and fairness of the service provided to the academic community. The proposed solution was applied to instances provided by ITC-2007.

Keywords: Combinatorial Optimization, Class Scheduling Problem, Genetic Algorithm, ITC.

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 do agendamento de horários de aulas, também conhecido como "Timetabling Problem" ou UCTP, é um desafio de otimização onde existe a necessidade de alocação de horários para disciplinas em um curso levando em consideração múltiplas restrições, tais como a disponibilidade dos docentes, as interdependências entre as disciplinas e as preferências dos estudantes (SUSAN; BHUTANI, 2019; QUEIROZ; NEPOMUCENO, 2017).

Este problema é um clássico de otimização combinatória e é um tema frequentemente abordado por inúmeros pesquisadores e estudiosos da área. Além disso, ele é um problema recorrente nas instituições de ensino superior, que precisam lidar a cada semestre com a montagem de novos horários de aulas considerando diversas variáveis, como cursos, disciplinas, salas de aulas disponíveis, dentre outros. Ou seja, existe a necessidade de que sejam propostas soluções cada vez mais eficazes e utilizando diferentes abordagens, que podem variar dependendo da forma como é modelado o problema.

Computacionalmente, o UCTP (do inglês *University Course Timetabling Problem*) pode ser considerado do tipo NP-Completo e NP-Difícil (TEOH; ABDULLAH; HARON, 2015; POULSEN; BUCCO; BANDEIRA, 2014). Ou seja, a busca pela solução ótima consome um tempo exponencial, no entanto, podem ser empregados algoritmos heurísticos que conseguem encontrar soluções satisfatórias em um período de tempo razoável.

Dado que no problema do agendamento de horários de aulas não é preciso encontrar encontrar uma solução ótima, mas sim uma solução de alta qualidade com um custo computacional substancialmente menor, pode-se dizer que uma metaheurística eficaz para abordar esse desafio é o Algoritmo Genético (GA).

Os Algoritmos Genéticos são uma técnica de otimização que aplica de forma sofisticada conceitos emprestados da Teoria Evolutiva da Biologia e se fundamenta nos princípios do cruzamento e da mutação (ALIXANDRE; DORN, 2017).

Esses mecanismos permitem que as soluções para um problema evoluam ao longo das gerações. Ao selecionar os indivíduos mais aptos através do cruzamento dos pais, os Algoritmos Genéticos favorecem a perpetuação de características

benéficas, aumentando as chances de se alcançar uma solução de qualidade. Além disso, as variações genéticas aleatórias, conhecidas como mutação, introduzem diversidade no conjunto de soluções, potencialmente gerando novas combinações genéticas que podem evoluir em direção a soluções ainda melhores.

Desenvolvidos por John Holland nos anos 1960 e aprimorados na década de 1970, os Algoritmos Genéticos são inspirados na teoria da evolução de Darwin. Eles operam utilizando estruturas de dados chamadas cromossomos, que representam possíveis soluções para um problema. Aplicados em uma ampla gama de problemas de otimização, desde funções matemáticas até questões práticas como o problema do caixeiro viajante e otimização de rotas, esses algoritmos funcionam através de processos de avaliação, seleção, recombinação e mutação dos cromossomos. O processo é iterativo, continuando até que uma condição de parada seja alcançada.

O agendamento de horários de aulas é de extrema relevância para as instituições de ensino superior, uma vez que um calendário eficaz pode influenciar positivamente na qualidade do ensino, na satisfação tanto de alunos quanto de professores, bem como na eficiência da gestão acadêmica. No entanto, a criação manual de horários é uma tarefa complexa e morosa, frequentemente resultando em soluções improvisadas que podem se mostrar ineficazes, desagradando os envolvidos.

1.1. Objetivos

Este trabalho propõe a automatização do agendamento de horários de aulas no contexto universitário. Essa automatização empregará o Algoritmo Genético em sua versão clássica com o objetivo de identificar soluções aproximadas para o problema.

A automatização de geração desses horários será capaz de lidar com diversas restrições do problema, dentre elas:

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

Acreditamos que esta iniciativa será uma valiosa contribuição para aprimorar o serviço de agendamento de horários de aulas oferecido à comunidade acadêmica, dado que será possível criar horários mais equitativos e eficazes, conferindo benefícios tangíveis 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. Logo, irá apresentar os conceitos essenciais que embasam a pesquisa, abordando tanto o problema de agendamento de horários de aulas quanto os princípios dos algoritmos genéticos. É também nesse capítulo que será formulado o problema, apresentadas as suas restrições, determinada a função objetivo, as instâncias utilizadas para teste e a validação das soluções.

O Capítulo 3 foca na abordagem proposta. Aqui será detalhada a estratégia utilizada para resolver o problema e serão discutidos os métodos empregados na nossa abordagem, justificando suas escolhas com base em fundamentações teóricas. Além disso, serão apresentados detalhes do Algoritmo Genético, explicando como ele foi implementado para solucionar o problema de agendamento de horários de aulas desse caso específico.

O Capítulo 4 concentra-se nos resultados e na avaliação. Aqui, serão apresentados os resultados obtidos ao longo da pesquisa, incluindo as etapas de construção (inicial) e melhoria (final) do algoritmo. Realizamos uma análise crítica dos resultados e sua interpretação à luz dos objetivos da pesquisa.

No Capítulo 5, concluímos o trabalho, resumindo as principais conclusões derivadas dos resultados. Além disso, destacamos as perspectivas futuras, delineando possíveis direções para pesquisas subsequentes e melhorias na solução proposta.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. O Problema do Agendamento de Horários

O problema de agendamento de horários é uma questão complexa de otimização que implica na distribuição eficiente de recursos dentro de intervalos de tempo definidos, a fim de cumprir determinadas tarefas ou compromissos e alcançar metas específicas. Esse desafio é de grande importância em várias áreas e pode ser aplicado em diversos contextos, como gestão de recursos humanos, programação de produção, logística e transporte, planejamento de eventos, cuidados de saúde, entre outros.

Os primeiros estudos sobre esse tema remontam a trabalhos de Gotlieb (1963) e Csima e Gotlieb (1964), e obras como "Introduction to the Theory of Scheduling" e "A Survey of Automated Timetabling" têm contribuído significativamente para o entendimento do assunto. Questões educacionais também são abordadas em publicações como "Time and School Learning". Em 1999, Schaerf estabeleceu três classificações principais para o problema de programação de horários educacionais, que auxiliam na compreensão da diversidade de desafios nesse campo:

- **Problema de Horários de Escolas (*School Timetabling Problem*):** envolve a distribuição semanal dos horários das turmas de uma escola, garantindo que professores não estejam em mais de uma turma ao mesmo tempo e evitando aulas simultâneas com mais de um professor.
- **Problema de Horários de Cursos Universitários (*University Course Timetabling Problem*):** relaciona-se à alocação de aulas das disciplinas em uma instituição de ensino superior, considerando disponibilidade e capacidade das salas, bem como permitindo que alunos escolham turmas de disciplinas de seus currículos, que podem ser compostas por alunos de diferentes currículos.
- **Problema de Horários de Exames (*Examination Timetabling Problem*):** trata da alocação de exames para cursos universitários, evitando conflitos de horários entre exames que envolvam estudantes em comum.

Devido à sua complexidade, esse problema apresenta vários desafios, como questões computacionais, restrições específicas, objetivos conflitantes, dinamismo das situações, incerteza nos dados, escala do problema, dimensionamento adequado de recursos e satisfação das partes interessadas. Para enfrentar esses desafios, uma variedade de abordagens tem sido desenvolvida, incluindo algoritmos de otimização, métodos de programação linear e técnicas de inteligência artificial, como algoritmos genéticos, busca local e aprendizado de máquina. A escolha da abordagem apropriada depende da natureza específica do problema, do contexto de aplicação e das restrições envolvidas.

É fundamental resolver de maneira eficiente o problema de agendamento de horários educacionais para assegurar que os alunos recebam uma educação de qualidade, prevenindo conflitos de horários e garantindo a utilização eficaz dos recursos da instituição.

Neste trabalho, concentramos nossa atenção no Problema de Horários de Cursos Universitários (*University Course Timetabling Problem*), uma vez que ele se assemelha mais à nossa realidade e contexto.

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

O ITC-2007, conhecido como *International Timetabling Competition*, foi um evento realizado em 2007 que teve um papel fundamental na promoção da pesquisa e no desenvolvimento de novos algoritmos para resolver desafios reais de programação de horários. Esta competição abrangeu três trilhas distintas:

- **Trilha 1:** Problema de Agendamento de Exames
- **Trilha 2:** Problema de Agendamento de Cursos Pós-Matrícula (PEB-CTP)
- **Trilha 3:** Problema de Agendamento de Cursos Baseado no Currículo (CB-CTP)

Cada uma dessas trilhas tinha seu próprio conjunto de instâncias de problemas, e os participantes foram convidados a submeter suas soluções para avaliação. Os vencedores de cada trilha foram determinados com base na qualidade de suas soluções, e essa qualidade foi medida a partir de um conjunto de métricas

de desempenho predefinidas. Além disso, o ITC também classificou o UCTP em duas subcategorias, que também podem funcionar como uma definição para os problemas das trilhas 1 e 2. São elas:

- **Grade Horária do Curso Baseada na Pós-Matrícula (PEB-CTP ou *Post Enrolment Based Course Timetabling Problem*):** Neste caso, o agendamento dos horários ocorre após a matrícula dos alunos. Isso implica a alocação de eventos letivos (aulas) para horários e salas específicas, levando em consideração as escolhas dos alunos e as disponibilidades. Os alunos têm a flexibilidade de selecionar as turmas das disciplinas de acordo com suas preferências, e a instituição deve acomodar essas escolhas dentro das limitações dos recursos disponíveis.
- **Grade Horária do Curso Baseada no Currículo (CB-CTP ou *Curriculum Based Course Timetabling Problem*):** Neste caso, o agendamento é feito considerando as necessidades de várias disciplinas que fazem parte do currículo de diferentes cursos universitários. Isso envolve a atribuição semanal de um conjunto de aulas para várias disciplinas, alocando-as para horários e salas específicas, considerando as capacidades das salas e as demandas de várias disciplinas em diferentes currículos.

Os algoritmos de programação de horários desenvolvidos durante o ITC-2007 ainda são amplamente utilizados atualmente por várias instituições, incluindo universidades e escolas, para resolver desafios reais de agendamento. Neste estudo, concentramos nossa atenção especificamente no desafio de Programação de Cursos Baseada no Currículo (CB-CTP).

A principal razão por trás deste estudo é o fato de que, atualmente, o agendamento das aulas para as turmas de Ciência da Computação no Departamento de Informática da UERN - Campus Central é feito manualmente. Portanto, a modelagem do problema em questão será realizada levando em consideração o contexto dos dados deste departamento.

Para entender melhor o processo existente, foi realizada uma entrevista com o técnico responsável pela elaboração do horário, que descreveu detalhadamente como esse procedimento é conduzido. Nesse sentido, será apresentada a seguir

uma descrição minuciosa desse processo, as restrições que devem ser consideradas e a solução proposta para o problema.

A implementação do Algoritmo Genético para o Problema de Otimização de Horários será validada neste trabalho com instâncias genéricas conseguidas na literatura, mas a utilização do método com instâncias 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

Considerando a montagem do horário (ou agendamento de aulas) a cada semestre, é essencial compreender que ele serve como base para o semestre subsequente. O agendamento segue uma estrutura que abrange os seguintes aspectos:

- **Divisão por Turmas**: As turmas representam períodos (ou semestres) do curso. Em períodos pares, as turmas são designadas para os semestres pares, e o mesmo vale para os períodos ímpares. Por exemplo, no semestre 2023.1, as turmas dos períodos I, III, V e VII do curso de Ciência da Computação foram oferecidas. Já no semestre 2023.2, as turmas dos períodos II, IV, VI e VIII estão sendo oferecidas. Importante notar que o curso de Ciência da Computação abrange um total de 8 períodos, cada um com disciplinas ministradas por diferentes professores;

- **Divisão por Dias da Semana:** As aulas podem ser agendadas de segunda a sábado para diferentes turmas. Neste contexto, abordaremos apenas os dias úteis da semana, excluindo os sábados letivos, já que estes são reservados para reposição de aulas e não afetam o planejamento do horário;

- **Divisão por Horários:** As aulas ocorrem em intervalos de tempo predefinidos, que não sofrem alterações a cada semestre. O que varia é apenas a alocação das combinações de disciplina/professor dentro desses horários. Além disso, a quantidade de horários em cada dia da semana é uniforme, de modo que o total disponível em uma semana pode ser determinado multiplicando-se a quantidade de horários em um dia pelo

número de dias da semana;

- **Identificação das Aulas:** Cada aula é identificada pela combinação de disciplina e professor(es) responsável(is). As disciplinas podem ser classificadas como obrigatórias ou optativas. Disciplinas obrigatórias são aquelas que os alunos devem cursar para concluir o curso, enquanto as disciplinas optativas oferecem aos alunos a oportunidade de escolher entre várias opções.

A estrutura do horário baseada no CB-CTP (Curso Baseado no Currículo) desempenha um papel crítico na organização e planejamento das atividades acadêmicas, garantindo que os alunos tenham acesso às disciplinas necessárias para sua formação, ao mesmo tempo em que oferece flexibilidade por meio das disciplinas optativas. A otimização deste processo é de fundamental importância para garantir uma experiência acadêmica eficiente 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)

Na Tabela 1, é apresentado o horário de aulas de uma turma de Ciência da Computação da Universidade do Estado do Rio Grande do Norte (UERN). Nela, é possível observar que especificamente essa turma pode ter aulas em até sete horários diferentes, dependendo do dia da semana. Além disso, as disciplinas oferecidas em cada horário e os respectivos professores responsáveis estão indicados.

Uma observação interessante é que, na quarta-feira, nos dois primeiros horários, essa turma tem aulas de duas disciplinas distintas, ministradas por dois professores diferentes. Isso é possível devido ao fato de essas disciplinas serem do tipo optativa. Ou seja, os alunos têm a liberdade de escolher qual delas desejam cursar, ou até mesmo optar por não cursar nenhuma delas e utilizar esse tempo livre para outras atividades.

Seguindo a lógica de que as turmas são organizadas com base no indicador de semestre (par ou ímpar), a montagem do horário do próximo semestre é inicialmente feita replicando o horário do semestre anterior. Por exemplo, o horário do semestre 2023.2 é baseado no horário do semestre 2022.2.

Posteriormente, o técnico entra em contato individualmente com cada professor para verificar se eles podem continuar ministrando as mesmas disciplinas nos mesmos dias e horários do agendamento anterior. Se um professor não estiver mais disponível, é necessário que ele informe sua disponibilidade, possibilitando, assim, o ajuste manual do horário pelo técnico.

Esse ajuste envolve a verificação de horários disponíveis que atendam à disponibilidade do professor para realocar suas aulas. Caso não haja horários vagos, o técnico precisa investigar se outro professor está disposto a efetuar uma troca. O horário será considerado eficaz quando puder ser viável, ou seja, quando professores e alunos que o utilizarem ficarem satisfeitos, pelo menos em certa medida.

O Departamento de Informática (DI) atualmente conta com 19 professores, embora haja a possibilidade de convidar professores de outros departamentos para ministrar aulas no curso, caso haja demanda. Além disso, a extensa grade de disciplinas do curso permite a oferta de até 28 disciplinas em um único semestre. Assim, a montagem manual do horário é um processo trabalhoso, suscetível a erros, e, de acordo com o técnico responsável, pode demandar cerca de 30 dias.

Para solucionar esse desafio, é essencial otimizar o processo. Para isso, será empregado um algoritmo genético, 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

De acordo com Pillay (2013), o *timetabling problem* consiste em alocar professores, turmas e salas em espaços de tempo a fim de satisfazer as restrições fortes e fracas de um problema. Uma restrição nada mais é do que uma condição que precisa ser atendida dentro do contexto do problema. Além disso, ela pode ser classificada como rígida ou flexível.

Segundo Mikuska (2015), são consideradas restrições fortes (ou rígidas) as condições que afetam a viabilidade da solução. Já as soluções fracas (ou flexíveis) são as condições que afetam a qualidade da solução proposta. O ideal é que nenhuma restrição rígida seja violada, já que isso poderia inviabilizar a solução. Já as restrições flexíveis, cujo atendimento é apenas desejável, podem ser violadas, pois apesar disso gerar uma solução de baixa qualidade, ainda assim ela seria executável.

No contexto desse trabalho, serão adotadas as seguintes restrições:

- **Restrições Rígidas**
    - **Dias de aula:** As aulas devem ocorrer apenas nos dias determinados pela instituição. Essa restrição é violada quando a instância indicar que as aulas devem ser alocadas em 5 dias, e o algoritmo agendar em mais de 5;
    - **Horário da aula:** As aulas devem ocorrer apenas nos horários determinados pela instituição. Essa restrição é violada quando a instância indicar que as aulas devem ser agendadas em 6 horários, e o algoritmo alocar em mais de 6;
    - **Alocação da disciplina:** Todas as aulas de uma disciplina devem ser distribuídas em horários diferentes. Esta restrição é violada quando uma disciplina não tem todas as suas aulas agendadas ou quando duas disciplinas diferentes da mesma turma são agendadas no mesmo horário;

- **Ocupação da sala:** Duas aulas não podem acontecer simultaneamente na mesma sala. Esta restrição é violada quando duas aulas são agendadas na mesma sala e no mesmo horário;
- **Disponibilidade do professor:** Cada professor só pode ministrar uma aula por horário. Esta restrição é violada se uma aula for marcada em um horário em que o professor não está disponível, seja por escolha própria ou porque já está agendado para outra disciplina;
- **Conflitos**: Aulas de disciplinas da mesma turma ou ministradas pelo mesmo professor devem ser programadas em horários diferentes. Esta restrição é violada quando aulas da mesma turma ou do mesmo professor são agendadas no mesmo horário.
- **Combinação professor/disciplina:** Algumas disciplinas só podem ser lecionadas por professores especializados na área. Esta restrição é violada quando um professor é designado para ministrar uma disciplina para a qual não possui especialização.

- **Restrições Flexíveis**
  - **Capacidade da sala:** É importante garantir que uma sala não seja alocada para aulas de uma disciplina com um número de alunos matriculados que exceda sua capacidade. Essa regra é quebrada quando uma sala com capacidade para 25 alunos é designada para uma disciplina com 30 alunos, por exemplo. Cada aluno além da capacidade receberá uma penalidade de 1 ponto;
  - **Dias mínimos de trabalho:** É importante garantir que uma disciplina não tenha aulas alocadas em menos dias que o necessário. O objetivo é evitar que uma disciplina com muitas aulas tenha todas as suas aulas agendadas em um único dia. No caso da UERN, por exemplo, as disciplinas têm diferentes cargas horárias, podendo variar entre 15, 30, 45, 60 e 90 horas. Portanto, a distribuição dessas horas deve respeitar um número mínimo de dias de aula por semana, conforme descrito abaixo:

        - Disciplinas com 15h devem ter aulas em 1 dia, com 1 aula por semana;
        - Disciplinas com 30h devem ter aulas em 1 dia, com 2 aulas por semana;
        - Disciplinas com 45h devem ter aulas em 1 dia, com 3 aulas por semana;
        - Disciplinas com 60h devem ter aulas em 2 dias, com 4 aulas por semana;
        - Disciplinas com 90h devem ter aulas em 2 dias, com 6 aulas por semana.

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

    - **Ociosidade da turma:** É necessário garantir que as aulas de uma turma ocorram em horários consecutivos, evitando longos intervalos entre os horários finais e iniciais. Cada intervalo isolado entre aulas receberá uma penalidade de 2 pontos;
    - **Estabilidade da sala:** Todas as aulas de uma disciplina devem acontecer na mesma sala, evitando que alunos e professores tenham que se deslocar entre salas para a mesma disciplina. Cada sala adicional usada para as aulas da mesma disciplina receberá uma penalidade de 1 ponto.

Considerando que uma solução deve cumprir todas as restrições rígidas para ser considerada viável e que a qualidade da solução é determinada pelo cumprimento das restrições flexíveis, podemos concluir que a tabela de horário ideal é aquela que satisfaz todas as restrições rígidas e maximiza a satisfação das restrições flexíveis (JAENGCHUEA; LOHPETCH, 2015; TEOH; ABDULLAH; HARON, 2015).

2.2.3. Função Objetivo

A função objetivo é uma equação matemática usada na otimização para avaliar a qualidade de uma solução em relação a determinados critérios. Ela é

crucial em problemas de otimização, pois orienta a busca pela melhor solução. Para chegar à função objetivo, precisamos entender as restrições que uma solução deve atender para ser considerada viável.

No contexto deste trabalho, as restrições influenciam na qualidade e na viabilidade da alocação de horários. Violar as 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. Além disso, existem restrições flexíveis, como a alocação da carga horária e a estabilidade da turma, que afetam a qualidade da solução, mas podem ser violadas se necessário.

Levando em conta que as restrições rígidas já serão verificadas na etapa de construção e que as soluções que violarem tais restrições serão descartadas, logo, a função objetivo proposta visa atender apenas às restrições flexíveis. Ou seja, minimizar o número de lacunas (intervalos sem aula em um mesmo dia), garantir a distribuição adequada da carga horária das disciplinas e garantir que as aulas de uma mesma disciplina ocorram sempre na mesma sala. Dessa forma, a tabela de horário ótima vai ser a que atender a todas essas restrições e minimizar as lacunas, garantindo assim a eficiência do processo de alocação.

Abaixo, segue a função que será utilizada neste trabalho.

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

Onde:

- *F(O)* é a função objetivo;
- ∑ é o somatório das penalidades aplicadas quando cada restrição flexível é violada;
- $P_{capacidade\_sala}$ é uma penalidade de valor igual a 1 para cada aluno além da capacidade da sala;
- $P_{dias\_mínimos}$ é uma penalidade de valor igual a 5 para cada aula

alocada abaixo do número mínimo de dias de aula por semana;

- $P_{compacidade\_turma}$ é uma penalidade de valor igual a 2 para cada aula alocada em horários não consecutivos;

- $P_{estabilidade\_sala}$ é uma penalidade de valor igual a 1 para cada aula de uma mesma disciplina alocada em salas diferentes.

Essa função calcula o custo total de uma solução para o problema de alocação de aulas. O custo total será a soma de todas as penalidades, ou seja, cada restrição violada aumenta o valor da função objetivo de acordo com seu peso e deixa a solução menos viável.

2.2.4. Instâncias

Para ajudar os competidores, o ITC disponibilizou conjuntos de instâncias que simulam as características de universidades reais, permitindo a comparação entre diferentes métodos de resolução.

Tabela 2 - Informações sobre cada Instância do ITC-2007

Instância	Currículos	Salas	Disciplinas	Horários por dia	Dias
comp01	14	6	30	6	5
comp02	70	16	82	5	5
comp03	68	16	72	5	5
comp04	57	18	79	5	5
comp05	139	9	54	6	6
comp06	70	18	108	5	5
comp07	77	20	131	5	5
comp08	61	18	86	5	5
comp09	75	18	76	5	5
comp10	67	18	115	5	5
comp11	13	5	30	9	5
comp12	150	11	88	6	6
comp13	66	19	82	5	5
comp14	60	17	85	5	5
comp15	68	16	72	5	5
comp16	71	20	108	5	5
comp17	70	17	99	5	5
comp18	52	9	47	6	6
comp19	66	16	74	5	5
comp20	78	19	121	5	5
comp21	78	18	94	5	5

Fonte: Website do ITC-2007

Neste trabalho, foram utilizadas as mesmas instâncias que foram usadas pelos competidores do ITC-2007 para resolver o problema do CB-CTP, totalizando 21 instâncias com diversos níveis de dificuldade. A organização da competição garantiu que houvesse soluções viáveis para todas as instâncias, confirmadas por meio de testes. No entanto, não foram fornecidas informações sobre violações de restrições flexíveis, dado que estas poderiam variar dependendo da instituição.

A Tabela 2 resume os dados relevantes de cada uma das 21 instâncias, incluindo informações sobre o número de currículos, salas, disciplinas, horários diários e dias. Os valores para currículos, salas e disciplinas variam consideravelmente entre as instâncias, enquanto os dados relacionados aos horários diários e dias são praticamente uniformes, com exceção da instância comp11.

Também neste trabalho, será implementado uma versão do algoritmo genético que será avaliada utilizando as instâncias do ITC-2007.

A Figura 1 fornece uma representação detalhada de cada instância, dividida nas seguintes seções:

- **Cabeçalho:** Esta seção fornece dados gerais e descreve características fundamentais da instância antes de entrar em detalhes específicos sobre cursos (disciplinas), salas, currículos (turmas) e restrições;
- **Cursos (Courses):** Esta seção fornece o total de cursos (ou disciplinas) envolvidos. Na Figura 1 é possível observar respectivamente o nome da matéria, 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 matéria;
- **Salas (Rooms):** Esta seção fornece as salas disponíveis. Os dados da Figura 1 apresentam respectivamente o nome da sala e sua capacidade;
- **Currículos (Curricula):** Esta seção fornece os currículos (ou turmas), que na Figura 1 são 2. Os dados exibem respectivamente o nome do currículo (turma), a quantidade e a lista de matérias da turma;
- **Restrições (Unavailability_Constraints):** Esta seção fornece as restrições

ou limitações que precisam ser consideradas no problema. Na instância da Figura 1, existem 8 restrições, e para cada uma delas são mostrados respectivamente o nome da matéria, o dia e o período (horário) do dia em que ela não pode ocorrer.

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: Website do ITC-2007

2.2.5. Arquivo de Saída

O arquivo de saída é um documento único que resulta da aplicação do algoritmo desenvolvido para resolver o problema de agendamento de horários educacionais. Ele adere ao formato do arquivo de saída utilizado na ITC-2007 e inclui a solução viável, representada por linhas que especificam as alocações de aulas. Essas alocações são determinadas usando a função objetivo descrita na seção 2.2.4 deste estudo. As linhas de alocação de aulas seguem este formato:

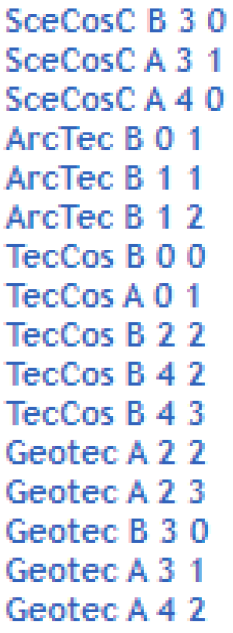
<Nome da disciplina> <Nome da Sala> <Dia> <Período do Dia>

Vale ressaltar que, considerando que os dias e períodos iniciam em 0, por exemplo, "TecCos A 2 3" indica que a disciplina "TecCos" será lecionada na quarta-feira (2) no quarto período (3) na sala A.

2.2.6. Validação das Soluções

Além de fornecer as instâncias modelo, o ITC-2007 também disponibilizou um validador com código fonte em C++ para verificar as soluções desenvolvidas pelos competidores.

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



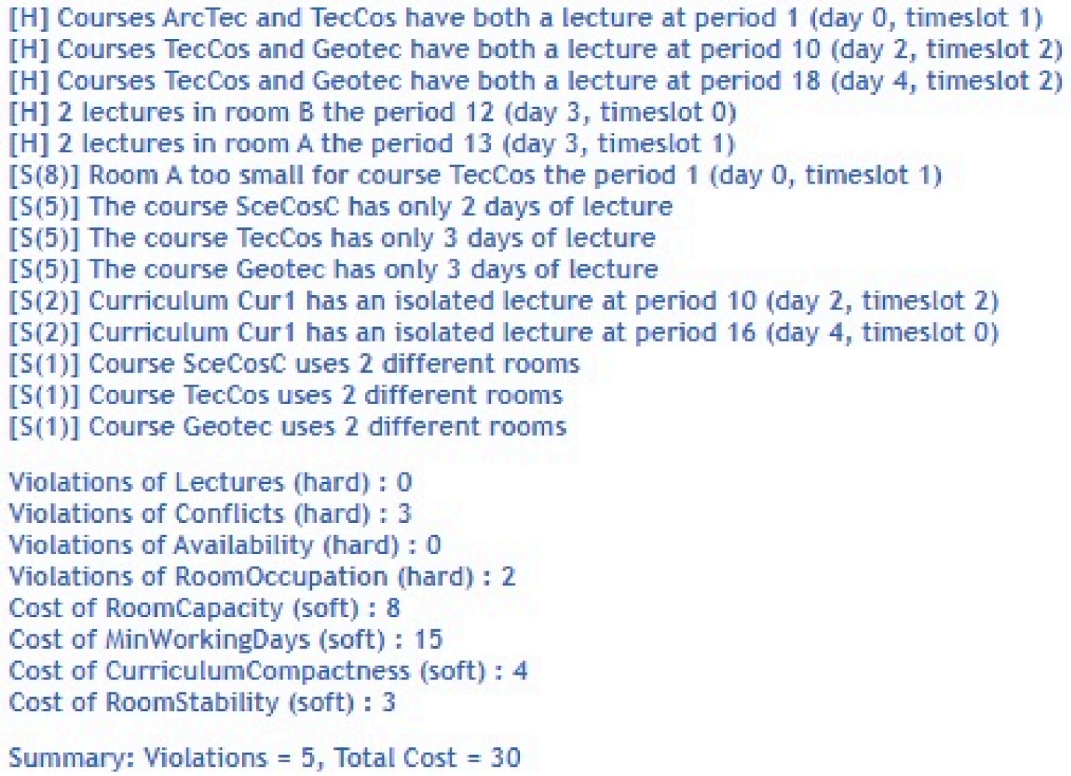
Fonte: Website do ITC-2007

O procedimento do validador é simples: inicialmente, ele lê o arquivo da

instância (arquivo de entrada) e o arquivo contendo a solução (arquivo de saída). Em seguida, produz uma saída padrão com a avaliação da solução e uma descrição detalhada de todas as violações de restrições rígidas e flexíveis.

Para uma melhor compreensão, a Figura 2 apresenta um exemplo de solução obtida a partir de uma instância, enquanto a Figura 3 mostra um exemplo da saída padrão gerada pelo validador.

Figura 3 - Saída do validador



Fonte: Website do ITC-2007

2.3. Abordagens para resolver Problemas de Programação de Horários Educacionais

Conforme mencionado no capítulo 1, diversos pesquisadores atuam no campo da otimização, empenhando-se na busca das soluções mais eficazes para resolver os desafios associados à programação de horários educacionais. Como resultado desse esforço, a literatura oferece uma variedade de métodos computacionais que abordam essa problemática, incluindo heurísticas, meta-heurísticas, abordagens híbridas, hiper-heurísticas, entre outros.

Nas subseções a seguir, abordaremos o uso de Heurísticas Construtivas, que

se concentram na progressiva construção de soluções iniciais, e as Abordagens de Otimização, que empregam técnicas diversas para aprimorar a qualidade das soluções encontradas. Isso engloba otimização de funções objetivas e a aplicação de algoritmos de busca local, com o objetivo de obter horários mais eficientes e adequados.

2.3.1. Heurísticas Construtivas

As heurísticas construtivas são métodos de resolução de problemas que se inspiram na abordagem humana para resolver desafios, construindo gradualmente uma solução final a partir de um estado inicial. Isso implica adicionar elementos de maneira incremental, com base em critérios específicos, até alcançar uma solução completa ou atender a um critério de parada predefinido.

Existem diversas abordagens para resolver o problema de programação de horários educacionais. Uma delas é o uso de algoritmos genéticos para automatizar o processo de alocação de horários. Outra alternativa é a utilização de algoritmos híbridos, os quais combinam diferentes metaheurísticas na busca da melhor solução possível. Como exemplo dessa segunda abordagem, podemos mencionar o trabalho de SENA (2021), que empregou 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 a resolução do problema. No entanto, é crucial estabelecer um ponto de partida: uma solução inicial que será iterativamente ajustada até alcançar a solução final ou a melhor solução viável.

As heurísticas construtivas podem ser categorizadas como aleatórias, gulosa ou parcialmente gulosa. Na construção aleatória, a solução é desenvolvida passo a passo, com a adição de elementos escolhidos aleatoriamente até a conclusão. Essa aleatoriedade pode 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, ao invés de escolher aleatoriamente, a heurística seleciona o elemento que parece ser a melhor escolha naquele momento, com base em alguma função de avaliação local. Isso geralmente resulta em soluções rápidas, embora não garanta a melhor solução global. Por fim, a heurística parcialmente gulosa é uma combinação das estratégias aleatória e gulosa. Ela usa a heurística gulosa para a maioria das decisões, mas ocasionalmente faz escolhas aleatórias

para introduzir diversidade na busca. Isso pode ajudar a evitar ficar preso em mínimos locais e a explorar mais amplamente o espaço de solução. Devido a isso, a heurística parcialmente gulosa frequentemente encontra soluções melhores em comparação com as heurísticas puramente gulosa, sem aumentar significativamente o tempo de execução. A finalização dos métodos ocorre quando todos os elementos são adicionados ou quando um critério de parada é satisfeito.

No contexto dos horários educacionais, essas técnicas podem ser aplicadas para preencher progressivamente um quadro de horários, inserindo uma aula de cada vez até que todas as aulas sejam alocadas ou até que um conflito seja identificado (SCHAERF, 1999).

Dentro dessas abordagens, ainda é possível termos heurísticas para o agendamento de eventos, onde cada uma possui sua própria lógica de priorização. São elas:

- **Maior Matrícula (LE):** Os eventos são ordenados em ordem decrescente com base no número de alunos matriculados. Eventos com mais alunos têm prioridade.
- **Maior Grau (LD):** Os eventos são priorizados com base no número de conflitos que têm com outros eventos. Quanto mais conflitos, maior a prioridade.
- **Maior Grau Ponderado (LWD):** Similar à LD, mas considera o número de alunos envolvidos em ambos os eventos, em vez de apenas contar o número de eventos com conflitos.
- **Maior Grau de Cor (LCD):** Variação da LD que considera o número de conflitos com eventos já agendados.
- **Grau de Saturação (SD):** Prioriza eventos com menos períodos disponíveis, levando em conta restrições rígidas do cronograma.

Essas heurísticas podem ser ainda classificadas em 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 ao longo do processo de construção. Todos os eventos são ordenados com base nesses valores

e são alocados em ordem em períodos viáveis. Se não houver períodos viáveis, o evento pode não ser alocado ou ser alocado aleatoriamente, aumentando o custo da restrição rígida. Já nas dinâmicas (LCD e SD), os valores heurísticos de cada evento não alocado mudam à medida que os eventos são alocados no cronograma. No início do processo, todos os eventos têm o mesmo valor heurístico. Conforme o cronograma é construído, o valor heurístico de um evento que compartilha alunos com um evento recém-alocado é ajustado. Por exemplo, no caso da heurística SD, o grau de saturação de um evento diminui à medida que o cronograma é construído, enquanto na heurística LCD, o valor heurístico de um evento não alocado é incrementado conforme eventos são alocados (PILLAY; ÖZCAN, 2019; PILLAY, 2016a).

Dentro das heurísticas estáticas, há três maneiras de selecionar o período para um evento:

- **Primeiro Período (FP):** O evento é colocado no primeiro período disponível que não viole as regras rígidas;
- **Período Aleatório (RP):** O período é escolhido de forma aleatória entre todos os períodos disponíveis;
- **Período de Custo Mínimo (MCP):** Cada período disponível é avaliado quanto ao custo associado à violação das regras flexíveis no cronograma atual. O evento é alocado no período com o menor custo de violação.

De modo geral, cada heurística tem sua própria abordagem para determinar a ordem de agendamento dos eventos, com base em diferentes critérios como número de alunos, conflitos ou períodos disponíveis. A abordagem escolhida para construção da solução inicial deste trabalho será abordada na subseção 3.2 do capítulo 3.

2.3.2. Abordagens de Otimização

Na literatura, encontramos diversas abordagens para a resolução de problemas de otimização. Esses métodos de resolução podem ser classificados em dois grupos principais: métodos exatos e métodos aproximados.

Os métodos exatos têm como objetivo encontrar a solução ideal para o

problema, garantindo uma resposta precisa e livre de erros. No contexto da programação de horários educacionais, isso significa criar um cronograma que atenda a todas as restrições de maneira ideal. Alguns métodos exatos incluem programação linear inteira (ILP), programação inteira-mista (MIP) e programação de restrições (CP), que podem ser aplicados para garantir uma solução precisa, embora possam ser computacionalmente intensivos.

Embora permitam encontrar a solução ideal, os métodos exatos apresentam algumas dificuldades que os tornam menos adequados para lidar com problemas complexos. Uma dessas dificuldades é o elevado consumo computacional, especialmente em situações em que o espaço de busca é extenso (BEHESHTI; SHAMSUDDIN, 2013). Para problemas mais intrincados e que pertencem à classe NP-Completo, como é o caso do UCTP, o uso exclusivo desses métodos seria praticável apenas para instâncias de pequena escala. A classificação da versão de otimização do problema como NP-Difícil implica em mais complexidade computacional, e isso inviabiliza a aplicação exclusiva de métodos exatos.

Alguns dos trabalhos que podem ser encontrados na literatura e que usaram 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 aproximados concentram-se em encontrar soluções que se aproximem da solução ideal, mas que possam ser calculadas de maneira mais eficiente. Isso envolve o uso de técnicas que equilibram a qualidade da solução com a velocidade de cálculo, tornando esse método mais adequado para resolver problemas com alta complexidade computacional.

Os métodos aproximados podem ser subdivididos em três categorias: métodos heurísticos, metaheurísticos e métodos especiais.

Os métodos heurísticos são estratégias aproximadas de resolução de problemas que buscam encontrar soluções aceitáveis em um tempo razoável,

embora não haja garantia de otimalidade. Geralmente, eles são desenvolvidos com base em intuição, experiência ou regras práticas específicas para orientar a busca por soluções. Exemplos comuns incluem Busca Local e Busca Tabu. Essas abordagens são frequentemente mais simples e são aplicadas a tipos específicos de problemas, aproveitando características particulares para encontrar soluções eficazes. No 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 abrangentes que coordenam a aplicação de métodos heurísticos em espaços de busca complexos e proporcionam soluções de boa qualidade e satisfatórias, superando abordagens manuais com menor esforço humano e em tempo computacional viável. No entanto, não garantem a otimalidade (JARDIM; SEMAAN; PENNA, 2015). Elas podem ainda ser classificadas com base em sua abordagem, podendo ser do tipo baseada em população ou do tipo solução única.

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

Apesar de as metaheurísticas populacionais possuírem a habilidade de explorar tanto global quanto localmente (BEHESHTI; SHAMSUDDIN, 2013), pesquisas sugerem que os algoritmos de busca local podem ter um desempenho mais eficiente em termos de tempo de execução, mesmo não garantindo a obtenção de soluções ótimas. Em contrapartida, os algoritmos populacionais se destacam pela capacidade de explorar novas regiões e se aproximar da solução ótima (HABASHI et al., 2018). Em resumo, esses métodos, por serem mais abstratos e adaptáveis,

mostram-se particularmente vantajosos para explorar de maneira eficaz espaços de busca vastos e complexos, como é o caso da programação de horários educacionais.

Os métodos especiais são abordagens personalizadas e adaptadas para resolver problemas específicos. Eles são desenvolvidos levando em consideração as características únicas de um problema particular, como restrições específicas, estruturas de dados particulares ou características distintivas. Esses métodos buscam otimizar a eficiência na busca por soluções, aproveitando aspectos específicos do problema. Os métodos especiais podem ser mais específicos e direcionados a uma classe específica de problemas, oferecendo soluções precisas para contextos particulares. No caso da programação de horários educacionais, isso pode incluir considerar preferências de professores, requisitos de salas de aula e requisitos de alunos, adaptando-se às peculiaridades do domínio.

Existem ainda os métodos híbridos e hiper-heurísticos. Os métodos híbridos, que combinam algoritmos de base populacional com técnicas de busca local, e hiper-heurísticas, operando em um nível mais abstrato ao combinar e adaptar várias heurísticas, têm se destacado 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). Essas abordagens avançadas visam superar as fragilidades individuais ao combinar estratégias, aproveitando as vantagens específicas de cada método. Os métodos híbridos buscam integrar a eficiência de algoritmos de base populacional com a precisão das técnicas de busca local, enquanto as hiper-heurísticas oferecem flexibilidade ao explorar o amplo espaço heurístico. Sua sofisticação e adaptabilidade tornam essas abordagens mais eficientes para lidar com as diversas restrições e preferências inerentes aos problemas de programação de horários educacionais, resultando em soluções mais eficazes e de alta qualidade.

A escolha da abordagem a ser adotada depende de diversos fatores, como a natureza do problema, o tamanho da instância, a disponibilidade de recursos computacionais e as preferências das partes envolvidas. Em muitos casos, uma combinação de métodos exatos e aproximados pode ser usada para obter um equilíbrio entre precisão e eficiência na resolução desse tipo de problema.

3. ABORDAGEM PROPOSTA

A solução para o CB-CTP é elaborada em duas etapas: construção e melhoria. Durante a etapa de construção, uma tabela de horários é criada iterativamente sem violar restrições rígidas, porém com a capacidade de violar restrições flexíveis. Já na etapa de melhoria, a solução inicial ou uma população de soluções serão aprimoradas gradualmente para que sejam violadas o mínimo possível de restrições flexíveis. Essas etapas são cruciais, pois influenciam a velocidade de convergência do algoritmo e a qualidade da solução final (WAHID; HUSSIN, 2016).

Nas seções a seguir, serão apresentados o Algoritmo Genético (GA), os detalhes da implementação, a construção da população inicial, as etapas de melhoria (avaliação da população, seleção, cruzamento, mutação e condição de parada) e o ajuste de parâmetros da metaheurística utilizada.

3.1. O Algoritmo Genético

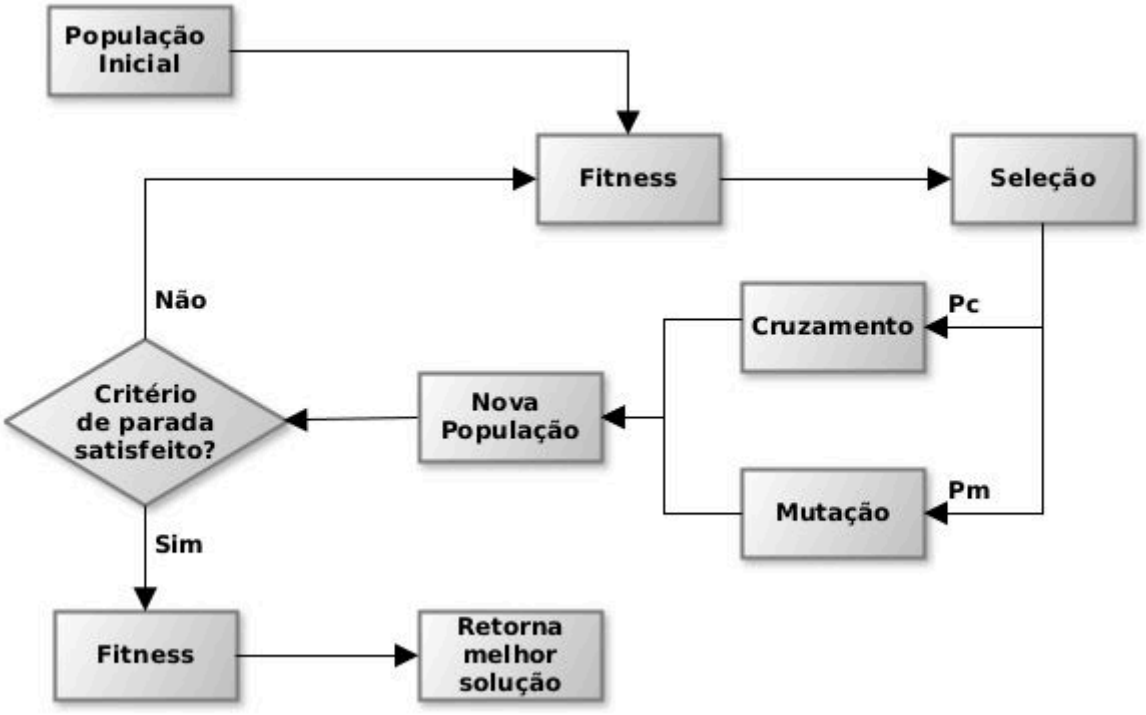
Os Algoritmos Genéticos são uma técnica de busca e otimização que se fundamenta no princípio darwiniano da seleção natural e da reprodução genética (GOLDBERG, 1989). Inventados por John Holland nos anos 1960 e refinados por seus alunos na Universidade de Michigan na década de 1970, esses algoritmos visavam explorar os fenômenos evolutivos de forma sistemática, espelhando os processos observados na natureza (AGUIAR, 1998). De acordo com a teoria de Charles Darwin, a evolução favorece os indivíduos mais adaptados ao ambiente, aumentando suas chances de sobrevivência e reprodução. Os Algoritmos Genéticos operam com estruturas de dados chamadas cromossomos, que representam possíveis soluções dentro do espaço de busca do problema. Esses cromossomos são submetidos a um processo que inclui avaliação, seleção, recombinação (ou cruzamento) e mutação (PACHECO, 1999).

Esses algoritmos têm sido aplicados em uma ampla gama de problemas de otimização (MICHALEWICZ, 1996), abrangendo desde funções matemáticas até questões combinatoriais e de planejamento, como o problema do caixeiro viajante e a otimização de rotas de veículos, além de problemas de layout de circuitos, distribuição, otimização de negócios e síntese de circuitos eletrônicos.

Segundo a biologia, a teoria da evolução postula que o ambiente seleciona os seres vivos mais aptos em cada geração de uma população, resultando na reprodução bem-sucedida apenas dos mais aptos. Durante o processo reprodutivo, ocorrem fenômenos como cruzamentos e mutações, que influenciam os genes armazenados nos cromossomos, promovendo a variabilidade dos organismos na população.

A seleção natural atua sobre essa população diversificada, permitindo a sobrevivência e a reprodução preferencial dos indivíduos mais adaptados ao ambiente. Os Algoritmos Genéticos são inspirados por esses princípios biológicos, o que justifica a adoção de muitos termos provenientes da biologia em sua descrição e aplicação. A Figura 4 ilustra o funcionamento de um AG padrão.

Figura 4 - Funcionamento de um Algoritmo Genético padrão



Fonte: Izidoro et al. (2014)

Um exemplo de Algoritmo Genético, em forma de pseudocódigo, é apresentado na Figura 5.

Figura 5 - Pseudocódigo do Algoritmo Genético

```
Algoritmo algoritmoGenetico
   // Inicialização da população de cromossomos
   população <- inicializarPopulação( )
   geração <- 1

   // Avaliação dos indivíduos na população (função objetivo e sobrevivência)
   avaliarPopulação(população)

   // Repetição do processo evolutivo
   Enquanto (geração <= máximoGerações OU nãoObjetivoFinal) Faça
      // Seleção de indivíduos para reprodução
      indivíduosSelecionados <- seleçãoReprodução(população)

      // Aplicação dos operadores de recombinação e/ou mutação
      novaPopulação <- aplicarOperadores(indivíduosSelecionados)

      // Avaliação dos indivíduos gerados na população
      avaliarPopulação(novaPopulação)

      // Seleção de indivíduos para sobreviver
      população <- seleçãoSobrevivência(população, novaPopulação)

      // Atualização da geração
      geração <- geração + 1
   FimEnquanto

FimAlgoritmo
```

Fonte: Autoria própria (2024)

Conforme pode ser visto no pseudocódigo, após a inicialização da população, em cada geração, o algoritmo genético selecionará os indivíduos mais aptos. Em seguida, serão aplicados os operadores de recombinação e mutação sobre esses indivíduos selecionados, gerando a prole que comporá a próxima geração. Esse processo será repetido iterativamente até que a condição de parada seja alcançada (RIBEIRO, 2012).

3.2. Inicialização da População

A etapa inicial da implementação do Algoritmo Genético envolve a criação da população inicial. Esta população é composta por uma variedade de agendamentos de horários, representando diferentes soluções para o problema. Durante o processo de construção desses horários, foi adotada uma heurística construtiva aleatória e foi

garantido que nenhuma restrição rígida fosse violada. Em caso de violação, as soluções eram descartadas imediatamente.

Figura 6 - Pseudocódigo da Geração da População Inicial

```
Função gerarPopulaçãoInicial(instancialTC, qtdePopulaçãoInicial)
   // Inicializa uma lista para armazenar a população inicial
   populaçãoInicial = [ ]

   // Repetição para criar cada indivíduo da população inicial
   Para cada i de 1 até qtdePopulaçãoInicial Faça
      // Preparação dos dados
      ordenaListaSalasAula(instanciaITC)
      criarEspacoTridimensional(instanciaITC.qtdeDiasDeAula,
instanciaITC.qtdeAulasPorDia, instanciaITC.qtdeSalasDeAula,
instanciaITC.turmas)

      // Alocação das disciplinas nos horários disponíveis
      agendamento = criarAgendamentoVazio( )
      Para cada disciplina em instanciaITC.disciplinas Faça
         alocarDisciplinaEmHorarioRandomico(instanciaITC, agendamento,
disciplina)
      FimPara

      // Adiciona o agendamento na população inicial
      populaçãoInicial.adicionar(agendamento)
   FimPara

   Retorne populaçãoInicial
FimFunção
```

Fonte: Autoria própria (2024)

Partindo de uma tabela de agendamento de horários completamente vazia, as aulas foram alocadas uma por uma. A heurística construtiva aleatória foi aplicada da seguinte forma:

- Uma estrutura tridimensional foi criada para representar a tabela de horários, abrangendo dias, horários e salas de aula;
- Inicia-se o processo de alocação das disciplinas dentro da agenda:
    - Itera-se sobre os horários (slots) disponíveis de forma aleatória;
    - É realizada uma busca local para identificar se já existe disciplina de uma mesma turma alocada na sala;

- Verifica-se se o horário (ou slot) está vazio e se atende às restrições fornecidas na instância;
- Se o horário (slot) for adequado, a disciplina é alocada nele;
- O horário alocado é removido da lista de horários disponíveis;
- Após a alocação de uma disciplina em um horário específico, a lista de horários indisponíveis é atualizada adicionando os horários que não estão mais disponíveis devido à alocação da disciplina;
- Este processo se repete até que todas as disciplinas tenham sido alocadas para os horários disponíveis.

Após a alocação de todas as disciplinas, o agendamento é salvo na população como uma de suas soluções, e uma nova estrutura tridimensional é criada. Esse ciclo se repete até que a quantidade desejada de soluções seja gerada para formar a população inicial.

Figura 7 - Representação do processo de agendamento considerando uma estrutura tridimensional

Sala A

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0					
aula 1					
aula 2				SecCosC	
aula 3			ArcTec		

Sala B

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0					
aula 1					
aula 2		ArcTec	ArcTec		
aula 3	GeoTec				GeoTec

Sala C

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0	TecCos		GeoTec	TecCos	GeoTec
aula 1		SecCosC		SecCosC	TecCos
aula 2	TecCos				
aula 3		TecCos		GeoTec	

Fonte: Autoria própria (2024)

Na Figura 7 é possível visualizar um exemplo de solução montada a partir de uma instância teste chamada Toy. Nesse exemplo, são exibidas 3 tabelas de agendamento, uma para cada sala de aula e cada uma contendo linhas e colunas,

que representam respectivamente os horários e os dias para alocação das aulas. As disciplinas da solução foram alocadas seguindo o passo a passo descrito na página anterior.

Para este trabalho, esse processo foi repetido no mínimo 50 vezes, resultando em uma população inicial composta por no mínimo 50 soluções distintas. A Figura 6 apresenta o pseudocódigo do processo de inicialização dessa população.

Esse processo resulta em uma variedade de soluções iniciais. Essas soluções, designadas como "pais", serão submetidas à etapa de avaliação, que será detalhada no próximo tópico.

3.3. Avaliação da População

Após a inicialização da população, cada solução construída é avaliada recebendo um valor de aptidão. Este valor é determinado pela função objetivo, que é a soma das penalidades atribuídas às restrições flexíveis violadas. A avaliação permite entender quão boa é cada solução em relação ao objetivo do problema. No nosso caso, as soluções com menor pontuação são consideradas melhores, indicando menos violações de restrições. Em outras palavras, quanto menor o valor calculado da função objetivo, maior será a aptidão da solução.

Antes de calcular o valor de aptidão usando a função objetivo, é necessário verificar se cada solução gerada dentro da população viola as restrições flexíveis. Se uma violação for identificada, penalidades são contabilizadas e aplicadas, conforme explicado na subseção 2.2.3 deste trabalho.

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

- **'restricaoCapacidadeSala':** Esta função verifica a restrição flexível "Capacidade da sala". Para isso, é verificado se a quantidade de alunos de cada disciplina excede a capacidade da sala onde ela foi alocada. Se for constatada a violação de capacidade, a penalidade é calculada multiplicando a diferença entre a quantidade de alunos e a capacidade da sala pelo fator de penalidade (neste caso, 1). Todas as penalidades são acumuladas e retornadas como o total de penalidades;

- '**restricaoDiasMinimosPorSemana':** Esta função verifica a restrição flexível "Dias mínimos de trabalho". Para isso, verifica-se em quantos dias diferentes da semana cada disciplina está alocada. Se esse número for menor que o mínimo exigido, uma penalidade é aplicada. Essa penalidade é calculada multiplicando a diferença entre o número mínimo de dias e o número de dias em que a disciplina foi alocada pelo fator de penalidade (neste caso, 5). Todas as penalidades são acumuladas e retornadas como o total de penalidades;

- **'restricaoOciosidadeTurma':** Esta função verifica a restrição flexível "Ociosidade da turma". Para cada turma na lista de turmas fornecida, ela identifica os dias diferentes em que as disciplinas estão alocadas e verifica se há intervalos entre as aulas das disciplinas no mesmo dia. Se forem encontrados intervalos, uma penalidade igual a 2 é aplicada para cada intervalo. Todas as penalidades são acumuladas e retornadas como o total de penalidades;

- **'restricaoEstabilidadeSala':** Esta função verifica a restrição flexível "Estabilidade da sala". Para cada disciplina na lista de disciplinas ofertadas, ela identifica todas as alocações correspondentes a essa disciplina. Se houver mais de uma sala sendo utilizada, a função conta o número de salas extras e adiciona esse valor como penalidade. O total de penalidades, que representa o número de salas extras onde uma disciplina é alocada, é então retornado.

Após calcular o valor de todas as violações de restrições usando essas 4 funções, podemos calcular o valor de aptidão na função objetivo. Para isso, foi implementada a função 'iniciarFuncaoObjetivo', que recebe uma lista de soluções (população inicial), uma lista de disciplinas ofertadas e uma lista de turmas. Essa função retorna a soma das penalidades de todas as restrições em cada solução. Esses dados são cruciais para a próxima etapa, que é a etapa de seleção.

3.4. Seleção

A escolha dos indivíduos para reprodução é determinada pelos operadores genéticos de seleção. Estes operadores têm impacto na eficácia da resposta, na

rapidez da convergência e na eficiência global do algoritmo. Alguns dos operadores de seleção mais utilizados são:

- **Torneio:** Os cromossomos são selecionados aleatoriamente na população para participar de um torneio play-off onde os vencedores são selecionados para sobreviver. Ou seja, o cromossomo com maior valor de aptidão será escolhido para compor o grupo que irá se reproduzir (GOMIDE, L. R.et al., 2009; RIBEIRO, 2012);
- **Classificação simples ou Elitismo:** Os cromossomos são ordenados e escolhidos de acordo com sua classificação. Essa escolha pode ser feita por meio de um número X de vagas ou por percentual. O objetivo é selecionar apenas os melhores indivíduos de acordo com seus valores de aptidão (SAKAWA, 2002);
- **Roleta:** Cada cromossomo recebe uma área em uma roleta que é proporcional ao seu valor de aptidão. A seleção dos cromossomos é feita de acordo com a probabilidade de serem selecionados. Quanto maior o valor de aptidão, maior a probabilidade de serem selecionados, a cada nova rodada da roleta (GOLDBERG, 1989).
- **Bi-classista:** Esse método considera a seleção proporcional ao valor ordenado do valor de aptidão da população, sendo selecionada uma porção para as melhores aptidões e outra para os piores, até completar a lista de seleção (GOMIDE, L. R. et al., 2009).

Neste trabalho, foi adotado o Elitismo como método de seleção. Para isso, desenvolvemos a função 'selecaoElite', que recebe a lista de soluções geradas durante a inicialização da população, junto com seus respectivos valores de aptidão. Essa função ordena as soluções a partir dos seus valores de aptidão e retorna a porcentagem de soluções com maior qualidade para que estas sejam empregadas na etapa seguinte (cruzamento).

Dado que temos uma população inicial composta por 50 soluções, optamos por selecionar para a etapa de cruzamento apenas 40% das melhores soluções (o equivalente a 20 soluções).

3.5. Cruzamento

O cruzamento desempenha um papel crucial na promoção da diversidade genética na população de soluções. Essa diversidade é fundamental para evitar a convergência prematura para ótimos locais e para garantir a descoberta de soluções de alta qualidade. Durante esse processo, os genes de dois ou mais indivíduos selecionados são combinados, simulando a reprodução sexual na natureza e resultando na criação de novas soluções, denominadas "filhos", nos quais o código genético dos cromossomos é misturado.

No contexto específico do Problema de Agendamento de Horários, em que o objetivo é otimizar a alocação de disciplinas em horários específicos, a escolha adequada do método de cruzamento é crucial para o sucesso do algoritmo genético.

Existem várias técnicas de cruzamento disponíveis, cada uma com sua abordagem específica para trocar genes entre os pais e criar descendentes. O cruzamento de ponto único (1PX) seleciona aleatoriamente um ponto de cruzamento ao longo dos cromossomos dos pais e troca os genes antes desse ponto. Por sua vez, o cruzamento de dois pontos (2PX) é semelhante ao 1PX, mas escolhe dois pontos de cruzamento, trocando os genes entre esses pontos. Já o cruzamento de múltiplos pontos (MPX) seleciona mais de dois pontos de cruzamento, trocando os segmentos entre esses pontos. Por fim, o cruzamento segmentado (SX) divide os cromossomos em segmentos e os troca alternadamente entre os pais para gerar os descendentes. Cada técnica tem suas vantagens e é escolhida com base nas características do problema e do algoritmo genético em uso.

No caso específico do Problema de Agendamento de Horários, é comum a utilização do cruzamento de ponto único, de dois pontos ou uma combinação desses dois métodos. No entanto, após tentativas mal sucedidas de implementar esses métodos de forma isolada e combinada, devido à violação da restrição rígida que exige a alocação das aulas de todas as disciplinas, optou-se pela utilização do cruzamento segmentado. Essa abordagem mostrou-se eficaz para lidar com a natureza dos dados do problema em questão, gerando soluções que respeitam a restrição mencionada anteriormente.

No código implementado, as soluções são representadas como listas de listas de objetos, onde cada objeto contém informações sobre uma disciplina, incluindo seu nome e horário. Na função 'gerarAlelos', os genes, correspondentes aos

horários das disciplinas, são trocados entre os pais para gerar os filhos. Esta função divide as disciplinas dos cromossomos em segmentos e os combina alternadamente entre os pais para criar os descendentes, garantindo uma manipulação precisa dos horários das disciplinas e o respeito às restrições rígidas do problema.

Figura 8 - Representação em tabelas do cruzamento segmentado implementado neste trabalho

Pai 1

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0	GeoTec - sala B	SecCosC - sala C	ArcTec - sala C	ArcTec - sala C	
aula 1	GeoTec - sala A	SecCosC - sala A	GeoTec - sala C	TecCos - sala B	
aula 2	TecCos - sala B	SecCosC - sala A	GeoTec - sala B	ArcTec - sala B	TecCos - sala B
aula 3	GeoTec - sala C	TecCos - sala B			TecCos - sala A

Pai 2

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0	TecCos - sala B	GeoTec - sala C	GeoTec - sala A		
aula 1	SecCosC - sala A	SecCosC - sala A	GeoTec - sala B	GeoTec - sala A	
aula 2	GeoTec - sala C	TecCos - sala A	SecCosC - sala B	ArcTec - sala B	TecCos - sala C
aula 3		ArcTec - sala A	TecCos - sala B	ArcTec - sala C	TecCos - sala A

Filho 1

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0	GeoTec - sala B				
aula 1	GeoTec - sala A / SceCosC - sala A	SecCosC - sala A	GeoTec - sala C	TecCos - sala B	
aula 2	TecCos - sala B		GeoTec - sala B/ SceCosC - sala B	ArcTec - sala B	TecCos - sala B
aula 3	GeoTec - sala C	TecCos - sala B / ArcTec - sala A		ArcTec - sala C	TecCos - sala A

Filho 2

	dia 00	dia 01	dia 02	dia 03	dia 04
aula 0	TecCos - sala B	GeoTec - sala C / SceCosC - sala C	GeoTec - sala A / ArcTec - sala C	ArcTec - sala C	
aula 1		SecCosC - sala A	GeoTec - sala B	GeoTec - sala A	
aula 2	GeoTec - sala C	TecCos - sala A / SceCosC - sala A		ArcTec - sala B	TecCos - sala C
aula 3			TecCos - sala B		TecCos - sala A

Fonte: Autoria própria (2024)

A Figura 8 mostra como esse método foi aplicado para cruzar os dados de 2 cromossomos gerados aleatoriamente. No Filho 1 é possível observar que constam disciplinas do Pai 1 (slots na cor verde) e disciplinas do Pai 2 (slots na cor azul). O mesmo pode ser observado no Filho 2, que também foi formado a partir de disciplinas do Pai 1 e Pai 2. Alguns dos cromossomos filhos gerados pelo cruzamento podem violar a restrição rígida de ocupação da sala, conforme pode ser

visto nos slots de cor laranja. No entanto, essa violação pode ser facilmente corrigida na etapa de mutação.

Neste trabalho específico, é importante destacar que o cruzamento foi aplicado a todas as soluções que retornaram da etapa de seleção. Essa abordagem foi adotada para promover o máximo de variabilidade nas soluções, contribuindo para uma exploração mais ampla do espaço de busca e aumentando as chances de encontrar soluções de alta qualidade.

Ao aplicar o cruzamento a todas as soluções que retornaram da etapa de seleção, foi possível garantir que todas elas contribuíssem para a diversidade genética da população e, assim, para a geração de soluções melhores no decorrer do processo de otimização.

3.6. Mutação

A mutação desempenha um papel crucial nos Algoritmos Genéticos (AG), permitindo a introdução de variação aleatória nos genes dos indivíduos. Enquanto o cruzamento combina informações de dois pais específicos, a mutação realiza pequenas alterações aleatórias nas soluções resultantes. Essas modificações são essenciais para explorar novas áreas no espaço de busca e evitar a convergência prematura para um ótimo local, ampliando assim a diversidade na busca do AG.

A operação de mutação ocorre após o processo de cruzamento, concedendo a cada indivíduo resultante uma chance pré-definida de sofrer alterações em seus genes. A fim de evitar uma exploração indiscriminada do espaço de soluções, essa probabilidade é geralmente mantida em um valor baixo. Entre os operadores de mutação mais comuns estão (GOLDBERG, 1989):

- **Mutação creep:** onde um valor aleatório é adicionado ou subtraído ao valor do gene;
- **Mutação aleatória:** onde cada gene selecionado para mutação recebe um valor aleatório dentro do conjunto válido de valores;
- **Mutação por troca:** onde pares de genes são selecionados e seus valores são trocados entre si, introduzindo uma mistura adicional de informações genéticas.

Neste trabalho, foi adotada a mutação por troca (swap), tendo-se em vista que sua implementação visa apenas a correção das violações de restrições rígidas encontradas nas soluções geradas após a etapa de cruzamento (conforme ilustrado na Figura 8).

3.7. Critério de sobrevivência

Após a criação da nova população através do cruzamento e da mutação, é necessário decidir quais indivíduos irão compor a próxima geração. Isso pode ser feito por meio de várias estratégias, como a substituição completa da população anterior pela nova, a combinação de ambas ou mantendo apenas os melhores indivíduos.

As regras mais comuns, segundo Goldberg (1989), são: (i) os pais sempre substituem os filhos; (ii) os filhos substituem os pais somente se a média do valor de aptidão dos filhos for maior que a média de aptidão dos pais. A maioria dos algoritmos, de fato, adota o método de sempre substituir os pais pelos filhos, pois isso ajuda a manter a diversidade dos indivíduos gerados.

Neste trabalho, optou-se pela combinação da população anterior com a nova, no entanto, isso ocorrerá da seguinte forma:

- A população gerada após os operadores de cruzamento e mutação será somada a população elite que passou por essas etapas anteriormente. Ou seja, os pais elite serão somados aos filhos e irão compor a nova população;
- Essa nova população passará novamente pelo processo de avaliação, seleção, cruzamento e melhoria, gerando uma nova população;
- Esse processo se repetirá 5 vezes (critério de parada).

3.8. Condição de parada

A execução do algoritmo genético é finalizada quando uma condição de parada específica é alcançada. Isso pode ocorrer após um número fixo de gerações, quando uma solução satisfatória é encontrada ou quando não há mais melhorias nas soluções ao longo de várias gerações.

Neste trabalho, será adotada uma abordagem diferente das mencionadas anteriormente. Dado que estão sendo utilizadas 21 instâncias diferentes e que não se conhece o melhor valor de aptidão dessas instâncias, optou-se por considerar como critério de parada a aplicação da etapa de melhoria (seleção, cruzamento e mutação) por 5 vezes.

A Tabela 3 abaixo exibe o cronograma de horários, resultante dos dados fornecidos pela melhor solução gerada ao executar a instância de teste Toy. Para essa instância, foi obtido 2 como valor de aptidão.

Tabela 3 - Representação da melhor solução gerada para a instância Toy

Currículo 1					
	dia 00	**dia 01**	**dia 02**	**dia 03**	**dia 04**
aula 0	TecCos		SecCosC		
aula 1	ArcTec	SecCosC	ArcTec	TecCos	TecCos
aula 2			TecCos	SecCosC	TecCos
aula 3				ArcTec	

Currículo 2					
	dia 00	**dia 01**	**dia 02**	**dia 03**	**dia 04**
aula 0	TecCos			GeoTec	GeoTec
aula 1	GeoTec			TecCos	TecCos
aula 2			TecCos	GeoTec	TecCos
aula 3			GeoTec		

Fonte: Autoria própria (2024)

4. RESULTADOS

Os experimentos computacionais foram conduzidos em um sistema equipado com processador 2,6 GHz Intel Core i7 6-Core, memória de 16GB 2667 MHz DDR4, gráficos AMD Radeon Pro 5300M 4 GB, Intel UHD Graphics 630 1536 MB e sistema operacional macOS Sonoma, versão 14.2.1. A implementação dos algoritmos foi realizada na linguagem Kotlin, seguindo o paradigma orientado a objetos. Para este estudo, foram utilizadas as 21 instâncias do problema de horários de curso baseadas no currículo disponibilizadas pelo ITC-2007.

4.1. Execução de Experimentos

Para avaliar o desempenho do algoritmo proposto neste trabalho foram conduzidos diferentes testes. A Tabela 4 apresenta os resultados obtidos pelo algoritmo de construção para as 21 instâncias do ITC-2007. Tais instâncias foram classificadas em três níveis de dificuldade: iniciais, atrasadas e ocultas. Para avaliar o desempenho do algoritmo, foram conduzidos 50 testes de execução com cada instância, considerando uma população inicial de 50 soluções em cada teste. Sendo assim, o número máximo de soluções viáveis que foram geradas para cada instância foi de 2.500.

Os dados apresentados na tabela são:

- **Melhor F(O):** melhor valor da função objetivo obtido com o algoritmo de construção;
- **Pior F(O):** pior valor da função objetivo;
- **Tempo Médio:** tempo médio em segundos para obter uma população de soluções viáveis;
- **Máx. Soluções:** número máximo de soluções viáveis geradas;
- **Média Soluções:** número médio de soluções geradas por população.

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

Tabela 4 - Resultados da Etapa de Construção para as Instâncias do ITC-2007

Níveis	Instâncias	Melhor F(O)	Pior F(O)	Tempo Médio	Máximo Soluções	Média Soluções
Iniciais	Comp01	1758	3013	90 ms	2500	50
	Comp02	6527	9615	933 ms	2500	50
	Comp03	4325	7420	529 ms	2500	50
	Comp04	4200	6115	755 ms	2500	50
	Comp05	6757	9893	594 ms	2500	50
	Comp06	5525	7777	1823 ms	2500	50
	Comp07	5181	7276	2979 ms	2500	50
Atrasadas	Comp08	3724	5662	1068 ms	2500	50
	Comp09	3724	5870	865 ms	2500	50
	Comp10	4609	6609	2015 ms	2500	50
	Comp11	1622	2652	95 ms	2500	50
	Comp12	2844	4160	1541 ms	2500	50
	Comp13	5246	7452	928 ms	2500	50
	Comp14	3098	4935	877 ms	2500	50
Ocultas	Comp15	4323	7438	526 ms	2500	50
	Comp16	4532	6453	1846 ms	2500	50
	Comp17	4689	7123	1409 ms	2500	50
	Comp18	1110	2066	285 ms	2500	50
	Comp19	4046	6674	738 ms	2500	50
	Comp20	6830	9336	2447 ms	2500	50
	Comp21	4440	7009	1259 ms	2500	50

Fonte: Autoria própria (2024)

A Tabela 4 revela que o tempo médio para gerar uma população com no máximo 50 soluções viáveis para cada instância é inferior a 3 segundos (3000 milissegundos). Isso indica que o algoritmo é viável computacionalmente na construção da população inicial. Apenas as instâncias Comp07, Comp10 e Comp20 consomem em média mais de 2 segundos (2000 milissegundos) para gerar a população completa. Essa discrepância pode ser atribuída à complexidade específica dessas instâncias, que possuem características que exigem um tempo de processamento maior. Em geral, o número médio de soluções geradas foi suficiente para alcançar os objetivos do estudo. A quantidade de soluções permitiu a avaliação da efetividade do algoritmo e a comparação com outros trabalhos da área.

A heurística construtiva se mostrou eficaz na geração de soluções iniciais

viáveis para o CB-CTP, dentro de um tempo aceitável e sem violar restrições rígidas. No entanto, a violação de restrições flexíveis indica que a heurística pode ser aprimorada em pesquisas futuras.

4.2. Comparação e Análise dos Resultados

A Tabela 5 traz a comparação entre as soluções iniciais geradas pelos algoritmos propostos por Teoh, Abdullah e Haron (2015), Segatto (2017), Wahid et al. (2019), Sena (2021) e a proposta deste trabalho. Os valores em azul indicam as soluções com menos penalidades para as restrições flexíveis. Já os valores em vermelho, indicam as soluções com mais penalidades.

Tabela 5 - Comparação de resultados da Etapa de Construção

Instâncias	Solução Inicial				
	Teoh, Abdullah e Haron (2015)	Segatto (2017)	Wahid et al. (2019)	Sena (2021)	Solução Proposta
Comp01	412	623	330	227	1758
Comp02	1678	1075	769	537	6527
Comp03	2146	849	702	478	4325
Comp04	1678	1273	694	479	4200
Comp05	2624	1189	1466	942	6757
Comp06	1493	2342	947	638	5525
Comp07	1482	2615	1043	680	5181
Comp08	1582	1522	790	528	3724
Comp09	1294	1021	847	586	3724
Comp10	1895	1891	898	586	4609
Comp11	417	396	230	53	1622
Comp12	1361	1496	1498	1229	2844
Comp13	1964	1787	793	500	5246
Comp14	809	1417	745	471	3098
Comp15	1929	849	702	461	4323
Comp16	1167	1697	949	606	4532
Comp17	1299	1836	902	646	4689
Comp18	777	600	583	417	1110
Comp19	1903	839	637	493	4046
Comp20	1313	3353	1042	697	6830
Comp21	2288	1329	928	640	4440

Fonte: Autoria própria (2024)

A comparação das soluções iniciais foi feita levando em conta apenas o valor obtido a partir da função objetivo para cada instância. Vale salientar que essa comparação tem como objetivo verificar se a abordagem proposta pode gerar soluções iniciais viáveis para o problema em questão, porém, observou-se que embora o tempo de execução fosse baixo, ainda ocorria uma quantidade muito alta de violações de restrições flexíveis, tornando necessário seguir para a etapa de melhoria e recalcular o valor de aptidão.

Tabela 6 - Análise de desempenho em diferentes populações na etapa de construção

Instâncias	Solução Inicial			
	População com 100 soluções		População com 1.000 soluções	
	Melhor F(O)	Tempo Médio	Melhor F(O)	Tempo Médio
Comp01	1789	231 ms	1819	1816 ms
Comp02	6735	1927 ms	6474	18497 ms
Comp03	4311	1111 ms	4543	10607 ms
Comp04	4334	1519 ms	4039	14802 ms
Comp05	7169	1229 ms	6705	11603 ms
Comp06	5545	3701 ms	5445	36178 ms
Comp07	5440	5966 ms	5262	59310 ms
Comp08	3756	2199 ms	3759	21479 ms
Comp09	3997	1773 ms	3588	17315 ms
Comp10	4680	4031 ms	4526	39839 ms
Comp11	1662	238 ms	1604	1863 ms
Comp12	3044	3174 ms	2618	30780 ms
Comp13	5292	1935 ms	5099	18699 ms
Comp14	3393	1838 ms	3234	17461 ms
Comp15	4750	1117 ms	4486	10586 ms
Comp16	4637	3789 ms	4436	36892 ms
Comp17	4759	2866 ms	4620	28052 ms
Comp18	1229	646 ms	1179	5746 ms
Comp19	4258	1526 ms	4000	14730 ms
Comp20	7069	4992 ms	6748	48883 ms
Comp21	4788	2600 ms	4273	25568 ms

Fonte: Autoria própria (2024)

A fim de explorar o potencial de geração de soluções iniciais de forma

aleatória, foram realizados ainda testes exploratórios diminuindo a quantidade de repetições e aumentando a quantidade de soluções geradas em cada instância. A Tabela 6 traz os resultados obtidos ao executar cada instância 5 vezes, porém com populações variando em 100 e 1000 soluções geradas na população. Os melhores resultados para cada instância estão destacados em azul.

Ao comparar os resultados gerados pelas populações com maior número de soluções iniciais (Tabela 4) com os resultados gerados pelas populações com maior número de soluções (Tabela 6), é possível concluir que ao aumentar o tamanho da população inicial o valor de aptidão diminui. No entanto, o tempo para processamento também aumenta.

Tabela 7 - Comparação de resultados da Etapa de Melhoria

Instâncias	**Solução Otimizada (Média)**				
	Abdullah (2012)	**Segatto (2017)**	**Kiefer (2017)**	**Sena (2021)**	**Solução Proposta**
Comp01	5	5	5	5	610
Comp02	53.9	55.1	41.5	81.9	1849
Comp03	84.2	84.95	71.7	94.5	1326
Comp04	51.9	40.8	35.1	43.1	1066
Comp05	339.5	328.45	305.2	450.6	3240
Comp06	64.4	58.35	47.8	68.7	1565
Comp07	20.2	28.7	14.5	36.6	2876
Comp08	47.9	45.45	41	48.8	1210
Comp09	113.9	107.9	102.8	121.2	1100
Comp10	24.1	25.05	14.3	38.8	1724
Comp11	0	0	0	0	598
Comp12	355.9	353.3	319.4	446.6	995
Comp13	72.4	76.75	60.7	83	1360
Comp14	63.3	61	54.1	67.8	2647
Comp15	88	84.95	72.1	98.6	2221
Comp16	51.7	44.4	33.8	46.6	1560
Comp17	86.2	85	75.7	100.7	1411
Comp18	85.8	84.85	66.9	93.3	793
Comp19	78.1	69.7	62.6	86.8	1729
Comp20	42.9	45.95	27.2	59.3	1691
Comp21	121.5	108.95	97	133.8	1951

Fonte: Autoria própria (2024)

Após a implementação da etapa de melhoria, o desempenho do algoritmo foi reavaliado através de 50 novas execuções independentes para cada instância do problema. Novamente, uma população inicial de 50 soluções foi utilizada em cada execução. A Tabela 7 apresenta os resultados obtidos. Para cada instância, é apresentado o valor da solução com melhor aptidão após a aplicação do processo de melhoria. Além disso, a tabela inclui resultados de outros estudos encontrados na literatura, permitindo a comparação do desempenho do algoritmo aprimorado com outras abordagens.

Os melhores resultados ou as melhores médias para cada instância estão destacadas em azul. Já os piores resultados foram indicados na cor vermelha. Mas de modo geral, o algoritmo proposto apresentou uma piora significativa quando comparado com todos os outros trabalhos.

Mais uma vez visando explorar o potencial de geração de soluções iniciais de forma aleatória, deu-se continuidade aos testes exploratórios diminuindo a quantidade de repetições e aumentando a quantidade de soluções geradas em cada instância. A Tabela 8 traz os resultados obtidos ao executar cada instância 5 vezes, porém com populações variando em 100 e 1000 soluções geradas na população. Os melhores resultados para cada instância estão destacados em azul.

Ao comparar os resultados gerados pelas populações com maior número de soluções iniciais (Tabela 8) com os resultados gerados pelas populações com maior número de soluções (Tabela 7), é possível concluir que ao aumentar o tamanho da população inicial, o número de aptidão diminui.

Em geral, os resultados da pesquisa indicam que a abordagem proposta para a geração de cronogramas de horários no Problema de Horários de Cursos Baseado em Currículo (CB-CTP) não apresenta a efetividade desejada. Os experimentos realizados demonstraram que o algoritmo genético clássico, em sua implementação específica para este trabalho, não alcança o mesmo nível de performance que outros métodos de resolução para o CB-CTP. Diante disso, torna-se inviável a aplicação da abordagem proposta em instâncias reais do problema, como as geradas a partir de dados do Departamento de Informática da UERN.

Tabela 8 - Análise de desempenho em diferentes populações na etapa de melhoria

Instâncias	Solução Otimizada			
	População com 100 soluções		População com 1.000 soluções	
	Melhor F(O)	Tempo Final	Melhor F(O)	Tempo Final
Comp01	908	9.2871 s	461	1m 5.34 s
Comp02	1225	34.8931 s	5518	8m 14.56 s
Comp03	4209	35.1832 s	3986	5m 39.32 s
Comp04	1075	46.0513 s	1130	8m 26.51 s
Comp05	6453	14.7799 s	6309	2m 24.62 s
Comp06	3086	1m 51.4050 s	3406	19m 15.37s
Comp07	3369	3m 7.8594 s	1846	25m 56.64 s
Comp08	1210	1m 17.5959 s	3444	14m 40.64 s
Comp09	1277	34.0058 s	2090	10m 44.80 s
Comp10	2982	2m 11.1035 s	4345	22m 10.22 s
Comp11	490	5.8845 s	1267	1m 30.92 s
Comp12	1090	33.7410 s	1687	7m 48.65 s
Comp13	3010	55.4988 s	1150	8m 51.41 s
Comp14	2809	56.9860 s	1624	7m 4.96 s
Comp15	1153	33.7264 s	1080	4m 31.22 s
Comp16	4405	2m 17.2505 s	4281	25m 1.44 s
Comp17	2826	1m 38.5468 s	1416	11m 1.89 s
Comp18	832	11.4973 s	682	1m 33.39 s
Comp19	3868	45.9811 s	3688	7m 37.68 s
Comp20	4156	2m 46.1866 s	6462	25m 33.88 s
Comp21	1330	1m 0.7440 s	3977	14m 47.96 s

Fonte: Autoria própria (2024)

5. CONCLUSÃO

Este trabalho apresentou uma implementação do Algoritmo Genético Clássico para o Problema de Horários de Cursos baseado em Currículo. A eficácia do algoritmo foi avaliada após a sua aplicação em um conjunto de 21 instâncias fornecidas pelo ITC-2007. Os resultados obtidos provaram que, apesar de ser computacionalmente viável, o algoritmo precisa de ajustes para conseguir fornecer soluções de qualidade. Isso já pôde ser observado na etapa de construção, que por ter sido feita de forma aleatória, não gerou uma população inicial boa o suficiente para evoluir e apresentou alto número de violações de restrições flexíveis.

Já na etapa de melhoria foram observados resultados satisfatórios, dado que quando foram aplicadas as operações de cruzamento e mutação, as soluções geradas apresentaram uma queda significativa no número de violações de restrições flexíveis e consequentemente nos valores de aptidão destas soluções.

Por fim, pode-se concluir que este trabalho trouxe resultados relevantes para o seu objetivo inicial, dado que já estão sendo gerados horários de aula que podem ser utilizados. Além disso, o algoritmo proposto consegue gerar soluções viáveis quando aplicado a instâncias pequenas (com poucas disciplinas para alocar ou com poucas restrições).

As perspectivas para pesquisas futuras incluem:

- Refatorar o código para corrigir falhas e aprimorar a eficiência;
- Implementar uma construção gulosa aleatória da população inicial para gerar soluções de melhor qualidade;
- 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;
- Ajustar parâmetros na etapa de melhoria para otimizar a geração de soluções;
- Tornar o algoritmo escalável para aplicação em instâncias maiores, visando sua utilização na UERN como um todo.

REFERÊNCIAS

AGUIAR, M. S. Análise Formal da Complexidade de Algoritmos Genéticos. 75 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática da Universidade Federal do Rio Grande do Sul, Universidade Federal do Rio Grande do Sul, Porto Alegre. 1998. Citado na página 36.

ALIXANDRE, B. F. d. F.; DORN, M. D-brkga: A distributed biased random-key genetic algorithm. In: 2017 IEEE Congress on Evolutionary Computation (CEC). [S.l.: s.n.], 2017. p. 1398–1405. ISSN null. Citado na página 12.

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 33.

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 33 e 34.

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 33.

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 33.

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.

GOLDBERG, D. E. Genetic Algorithms in Search, Optimization and Machine Learning,1. ed. Boston: Addison-Wesley, 1989. 432 p. Citado nas páginas 36, 43, 46 e 47.

GOMIDE, L. R.; ARCE, J. E.; DA SILVA, A. C. L. Uso do Algoritmo Genético no Planejamento Florestal considerando seus Operadores de Seleção. Cerne, v. 15, n.4, p. 460 – 467, 2009. Citado na página 43.

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 34 e 35.

IZIDORO, S. C.; DE MELO-MINARDI, R. C. e PAPPA, G. L. (2014). GASS: identifying enzyme active sites with genetic algorithms. Bioinformatics, 31(6):864–870. Citado na página 37.

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 34.

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 35.

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 33.

MICHALEWICZ, Z. Genetic Algorithms + Data Structures = Evolution Programs. 3. ed. Berlin: Springer-Verlag, 1996. 387 p. Citado na página 36.

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 33.

PACHECO, M. A. Algoritmos Genéticos: Princípios e Aplicações. Rio de Janeiro, Universidade Pontifícia do Rio de Janeiro, 1999. Apostila. Citado na página 36.

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 33.

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 nas páginas 21 e 32.

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 nas páginas 12 e 33.

RIBEIRO, M. D. R. Um Estudo sobre Algoritmos Genéticos e Culturais e Métodos de Descoberta de Conhecimento, Perfis e Reputação em Redes Sociais. 79 f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas. 2012. Citado nas páginas 38 e 43.

SAKAWA, M. Genetic Algorithms and Fuzzy Multiobjective Optimization. Netherlands: Springer, 2002. 306 p. Citado na página 43.

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 31.

SEGATTO, E. d. A. Um estudo de estruturas de vizinhanças no GRASP aplicado ao Problema de Tabela-Horário para Universidades. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - ES, 2017. Citados nas páginas 51 e 53.

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 nas páginas 30, 51 e 53.

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 33.

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 33.

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 nas páginas 12 e 35.

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 nas páginas 12, 23 e 51.

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 34.

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 51.