
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DANIELA COSTA DE SENA

# UMA ABORDAGEM HÍBRIDA PARA O PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE CURSOS UNIVERSITÁRIOS

MOSSORÓ - RN

2021

**DANIELA COSTA DE SENA**

# UMA ABORDAGEM HÍBRIDA PARA O PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE CURSOS UNIVERSITÁRIOS

Dissertação apresentada ao 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, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Dr. Carlos Heitor Pereira Liberalino
Co-orientador: Dr. Francisco Chagas Lima Júnior

**MOSSORÓ - RN**

**2021**

**DANIELA COSTA DE SENA**

# UMA ABORDAGEM HÍBRIDA PARA O PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE CURSOS UNIVERSITÁRIOS

Dissertação apresentada ao 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, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

APROVADA EM:____ / ____ / _____.

BANCA EXAMINADORA

---

Dr. Carlos Heitor Pereira Liberalino
**Orientador**

---

Dr. Francisco Chagas de Lima Júnior
**Co-orientador**

---

Dr. Fábio Francisco da Costa Fontes
**Convidado - Membro Interno**

*Aos meus pais José Vicente e Josefa,*
*por todo amor, cuidado e compreensão.*

# Agradecimentos

Em primeiro lugar, sou grata a Deus, pois ele é a fortaleza do meu coração, a minha porção e nele deposito toda a minha esperança.

Agradeço aos meus pais, Josefa e José Vicente, por todo o apoio, compreensão, pelas despesas pagas quando eu não tinha a possibilidade de pagar e por cada moeda desembolsada para que eu pudesse estar alcançando os meus objetivos pessoais e profissionais.

Agradeço ao meus orientadores, Prof. Dr. Carlos Heitor Pereira Liberalino e Prof. Dr. Francisco Chagas de Lima Júnior, por todos os ensinamentos, pela paciência e pela disponibilidade em orientar naquilo que era necessário.

Agradeço também ao Prof. Dr. Fábio Francisco da Costa Fontes, pelos ensinamentos, pela disposição constante no esclarecimento de dúvidas e sugestões para o trabalho de dissertação, pela confiança depositada na minha pessoa e por todo o acompanhamento durante o período de estágio. O ensinamentos do Prof. Fábio foram primordiais para a conclusão do mestrado.

Agradeço a todos os professores e coordenadores do PPgCC pela dedicação ao programa, pela paciência e disposição em ensinar os alunos. Obrigado por resolverem todas as questões burocráticas e por sempre estarem em busca de novas ferramentas tecnológicas e apoio financeiro.

E claro, mais não menos importante, sou grata a Deus por cada um dos amigos que conheci durante o período do mestrado. Entre eles, destaco: Virgínia, Rayana, Lohanna, Israel, Elizeu, Eric, Tailânio, Alan, Otília, Johnattan, Naylson, Júlio, Jesaías, Leonardo, Cynthia, Thaíza e Thiago. Os momentos de diálogo, brincadeiras e risos no corredor do LCC da UFERSA foram essenciais para aliviar o estresse causado pelas atividades do mestrado. O que resta agora é a saudade. Saudades das pessoas e das aulas. No entanto, o coração está grato por cada momento vivido.

À CAPES pelo apoio financeiro para realização deste trabalho de pesquisa. O apoio financeiro foi de fundamental importância para a minha permanência no programa e, por isso, sou imensamente grata.

A todos que contribuíram diretamente ou indiretamente, Muito Obrigado.

”QEQ CQ QR QER RER”
*Unknown*

# Resumo

O Problema de Horários de Cursos Universitários é um problema clássico de otimização combinatória estudado por diversos pesquisadores. Na Segunda Competição Internacional de Horários (ITC-2007) foi apresentado três variações desse problema. Neste trabalho será abordado o Problema de Horários de Cursos baseado em Currículo que é a terceira formulação apresentada na competição ITC-2007. Esse problema consiste em alocar horários e salas para as aulas das disciplinas de um determinado curso, considerando os conflitos entre disciplinas de um mesmo currículo e satisfazendo um conjunto de restrições. Diferentes heurísticas, metaheurísticas e técnicas inteligentes, cada vez mais eficientes, são implementadas para resolução desse problema, permitindo encontrar soluções de qualidade em tempo computacional viável. O objetivo deste trabalho é apresentar uma abordagem híbrida que utiliza a metaheurística *Biased Random-Key Genetic Algorithm*, a metaheurística *Simulated Annealing* como estratégia de busca local e a Cadeia de Kempe como método de pertubação, para resolução do problema, considerando os aspectos e as instâncias impostas no ITC-2007. Os resultados apontam que o algoritmo híbrido é capaz de produzir soluções viáveis e de qualidade, apresentado resultados superiores para um trabalho da literatura.

**Palavras-chave**: Otimização Combinatória, Problema de Horários de Cursos Universitários, Algoritmo Híbrido.

# Abstract

The University Course Timetabling Problem is a classic combinatorial optimization problem studied by several researchers. In the Second International Timetabling Competition (ITC-2007) three variations of this problem were presented. In this work, the Curriculum-Based Course Timetabling Problem, which is the third formulation presented in the ITC-2007 competition, will be addressed. This problem consists of allocating timetables and rooms for the classes of the subjects of a given course, considering the conflicts between subjects of the same curriculum and satisfying a set of restrictions. Different heuristics, metaheuristics and intelligent techniques, more and more efficient, are implemented to solve this problem, allowing to find quality solutions in viable computational time. The objective of this work is to present a hybrid approach that uses the Biased Random-Key Genetic Algorithm metaheuristic, the Simulated Annealing metaheuristic as a local search strategy and the Kempe Chain as a perturbation method to solve the problem, considering the aspects and instances imposed in the ITC-2007. The results show that the hybrid algorithm is capable of producing viable and quality solutions, presenting superior results for a work in the literature.

**Keywords**: Combinatorial Optimization, University Course Timetabling Problem, Hybrid Algorithm.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| ABC | *Artificial Bee Colony* |
| ACO | *Ant Colony Optimization* |
| ALNS | - *Adaptive Large Neighborhood Search* |
| BRKGA | *Biased Random-Key Genetic Algorithm* |
| CB-CTP | *Curriculum Based Course Timetabling Problem* |
| CMLS | *Competition-guided Multi-neighborhood Local Search* |
| FBT | *Feature-Based Tuning* |
| FP | *First Period* |
| GA | *Genetic Algorithm* |
| GCHH | *Generation Constructive Hyper-heuristics* |
| GPHH | *Generation Perturbative Hyper-heuristics* |
| GRASP | *Greedy Randomized Adaptive Search Procedures* |
| HC | *Hill Climbing* |
| ILS | *Iterated Local Search* |
| 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* |
| MOGA | *Multi-Objective Genetic Algorithm* |
| MOSA | *Multi-Objective Simulated Annealing* |

| | |
|---|---|
| PATAT | *International Conference on the Practice and Theory of Automated Timetabling* |
| PEB-CTP | *Post Enrolment Based Course Timetabling Problem* |
| PSO | *Particle Swarm Optimization* |
| RIICN | *Randomized Iterative Improvement with Composite Neighboring Algorithm* |
| RKGA | *Random-Key Genetic Algorithm* |
| RP | *Random Period* |
| SA | *Simulated Annealing* |
| SCHH | *Selection Constructive Hyper-heuristics* |
| SD | *Saturation Degree* |
| SPHH | *Selection Perturbative Hyper-heuristics* |
| TS | *Tabu Search* |
| UCTP | *University Course Timetabling Problem* |
| VNS | *Variable Neighborhood Search* |

# Sumário

# 1 Introdução

As instituições educacionais enfrentam diversos problemas semelhantes, um deles consiste em designar horários para as aulas das disciplinas, considerando um conjunto de solicitações e limitações. Na literatura, esse problema é conhecido como Problema de Programação de Horários. Existem três classificações para esse problema: Problema de Horários de Escolas, Problema de Horários de Cursos Universitários e o Problema de Horários de Exames (CARVALHO et al., 2016; MOREIRA et al., 2016).

O Problema de Horários de Cursos Universitários (UCTP, do inglês *University Course Timetabling Problem*) é um problema clássico de otimização combinatória enfrentado pelas instituições de ensino superior. O UCTP trata da atribuição das aulas das disciplinas de vários cursos universitários em um determinado número de salas e intervalos de tempo, satisfazendo um conjunto de restrições (SUSAN; BHUTANI, 2019; QUEIROZ; NEPOMUCENO, 2017).

Em termos de complexidade computacional, o UCTP é classificado como NP-Completo e NP-Difícil, no qual não existe um algoritmo que obtenha soluções ótimas em tempo polinomial (TEOH; ABDULLAH; HARON, 2015; POULSEN; BUCCO; BANDEIRA, 2014). O problema é NP-Completo para determinar a existência de uma solução válida (problema de decisão) e NP-Difícil para determinar a melhor solução dentre as soluções válidas (problema de otimização) (NEUKIRCHEN et al., 2014; TEOH; ABDULLAH; HARON, 2015).

Em 2002, 2007, 2011 e 2019 foram realizadas edições da Competição Internacional de Horário (ITC, do inglês *International Timetabling Competition*) com o intuito de intensificar o estudo de diferentes abordagens para solucionar o problema de cronograma educacional. Um dos principais apoiadores dessa competição é a Conferência Internacional sobre a Prática e Teoria do Horário Automatizado (PATAT, do inglês *International Conference on the Practice and Theory of Automated Timetabling*).

O UCTP é um tema de pesquisa ativa na comunidade científica, onde existe a necessidade de propor algoritmos com diferentes abordagens que produzam soluções de boa qualidade para diferentes instâncias em tempo hábil (HABASHI et al., 2018). Devido ao sucesso das edições anteriores, foi realizado em 2019, a quarta competição internacional de horários com o objetivo de criar um conjunto de dados robustos do mundo real e motivar pesquisas futuras sobre problemas complexos de horários de cursos universitários. A competição apresentou um conjunto de novas instâncias com diversas características e com algumas novidades.

As edições do ITC deram uma maior visibilidade ao tema. O impulsionamento na

realização de novas pesquisas também está relacionado com a recorrência do problema anualmente ou semestralmente em universidades e escolas. Diante disso, diversos métodos computacionais tem sido frequentemente apresentados e utilizados para resolver os problemas de cronograma educacional, entre eles destacam-se as metaheurísticas e as abordagens híbridas.

As metaheurísticas são métodos heurísticos, baseadas em população ou em solução única, que resolvem problemas de otimização de forma genérica, ou seja, não se atém a detalhes específicos do problema, mas trabalha em cima de um conjunto de soluções que será explorado em busca da melhor solução. As metaheurísticas não garantem a solução ótima, mas podem gerar soluções próximas da ótima em tempo computacional viável (ROCHA, 2013).

Os métodos híbridos são abordagens que combinam duas ou mais metaheurísticas. Esse tipo de abordagem surgiu para melhorar a qualidade das soluções produzidas por algoritmos de base populacional usando técnicas de busca local (SUSAN; BHUTANI, 2019; HABASHI et al., 2018). Os métodos híbridos apresentam-se como soluções altamente eficientes para o problema de cronograma de cursos, onde é possível alcançar soluções rápidas e otimizadas, no entanto, sem garantia de encontrar a melhor solução (JARDIM; SEMAAN; PENNA, 2015).

Este trabalho trata dos aspectos impostos no ITC-2007. Tratar dos aspectos do ITC-2019 se tornou inviável, pois quando a proposta deste trabalho foi elaborada e iniciada, o ITC-2019 estava em andamento, sendo finalizado apenas em setembro de 2020. No ITC-2007, foi apresentado duas variações do UCTP: O Problema de Horários de Cursos baseado em Pós-Inscrição (PEB-CTP, do inglês *Post Enrolment Based Course Timetabling Problem*) e o Problema de Horários de Cursos baseado em Currículo (CB-CTP, do inglês *Curriculum Based Course Timetabling Problem*). Esses problemas atendem ao mesmo propósito do UCTP. O que difere entre eles é que, no primeiro os alunos precisam selecionar as disciplinas que desejam se matricular antes que o cronograma seja construído, enquanto no segundo, os conflitos entre disciplinas é especificado pelos currículos de cada curso e não com base nas matrículas dos alunos.

O presente trabalho aborda o Problema de Horários de Cursos baseado em Currículo, tendo em vista, a sua importância, a dificuldade de resolvê-lo e a semelhança do problema com a realidade das universidades brasileiras. Além disso, no *website* do ITC-2007 foi disponibilizado um conjunto de instâncias para o CB-CTP com diferentes níveis de complexidade e há na literatura uma série de trabalhos que abordaram o mesmo tema, possibilitando a comparação de resultados.

Diante do exposto, este trabalho apresenta e avalia o desempenho de uma abordagem híbrida que utiliza o Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês *Biased Random-Key Genetic Algorithm*), a metaheurística Recozimento Simulado

(SA, do inglês *Simulated Annealing*) como busca local e uma Cadeia de Kempe como estratégia de pertubação. As principais razões que motivaram a escolha do BRKGA foram: A) A flexibilidade em lidar com problemas complexos de otimização combinatória; B) Os bons resultados da literatura obtidos com esta metaheurística em diversos problemas de otimização; C) Não foi encontrados trabalhos que utilizam o BRKGA para o problema tratado neste trabalho. Os motivos que justificam a utilização da metaheurística *Simulated Annealing* é a sua utilização satisfatória como algoritmo de busca local em diversos trabalhos da literatura como em Susan e Bhutani (2019), Monteiro et al. (2017), Segatto (2017), Rocha (2013), entre outros.

Segundo Jaengchuea e Lohpetch (2015), alguns estudos relataram que a hibridação entre métodos de busca local (por exemplo, Busca Tabu, Recozimento Simulado) e métodos populacionais (por exemplo, Algoritmo Genético, Otimização de Colônias de Formigas), trouxe melhores resultados para resolver o UCTP do que o uso de metaheurísticas isoladamente, pois ao combinar as boas propriedades do algoritmo baseado em busca local e o algoritmo baseado em população, o algoritmo híbrido pode ser capaz de fazer um equilíbrio entre a capacidade de exploração global e a local.

## 1.1 Objetivos

O objetivo principal deste trabalho é apresentar e avaliar uma abordagem híbrida que utiliza a metaheurística *Biased Random-Key Genetic Algorithm* com *Simulated Annealing* como estratégia de busca local e uma Cadeia de Kempe como método de pertubação para resolver o Problema de Horários de Cursos baseado em Currículos, visando comparar os resultados produzidos com trabalhos da literatura. Os objetivos específicos são:

- Investigar algoritmos de construção de soluções inicias para que a abordagem seja capaz de gerar soluções válidas para todas as instâncias;
- Analisar diferentes movimentos para geração de soluções vizinhas do problema;
- Analisar o desempenho da abordagem híbrida em um conjunto de instâncias do ITC-2007;
- Comparar os resultados da abordagem com os resultados obtidos no ITC-2007 e também com outros trabalhos da literatura;

## 1.2 Organização do Trabalho

O restante deste trabalho está organizado como segue: O Capítulo 2 apresenta uma fundamentação teórica sobre os principais conceitos que envolvem este trabalho; O

Capítulo 3 apresenta um conjunto de trabalhos da literatura que propuseram métodos para a resolução do problema abordado; O Capítulo 4 descreve os métodos de solução utilizados na abordagem proposta; O Capítulo 5 apresenta uma descrição detalhada do algoritmo híbrido e do ajuste de parâmetros; No Capítulo 6 são apresentados os resultados da fase inicial e final do algoritmo e, por fim, o Capítulo 7 descreve a conclusão dos resultados e trabalhos futuros.

# 2 Fundamentação Teórica

## 2.1 O Problema de Programação de Horários Educacionais

A programação de horários é uma tarefa de suma importância em diversas áreas, tais como indústrias, hospitais, empresas de transporte, escolas, universidades, entre outros. A solução desse problema não é uma tarefa simples e se torna mais complicada na medida que aumenta o número de associações e restrições. Desde a década de 60, esse problema vem sendo estudado. Gotlieb (1963) e Csima e Gotlieb (1964) tornaram-se os precursores apresentando os primeiros trabalhos sobre o tema (SOUZA, 2000; CARVALHO et al., 2016).

Schaerf (1999) estabelece três classificações principais e usuais para o problema de programação de horários educacionais. Essas classificações são descritas a seguir:

- **Problema de Horários de Escolas** (*School Timetabling Problem*): consiste na programação semanal dos horários das turmas de uma escola. Nessa categoria, existem um conjunto de turmas, professores e horários para a realização das aulas. Para cada turma há um conjunto de disciplinas, com suas cargas horárias e para cada professor há um conjunto de matérias e turmas que o mesmo lecionará. Uma característica desse problema é que, as turmas são conjuntos disjuntos de alunos que recebem suas aulas em uma mesma sala, ficando a cargo do professor o deslocamento para lecionar a matéria para determinada turma. O objetivo em questão é evitar que um professor esteja alocado a mais de turma em um mesmo horário e que uma turma não tenha aula com mais de um professor no mesmo horário (SCHAERF, 1999; SOUZA, 2000; QUEIROZ; NEPOMUCENO, 2017);

- **Problema de Horários de Cursos Universitários** (*University Course Timetabling Problem*): consiste na alocação das aulas das disciplinas de uma instituição de ensino superior, respeitando as disponibilidades e capacidades das salas. Nessa categoria, há um conjunto de disciplinas (Algoritmos, Circuitos Digitais, Lógica), e para cada disciplina uma carga horária, um conjunto de currículos (Sistemas de Informação, Ciência da Computação, Matemática), onde cada currículo envolve um conjunto de disciplinas, um conjunto de horários e um número limitado de salas. Diferente do *School Timetabling Problem*, nesse problema os alunos podem escolher em quais turmas das disciplinas do seu currículo irão matricular-se. Além disso, uma turma de uma disciplina pode ter alunos de currículos diferentes. Uma característica desse problema é que, há uma maior flexibilidade com relação aos horários disponibi-

lizados para a realização das aulas, como também, na configuração das aulas das disciplinas (SCHAERF, 1999; SOUZA, 2000; QUEIROZ; NEPOMUCENO, 2017);

- **Problema de Horários de Exames** (*Examination Timetabling Problem*): trata da alocação de exames para cursos universitários. Há um conjunto de alunos que estão matriculados em determinados cursos, um conjunto de exames e um conjunto de horários para realização dos exames. O objetivo é evitar sobreposições de exames que possuem estudantes em comum, sendo que um aluno tem um número determinado de exames a ser realizado em um dia (SCHAERF, 1999; SOUZA, 2000; QUEIROZ; NEPOMUCENO, 2017).

Em 2007, foi realizada a segunda Competição Internacional de Horário [1] (ITC-2007). O ITC-2007 foi dividido em três diferentes faixas, sendo elas: Faixa 1 – O Problema de Horários de Exames[2], Faixa 2 - O Problema de Horários de Curso baseado em Pós-Inscrição[3] e a Faixa 3 – O Problema de Horários de Curso baseado em Currículo[4]. Cada faixa da competição possui seu próprio conjunto de instâncias para testes. A definição dos problemas da faixa 1 e 2 são descritas a seguir:

- **Problema de Horários de Cursos baseado em Pós-Inscrição** (PEB-CTP, do inglês *Post Enrolment Based Course Timetabling Problem*): consiste em atribuir um conjunto de eventos (como aulas, cursos, palestras, laboratórios, tutoriais, seminários e assim por diante) a intervalos de tempo e salas adequadas de acordo com os dados de inscrição dos alunos, atendendo as restrições rígidas e flexíveis. Nessa variante, os alunos precisam selecionar os cursos que desejam frequentar antes que o horário seja construído. Após a inscrição do aluno, o horário é construído de tal maneira que todos os alunos possam participar dos eventos em que estão matriculados (LEWIS; PAECHTER; MCCOLLUM, 2007; JAENGCHUEA; LOHPETCH, 2015; NOGAREDA; CAMACHO, 2017);

- **Problema de Horários de Cursos baseado em Currículo** (CB-CTP, do inglês *Curriculum Based Course Timetabling Problem*): consiste no agendamento semanal das aulas de vários cursos universitários em um determinado número de salas e períodos de tempo, satisfazendo um conjunto de objetivos e restrições. Nessa variante, os conflitos entre os cursos são definidos pelos currículos e não com base nos dados das matriculas dos alunos (GASPERO; MCCOLLUM; SCHAERF, 2007; QUEIROZ; NEPOMUCENO, 2017).

[1] http://www.cs.qub.ac.uk/itc2007/
[2] http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index.htm
[3] http://www.cs.qub.ac.uk/itc2007/postenrolcourse/course_post_index.htm
[4] http://www.cs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index.htm

Essas duas variantes atendem ao mesmo propósito do UCTP, apesar de na essência serem problemas diferentes. Por esse motivo, muitos trabalhos na literatura agrupam essas duas variantes denominando-as de *Course Timetabling Problem*, *University Timetabling Problem*, *University Course Timetabling Problem* ou *Class/Teacher Timetabling Problem* (POULSEN; BUCCO; BANDEIRA, 2014).

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

o foco deste trabalho é o problema de horários de cursos baseado em currículo apresentado no ITC-2007. No *website* da competição foi disponibilizado um relatório técnico detalhado sobre o problema. Essa seção apresenta alguns detalhes da formulação do problema.

### 2.2.1 Formulação do Problema

O CB-CTP foi definido como um problema em que um conjunto de palestras ou aulas de vários cursos universitários devem ser atribuídas a intervalos de tempo e salas específicas, considerando um conjunto de restrições. As entidades envolvidas são:

- **Dias, Intervalos de Tempo e Períodos**: existe vários dias na semana em que as aulas podem ser ministradas (normalmente 5 ou 6). Cada dia é dividido em um número fixo de intervalos de tempo (igual para todos os dias). Um período é um par composto por um dia e um intervalo de tempo (ou período do dia). A quantidade de períodos disponíveis é o produto de dias e intervalos de tempo.
- **Cursos e Professores**: cada curso é composto por um número de alunos, um professor e um número de aulas a serem agendadas em períodos distintos. Cada curso possui um número mínimo de dias em que as aulas do curso devem ser distribuídas, além disso, existem alguns períodos em que o curso não pode ser agendado.
- **Salas**: cada sala tem uma capacidade, em termos de assentos disponíveis.
- **Currículos**: um currículo é composto por um grupo de cursos em que qualquer par de cursos tenha alunos em comum. Os conflitos entre cursos são produzidos de acordo com os currículos.

A solução do problema é a atribuição de um período (dia e horário) e uma sala para todas as aulas de cada curso, respeitando um conjunto de restrições.

### 2.2.2 Restrições

As restrições são classificadas como restrições rígidas (*hard constraints*) e flexíveis (*soft constraints*). Nenhuma das restrições rígidas podem ser violadas, pois inviabilizam a solução. Por outro lado, as restrições flexíveis são aquelas cujo atendimento é desejável, no entanto, a violação não inviabiliza a solução (MONTEIRO et al., 2017; SEGATTO et al., 2015). Essas restrições são descritas a seguir:

- **Restrições Rígidas**

  1 **Disciplinas**: Todas as aulas de uma disciplina devem ser agendadas e atribuídas a períodos distintos. Uma violação ocorre se uma aula não estiver agendada ou se duas aulas distintas forem agendadas no mesmo período.

  2 **Ocupação da Sala**: Duas aulas não podem ser realizadas na mesma sala no mesmo período. Duas aulas na mesma sala no mesmo período representam uma violação.

  3 **Conflitos**: As aulas de disciplinas do mesmo currículo ou ministradas pelo mesmo professor devem ser agendadas em períodos diferentes. Duas aulas conflitantes no mesmo período representam uma violação.

  4 **Disponibilidades**: Uma aula não pode ser alocada em um período em que o professor da disciplina estiver indisponível para ministra-la, da mesma forma que existem horários que determinadas disciplinas não podem ser alocadas.

- **Restrições Flexíveis**

  5 **Capacidade da sala**: O número de alunos que participam das aulas de uma disciplina deve ser menor ou igual ao número de cadeiras da sala onde a aula foi alocada. Cada aluno acima da capacidade conta como 1 ponto de penalidade.

  6 **Dias Mínimos de Trabalho**: As aulas de cada disciplina devem ser distribuídas no número mínimo de dias especificado. Cada dia abaixo do mínimo conta como 5 pontos de penalidade.

  7 **Compacidade do Currículo**: As aulas pertencentes a um currículo devem ser adjacentes umas às outras (ou seja, em períodos consecutivos). Para um determinado currículo, contabilizamos uma violação toda vez que há uma aula não adjacente a outra aula no mesmo dia. Cada aula isolada em um currículo conta como 2 pontos de penalidade.

  8 **Estabilidade da sala**: Todas as aulas de uma disciplina devem ser ministradas na mesma sala. Cada sala distinta usada para as aulas de uma disciplina conta como 1 ponto de penalidade.

As soluções sem violação das restrições rígidas são conhecidas como cronograma de horários viáveis, enquanto a satisfação das restrições flexíveis deve ser maximizada, pois elas determinam a qualidade da solução. Nesse contexto, a tabela de horário ótima tem que satisfazer todas as restrições rígidas e deve satisfazer as restrições flexíveis o máximo possível (JAENGCHUEA; LOHPETCH, 2015; TEOH; ABDULLAH; HARON, 2015).

### 2.2.3 Função Objetivo

Cada solução gerada possui um valor que reflete a sua qualidade. A função que calcula esse valor é chamada de função objetivo (F). Cada restrição violada aumenta o valor da função objetivo de acordo com seu peso. Uma solução viável deve atender a todas as restrições rígidas. A melhor solução S para o problema é aquela que minimiza o valor da função objetivo, dado pela formula:

$$\textbf{Minimizar } \mathbf{F(S)} = \sum_{v=1,}^{v} [w_v \times f_v(X)]$$

onde, V é o conjunto de restrições do problema com elementos $v \in$ V e identificados por $v = 1$ ... 8; $w_v$ é o peso da restrição de avaliação $f_v$(X); $f_v$(X) representa o número de violações da restrição corrente. Os pesos das restrições flexíveis foram definidos com base na formulação do ITC-2007. Para as restrições rígidas, é uma boa prática atribuir valores altos para que as mesmas sejam priorizadas (BARBOSA; SOUZA, 2011; SPINDLER; CHIWIACOWSKY, 2010). Neste trabalho, foi utilizado uma constante com peso igual a 10000. As soluções com custo igual ou superior a 10000 são inviáveis para o problema tratado.

### 2.2.4 Instâncias

Uma das grandes contribuições do ITC-2007 foi a disponibilização de um conjunto de instâncias com características de universidades reais que permite a comparação de diferentes métodos de solução. A competição realizada em 2007 foi dividida em três formulações, sendo que, cada formulação apresentava seu conjunto próprio de instâncias.

Para o CB-CTP foi disponibilizado um conjunto de 21 instâncias dividido em três níveis: conjunto de instâncias iniciais (comp01, comp02, comp03, comp04, comp05, comp06, comp07), conjunto de instâncias atrasadas (comp08, comp09, comp10, comp11, comp12, comp13, comp14) e conjunto de instâncias ocultas (comp15, comp16, comp17, comp18, comp19, comp20, comp21). Os organizadores garantem que existe pelo menos uma solução viável para todas as instâncias, mas não foi informado um valor ideal para a violação de restrições flexíveis.

Cada instância é um arquivo único dividido em cinco partes: cabeçalho, cursos, salas, currículos e restrições. O formato exato de cada instância é ilustrado na Figura 1.

Figura 1 – Arquivo de Entrada da Instância

```
Name: ToyExample
Courses: 4
Rooms: 2
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:
A 32
B 50

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

As informações do arquivo devem ser interpretadas da seguinte forma:

- *Courses*: <Nome do Curso> <Nome do Professor> <Número de Aulas> <Número Mínimo de Dias> <Número de Alunos>;
- *Rooms*: <Nome da Sala> <Capacidade>;
- *Curricula*: <Nome do Currículo> <Número de Cursos> <Membro ID> ... <Membro ID>;
- *Unavailability_Constraints*: <Nome do Curso> <Dia> <Período do Dia>.

Os dias e os períodos começam em 0. Por exemplo, a restrição **TecCos 3 2** estabelece que o curso TecCos não pode ser agendado no terceiro (2) período da quinta-feira (3).

A Tabela 1 apresenta as informações comuns em cada instância. Segundo Rocha (2013), o número de conflitos e o número de horários disponíveis influência diretamente

no tempo de execução do algoritmo e na busca de uma solução viável, pois quanto mais conflitos e menos disponibilidade, mais difícil será encontrar uma solução que não viole restrições fortes.

Tabela 1 – Informações sobre as Instâncias do ITC-2007

| **Nome** | **Cursos** | **Salas** | **Dias** | **Períodos por Dia** | **Currículos** | **Restrições** |
|---|---|---|---|---|---|---|
| comp01 | 30 | 6 | 5 | 6 | 14 | 53 |
| comp02 | 82 | 16 | 5 | 5 | 70 | 513 |
| comp03 | 72 | 16 | 5 | 5 | 68 | 382 |
| comp04 | 79 | 18 | 5 | 5 | 57 | 396 |
| comp05 | 54 | 9 | 6 | 6 | 139 | 771 |
| comp06 | 108 | 18 | 5 | 5 | 70 | 632 |
| comp07 | 131 | 20 | 5 | 5 | 77 | 667 |
| comp08 | 86 | 18 | 5 | 5 | 61 | 478 |
| comp09 | 76 | 18 | 5 | 5 | 75 | 405 |
| comp10 | 115 | 18 | 5 | 5 | 67 | 694 |
| comp11 | 30 | 5 | 5 | 9 | 13 | 94 |
| comp12 | 88 | 11 | 6 | 6 | 150 | 1368 |
| comp13 | 82 | 19 | 5 | 5 | 66 | 468 |
| comp14 | 85 | 17 | 5 | 5 | 60 | 486 |
| comp15 | 72 | 16 | 5 | 5 | 68 | 382 |
| comp16 | 108 | 20 | 5 | 5 | 71 | 518 |
| comp17 | 99 | 17 | 5 | 5 | 70 | 548 |
| comp18 | 47 | 9 | 6 | 6 | 52 | 594 |
| comp19 | 74 | 16 | 5 | 5 | 66 | 475 |
| comp20 | 121 | 19 | 5 | 5 | 78 | 691 |
| comp21 | 94 | 18 | 5 | 5 | 78 | 463 |

### 2.2.5 Arquivo de Saída

O arquivo de saída, ilustrado na Figura 2, deve ser um arquivo único de forma que cada linha represente a atribuição de uma aula no seguinte formato:

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

A primeira linha informa que uma palestra do SceCosC é ministrada na quinta-feira (3) no primeiro período (0) na sala B.

### 2.2.6 Validação das Soluções

No *website*[5] do ITC-2007 foi disponibilizado um validador com código fonte em C++ para validar as soluções geradas pelas abordagens desenvolvidas. O validador leva dois argumentos de linha de comando: o arquivo de entrada (ou instância) e o arquivo de saída e produz na saída padrão a avaliação da solução junto com a descrição detalhada de todas as violações de restrições rígidas e flexíveis. A Figura 3 ilustra um exemplo da saída padrão do validador.

[5] http://www.cs.qub.ac.uk/itc2007/curriculmcourse/validator.cc

Figura 2 – Arquivo de Saída

```
SceCosC B 3 0
SceCosC A 3 1
SceCosC A 4 0
ArcTec B 0 1
ArcTec B 1 1
ArcTec B 1 2
TecCos B 0 0
TecCos A 0 1
TecCos B 2 2
TecCos B 4 2
TecCos B 4 3
Geotec A 2 2
Geotec A 2 3
Geotec B 3 0
Geotec A 3 1
Geotec A 4 2
```

Fonte: *website* do ITC-2007

Figura 3 – Saída do Validador

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

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

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

Fonte: *website* do ITC-2007

### 2.2.7 Benchmarking

Os organizadores do ITC-2007 disponibilizaram um programa de *benchmark* para testar a velocidade da máquina de cada competidor na resolução dos problemas de

programação de horários. Esse programa foi projetado para possibilitar uma avaliação comparativa entre os diferentes algoritmos desenvolvidos para solucionar os problemas apresentados na competição.

O programa de *benchmark* informa por quanto tempo cada algoritmo pode ser executado nas instâncias de problemas de cronograma da competição. O *benchmark* só é adequado para máquinas individuais com um único processador. Não é adequado, por exemplo, para máquinas paralelas especializadas ou *clusters*.

Para executar o programa, é necessário o executável e o arquivo de dados *benchmarkinput.ttp*. Os dois arquivos precisam estar no mesmo diretório. O programa deve ser executado quando a máquina não está sendo usada para mais nada. Os itens a serem verificados são:

- Não há janelas desnecessárias abertas;
- Não há processos significativos em segundo plano do sistema operacional em andamento (por exemplo, backup);
- Não há usuários remotos no computador;
- Não há processos de compartilhamento de CPU em execução.

O programa irá relatar quanto tempo levou e, portanto, quanto tempo determinada máquina tem disponível para executar o algoritmo de tabela de horários (para cada instância).

### 2.2.8 Regras do ITC-2007

A competição apresentou um conjunto de regras que os participantes deveriam cumprir. Algumas delas serão apresentas a seguir, as demais não são discutidas neste trabalho, pois são regras que deveriam ser cumpridas no decorrer da competição.

- **Regra 6**: Cada participante da competição deve implementar um algoritmo para resolver o problema em uma máquina com um único processador e o algoritmo pode ser implementado em qualquer linguagem de programação.
- **Regra 7**: Cada algoritmo deve produzir cronogramas viáveis (ou seja, sem violação de restrições rígidas), minimizando o número de violações de restrições flexíveis.
- **Regra 9**: Os participantes devem comparar sua máquina com o programa fornecido para saber quanto tempo têm disponível para executar o algoritmo em suas máquinas.

- **Regra 10**: Os algoritmos devem tomar como entrada um arquivo do problema no formato descrito e produzir como saída uma solução no tempo de CPU permitido. A mesma versão do algoritmo deve ser usada para todas as instâncias. Ou seja, o algoritmo não deve saber qual instância está resolvendo. O programador não deve definir parâmetros diferentes para instâncias diferentes, embora o programa esteja fazendo isso automaticamente, então isso é aceitável.

- **Regra 11**: O algoritmo pode ser determinístico ou estocástico. Em ambos os casos, os participantes devem estar preparados para mostrar que esses resultados podem ser repetidos no tempo de computador determinado.

- **Regra 13**: Os participantes também devem apresentar uma descrição concisa e clara de seu algoritmo, para que, em princípio, outros possam implementá-lo.

- **Regra 15**: Para os finalistas da competição será solicitado o executável do algoritmo que será executado e testado pelos organizadores. O solucionador dos finalistas será executado novamente pelos organizadores em todas as instâncias. É responsabilidade do competidor garantir que todas as informações sejam fornecidas para permitir que os organizadores recriem a solução. O solucionador fornecido pelo finalista deve exigir como argumentos de linha de comando, nomes de arquivo de entrada e saída e, apenas para solucionadores estocásticos, a semente aleatória. Por exemplo (solucionador estocástico): > my_solver.exe dataset1.tim dataset1.sln 1542955064

As demais regras impostas na competição estão disponíveis em um relatório técnico descrito por Gaspero, McCollum e Schaerf (2007). Além das regras, foi definido um número máximo de 10 execuções independentes para cada instância. Logo, o vencedor da competição foi aquele que apresentou a média mais baixa das 10 execuções.

## 2.3 Abordagens para resolver Problemas de Programação de Horários Educaionais

Na literatura, existe uma variedade de métodos computacionais para solucionar o problema de programação de horários em insituições de ensino superior, entre eles estão as heurísticas, os métodos exatos, as metaheurísticas, as abordagens híbridas e as hiper-heurísticas. A subseção 2.3.1 apresenta as heurísticas de construção que podem produzir uma solução inicial que serve como entrada para uma técnica de otimização. A subseção 2.3.2 descreve as técnicas de otimização existentes para solucionar o problema.

### 2.3.1 Heurísticas Construtivas

As heurísticas construtivas foram uma das primeiras técnicas usadas para resolver o problema de programação de horários educacionais. Elas são baseadas na forma como o ser humano resolve o problema manualmente, ou seja, partindo de uma tabela de horário vazia, as aulas são acrescentadas uma a uma até que todas estejam alocadas (SCHAERF, 1999).

As heurísticas de coloração de grafo geralmente são utilizadas para criar uma solução inicial para o agendamento de horários educacionais. Essas heurísticas, também conhecidas como heurísticas de baixo nível, são essencialmente uma medida da dificuldade de agendar um evento ou uma aula (PILLAY; ÖZCAN, 2019; PILLAY, 2016a). As heurísticas existentes para selecionar um evento no processo de construção de uma tabela de horário inicial incluem:

- **Maior Matrícula** (LE, do inglês *Largest Enrolment*): Os eventos com maior número de alunos têm prioridade de agendamento. Logo, os eventos a serem agendados são ordenados em ordem decrescente.
- **Maior Grau** (LD, do inglês *Largest Degree*): Os eventos com maior número de conflitos têm prioridade no agendamento. O número de conflitos, ou seja, outros eventos com os quais o evento possui alunos em comum, é usado para avaliar a dificuldade do agendamento.
- **Maior Grau Ponderado** (LWD, do inglês *Largest Weighted Degree*): Os eventos com maior grau de ponderação têm prioridade no agendamento. Essa heurística é semelhante a heurística LD, no entanto, em vez de contar o número de eventos com os quais um evento possui alunos em comum, ela conta o número de alunos envolvidos em ambos os eventos.
- **Maior Grau de Cor** (LCD, do inglês *Largest Colour Degree*): Essa heurística é uma variação da LD que considera o número de conflitos com os eventos que já foram agendados.
- **Grau de Saturação** (SD, do inglês *Saturation Degree*): Os eventos com menos períodos disponíveis são agendados primeiro. Essa heurística considera o número de períodos viáveis para cada evento, ou seja, períodos que não resultam em violações de restrições rígidas no ponto atual de construção do cronograma.

As heurísticas LE, LD e LWD são estáticas e 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 em ordem crescente ou decrescente de acordo com o valor heurístico e são alocados em ordem em um período viável. Caso não

haja períodos viáveis no momento da alocação, o evento não será alocado ou será alocado para um período selecionado aleatoriamente e o custo da restrição rígida é incrementado. Existem três heurísticas para selecionar um período para o evento:

- **Primeiro Período** (FP, do inglês *First Period*): O evento é alocado no primeiro período viável encontrado. Um período viável é aquele que não infrinja restrições rígidas quando o evento é alocado a ele.
- **Período Aleatório** (RP, do inglês *Random Period*): O período é escolhido aleatoriamente entre todos os períodos viáveis.
- **Período de Custo Mínimo** (MCP, do inglês *Minimum Cost Period*): Calcula-se o custo de violação de restrições flexíveis para cada um dos períodos viáveis de acordo com o cronograma atual parcialmente construído. O evento é alocado no período com custo mínimo de restrição flexível.

As heurísticas LCD e SD são dinâmicas, pois o valor heurístico de cada evento não alocado muda na medida que os eventos são alocados na tabela de horário. No início do processo de construção do cronograma de horários todos os eventos possuem o mesmo grau de saturação, ou seja, o número de períodos do cronograma. Na medida que o cronograma é construído, o grau de saturação de um evento X que possui alunos em comum com o evento Y recém alocado é diminuído. Da mesma forma, a heurística LCD tem valor inicial zero para todos os eventos. Conforme os eventos são alocados, o valor heurístico para um evento não alocado é incrementado (PILLAY; ÖZCAN, 2019; PILLAY, 2016a).

### 2.3.2 Abordagens de Otimização

Os métodos exatos, as metaheurísticas, as abordagens híbridas e as hiper-heurísticas são as técnicas de otimização utilizadas para resolver o problema de cronograma educacional. É possível encontrar na literatura alguns trabalhos que utilizaram métodos exatos, baseados em técnicas da pesquisa operacional como: 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).

Os métodos exatos apresentam algumas dificuldades de utilização, visto que consomem muito tempo computacional para fornecer um solução apropriada para problemas de otimização em um espaço de busca de alta dimensão (BEHESHTI; SHAMSUDDIN, 2013). Segundo Schaerf (1999), o UCTP é classificado como NP-Completo para a maioria das

formulações, e assim, uma solução exata só pode ser garantida para instâncias pequenas. O fato da versão de otimização do problema ser classificado como NP-Difícil é o que limita o uso exclusivo de métodos exatos.

Devido a alta complexidade computacional, a exploração de espaços de busca de soluções aplicando metaheurísticas atua de maneira mais eficiente na resolução do problema. As metaheurísticas permitem obter soluções satisfatórias e de boa qualidade, melhores que as manuais, com menos esforço humano e em tempo computacional viável. No entanto, sem a garantia da otimalidade (JARDIM; SEMAAN; PENNA, 2015).

As metaheurísticas são uma classe de algoritmos que empregam algum mecanismo que lhe fornece a capacidade de escapar de ótimos locais (NETO; MARIANO; FARIAS, 2017). As metaheurísticas são classificadas como abordagens de base populacional, que são algoritmos inspirados na natureza, e abordagens de solução única (MATIAS; FAJARDO; MEDINA, 2018; HABASHI et al., 2018). Uma abordagem baseada na população visa encontrar a melhor solução no espaço de busca e uma abordagem de base única trabalha com uma única solução e tenta melhorar durante o processo de busca (VRIELINK et al., 2019).

Diversas metaheurísticas têm sido aplicadas com sucesso para problemas de agendamento educacional. Entre elas, destacam-se as baseadas em população como: 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*). E também as metaheurísticas baseadas em soluções únicas como: Busca Tabu (TS, do inglês *Tabu Search*), Recozimento Simulado (SA, do inglês *Simulated Annealing*), Subida da Encosta (HC, do inglês *Hill Climbing*) e Busca de Vizinhança Variavél (VNS, do inglês *Variable Neighborhood Search*) (VRIELINK et al., 2019).

As metaheurísticas baseadas em população são as que possui alta capacidade no tratamento de problemas complexos de otimização, pois apresentam capacidade de exploração global e local (BEHESHTI; SHAMSUDDIN, 2013). Segundo Habashi et al. (2018), muitos estudos concluíram que os algoritmos de busca local podem apresentar um desempenho melhor com relação ao tempo de execução, mas sem a garantia de encontrar soluções ótimas, enquanto os algoritmos baseados em população garantem a exploração de novas áreas e podem encontrar soluções ainda melhores, próximo da solução ótima.

Contudo, as abordagens mencionadas anteriormente não são as mais eficientes para resolver o problema de cronograma, pois os conjuntos de dados são diversos. Consequentemente, os métodos híbridos e as hiper-heurísticas tem atraído mais atenção por apresentar melhor qualidade e desempenho na solução de problemas (MATIAS; FAJARDO; MEDINA, 2018).

As abordagens híbridas surgiram para melhorar a qualidade das soluções produzidas por algoritmos de base populacional usando técnicas de busca local (SUSAN; BHUTANI, 2019; HABASHI et al., 2018). Portanto, é evidente que os métodos e técnicas mencionados apresentam fragilidades, combinando esses métodos é possível eliminar as fragilidades e explorar efetivamente a força de cada algoritmo, levando a uma abordagem híbrida superior (MATIAS; FAJARDO; MEDINA, 2018).

Por outro lado, as hiper-heurísticas são métodos de busca que operam em um nível mais alto de abstração. Em outras palavras, a hiper-heurística é um método de busca em que várias heurísticas são combinadas e adaptadas. A diferença entre a metaheurística e a hiper-heurística é que a hiper-heurística busca soluções no espaço heurístico em vez de buscar no espaço de soluções simples (VRIELINK et al., 2019).

# 3 Trabalhos Relacionados

Este capítulo apresenta uma série de trabalhos relacionados ao problema de horários de cursos baseado em currículo. Inicialmente, será apresentado algumas revisões encontradas na literatura. Essas revisões fornecem um conjunto de informações sobre as principais técnicas utilizadas para resolver o CB-CTP. Logo após, é realizado uma descrição de trabalhos da literatura do ano de 2015 à 2021. Todos os trabalhos descritos neste capítulo utilizaram as 21 instâncias dispostas no ITC-2007 para o CB-CTP.

Babaei, Karimpour e Hadidi (2015) realizaram uma análise das abordagens utilizadas para resolver problemas de horários de cursos universitários. As abordagens apresentadas são as seguintes:

1 Técnicas baseadas em pesquisas operacionais, incluindo Coloração de Grafos, Método de Programação Inteira/Linear e Programação de Satisfação de Restrições.

2 Abordagens metaheurísticas, incluindo abordagens populacionais e abordagens baseadas em solução única, nas quais as abordagens populacionais incluem Algoritmos Genéticos, Otimização de Colônias de Formigas, Algoritmo Memético, Algoritmo de Busca Harmônica, Otimização de Enxame de Partículas, Colônias de Abelhas Artificiais e os algoritmos de solução única incluem Busca Tabu, Busca de Vizinhança Variável, Melhoria Iterativa Aleatória com o Algoritmo Vizinho Composto (do inglês, *Randomized Iterative Improvement with Composite Neighboring Algorithm* - RIICN), Recozimento Simulado e Grande Dilúvio;

3 Abordagens multi-objetivo e multicritério;

4 Novas abordagens inteligentes, como abordagens híbridas (Exemplo: Heurística Sequencial e Recozimento Simulado; Heurísticas Construtivas, Algoritmo de Melhoria Iterativa Aleatória e Recozimento Simulado; Algoritmo Memético e Busca Tabu), abordagens baseadas em inteligência artificial, abordagens baseadas em teoria difusa, abordagens baseadas em algoritmos de *clustering*;

5 Abordagem baseada em sistemas multi-agente distribuído.

Babaei, Karimpour e Hadidi (2015) mencionam que as abordagens baseadas em métodos de pesquisa operacional não apresentam uma boa eficiência na solução desse problema, tendo em vista que, o tempo computacional aumenta exponencialmente à medida que o tamanho do problema aumenta. Diante disso, a exploração do espaço de busca por soluções tem um desempenho mais eficiente com a aplicação de metaheurísticas e novas técnicas inteligentes.

Na revisão de Teoh, Abdullah e Haron (2015) foram examinadas as propriedades dos problemas de agendamento acadêmico e abordado as várias técnicas e estratégias metaheurísticas usadas para resolver os problemas. A partir desse estudo, foi possível descrever as metaheurísticas utilizadas em diversos trabalhos, entre elas, destaca-se: Busca Tabu, Algoritmo Genético, Recozimento Simulado, Otimização de Enxame de Partículas, Lógica *Fuzzy*, Otimização de Colônias de Formigas e Hiper-heurísticas. Levando em conta aspectos como a exploração do espaço de busca, pode-se observar que o algoritmo genético e o algoritmo de otimização de enxame de partículas parecem ter um melhor desempenho na resolução dos problemas de cronograma acadêmico.

Pillay (2016b) apresenta, em seu trabalho de revisão, uma visão geral e uma análise das hiper-heurísticas para resolver os problemas de horários educacionais, incluindo 1) o problema de horários de exames universitários, 2) horários de cursos universitários e 3) horários de escolas. As hiper-heurísticas foram classificadas em 4 categorias: Hiper-heurísticas Construtivas de Seleção (*Selection Constructive Hyper-heuristics* - SCHH), Hiper-heurísticas Perturbativas de Seleção (*Selection Perturbative Hyper-heuristics* - SPHH), Hiper-heurísticas Construtivas de Geração (*Generation Constructive Hyper-heuristics* - GCHH) e Hiper-heurísticas Perturbativas de Geração (*Generation Perturbative Hyper-heuristics* - GPHH). Para o problema de cronograma de exames universitários foi apresentado um conjunto de estudos que utilizaram as hiper-heurísticas SCHH, SPHH, GCHH e GPHH. Para o problema de horários de cursos universitários foi analisados alguns trabalhos que fizeram uso das hiper-heurísticas SCHH, SPHH e GPHH. Já para o problema de horário escolar foi examinado diversos trabalhos que utilizaram a SCHH, SPHH e GCHH. Para cada classe de problema foi apresentado o estudo com melhor desempenho considerando cada categoria de hiper-heurísticas.

No trabalho de Altunay e Eren (2017) foi apresentado uma revisão da literatura sobre o problema de programação de horários de cursos, onde foi analisado os principais pontos dos estudos realizados sobre o problema desde 1960. Nesse estudo, os autores apresentaram algumas informações sobre a definição e recursos do problema, como também, as abordagens de solução que foram aplicadas pelos pesquisadores para solucionar o problema. Mais de 200 publicações científicas foram examinadas. Os métodos de solução utilizados nos estudos foram avaliados em três categorias que são: Abordagens baseadas em Pesquisa Operacional, Abordagens Metaheurísticas e Novas Abordagens.

Altunay e Eren (2017) descrevem que as abordagens baseadas em pesquisa operacional utilizadas em vários estudos para solucionar o problema foram: programação matemática (programação inteira/programação linear), coloração gráfica, programação de restrições, modelos de rede e métodos de modelagem multi-critério/multiuso. No entanto, os autores pontuam que esses métodos exigem sistemas de computadores poderosos e muito tempo de computação, não sendo adequados para serem utilizados na solução do

problema para instâncias grandes de universidade e faculdades. O fato de que a melhor solução não pode ser obtida em tempo polinomial direcionou os pesquisadores a abordagens metaheurísticas (solução única ou algoritmo baseado em população), entre elas estão: Busca Tabu, Recozimento Simulado, Busca de Vizinhança Variável, Busca Local, Algoritmo Genético, Otimização de Colônias de Formigas, Otimização de Enxame de Partículas, Algoritmo Memético e Algoritmo de Busca Harmônica. Além disso, novas abordagens disseminaram nos últimos anos, esses novos métodos são: Algoritmos Híbridos (Exemplo: Recozimento Simulado e Otimização de Colônias de Formigas; Busca Tabu e Otimização de Colônias de Formigas; Algoritmo Memético e Recozimento Simulado), Métodos Difusos, Algoritmos de *Clustering*, Sistemas de Apoio à Decisão/ Sistemas Especialistas, Redes Neurais Artificiais, Sistemas de Múltiplos Agentes e Hiper-heurísticas. Com esse estudo, os autores observaram que nos últimos anos, os algoritmos híbridos e as hiper-heurísticas tem ganhado mais destaque na solução do problema.

Hosny (2019) realizou uma revisão sobre os diferentes problemas de horários em universidades como: o Problema de Horário de Exame, o Problema de Horário de Curso, o Problema de Horário da Equipe (*do inglês, Staff Timetabling Problem*) e propôs um novo problema, que é o Problema de Horário do Projeto (*do inglês, Project Timetabling Problem*). Além disso, foi apresentado e discutido alguns estudos de caso de instituições de ensino do Oriente Médio que utilizaram diferentes abordagens heurísticas e metaheurísticas para resolução desses problemas. Esse trabalho foi limitado a artigos publicados na região do Oriente Médio durante o período de 2011 a 2018. Essa limitação se deve ao grande número de artigos publicados em todo o mundo nesta área de pesquisa. Os resultados apontam que os algoritmos evolutivos, em especial os algoritmos genéticos, são as abordagens mais populares para resolução desses problemas. Outras metaheurísticas como Algoritmos de Abelhas (*Bees Algorithm*), Otimização de Enxame de Partículas (*Particle Swarm Optimization*), Busca Tabu (*Tabu Search*), Recozimento Simulado (*Simulated Annealing*), Busca de Dispersão (*Scatter Search*) e Algoritmos Demoníacos (*Demon Algorithms*) também são utilizados, independentemente ou hibridizados com outros algoritmos.

Os resultados da pesquisa realizada por Hosny (2019) indicam que existem várias direções que ainda estão em aberto para uma investigação mais aprofundada. Entre as orientações de pesquisa destacadas está: o foco na conveniência das partes interessadas; investigar o uso de metaheurísticas para gerar soluções iniciais que serão melhoradas por outra metaheurística; explorar novas abordagens metaheurísticas que ganharam popularidade recentemente, isso inclui, o Algoritmo Vaga-lume (*Firefly Algorithm*), o Algoritmo de Acasalamento de Abelhas (*Bee Mating Algorithm*), o Algoritmo de Morcego (*Bat Algorithm*), o Algoritmo Big-Big Big-Crunch (*Big-Bang Big-Crunch Algorithm*), a Busca Cuco (*Cuckoo Search*), o Campeonato da Liga (*League Championship*), o Algoritmo de Salto de Sapos (*Frog Leaping Algorithm*), etc. Além disso, o problema de cronograma de projeto é outra área que está em aberto para uma investigação mais aprofundada.

Segatto et al. (2015) utilizaram a metaheurística *Greedy Randomized Adaptive Search Procedures* (GRASP) com a metaheurística *Simulated Annealing* para a busca local e *Path-Relinking* para refinamento das soluções locais. Três movimentos foram utilizados para geração de vizinhos, entre eles: Move, Swap e Cadeia de Kempe. O GRASP híbrido foi implementado para melhorar resultados obtidos por outra versão do GRASP que não incluía a Cadeia de Kempe. Os testes computacionais apontam que a versão do GRASP que utiliza Cadeia de Kempe gerar soluções com qualidade superior a versão sem Cadeia de Kempe. No entanto, os resultados dos primeiros colocados do ITC-2007 foram superiores ao GRASP híbrido.

Soria-Alcaraz et al. (2016) descrevem um algoritmo de Busca Local Iterada (ILS, do inglês *Iterated Local Search*) hibridizado com uma hiper-heurística generativa que gera heurísticas baseadas em operações de adição e exclusão para resolver o problema de horário de curso baseado em currículo e o problema baseado em pós-inscrição do ITC-2007. A operação de exclusão tem a função de remover um evento programado, enquanto a operação de adição atribui um evento não programado a um intervalo de tempo com base em um conjunto de heurísticas de baixo nível. Os resultados mostram que a abordagem apresentam resultados competitivos com os resultados da literatura para os dois domínios do problema.

Bellio et al. (2016) utilizaram a metaheurística *Simulated Annealing* e aplicaram dois métodos para ajustar os parâmetros da metaheurística: *Feature-Based Tuning* (FBT) e *F-Race*. O algoritmo utiliza dois movimentos para geração de vizinhos: Move e Swap. O SA proposto difere do SA padrão no resfriamento da temperatura e na critério de parada. O resfriamento da temperatura é baseado em corte, ou seja, em vez de definir um número fixo de soluções em cada nível da temperatura, o algoritmo pode diminuir a temperatura caso o número de aceites de soluções ruins tenha atingido um certo parâmetro. Em relação ao critério de parada, os autores utilizaram um número limite de iterações equivalente ao tempo disponível para execução do algoritmo. Os resultados obtidos com os parâmetros ajustados pelo método FBT foram os mais competitivos com os da literatura, apresentando resultados superiores para 10 instâncias das 21.

Kiefer, Hartl e Schnell (2017) apresentaram uma metaheurística baseada na busca adaptativa de grande vizinhança (ALNS, do inglês *Adaptive Large Neighborhood Search*) para resolver o CB-CTP. A variante de ALNS incorpora vários operadores de destruição e reparo, com probabilidade de seleção tendenciosa. O algoritmo foi capaz de gerar resultados superiores a outras abordagens da literatura e superou os resultados dos melhores algoritmos do ITC-2007.

Monteiro et al. (2017) desenvolveram um algoritmo híbrido que utiliza a metaheurística *Iterated Local Search* com a metaheurística *Simulated Annealing* como busca local, e Cadeia de Kempe como movimento de perturbação. Quatro movimentos foram utilizados

para geração de vizinhos, entre eles: *Move*, *Swap*, *Time Move* e *Room Move*. O algoritmo foi comparado com as melhores soluções obtidas no ITC-2007 e também com dois trabalhos da literatura. Os resultados demonstram que o algoritmo foi eficiente, apresentando soluções de qualidade. O ILS+SA obteve a média de resultados 38% inferior ao primeiro colocado e 14% superior ao quarto colocado no ITC-2007. O ILS+SA obteve um resultado 19% superior quando comparado com a média dos resultados obtidos por Segatto et al. (2015) e 52% inferior quando comparado com os melhores resultados obtidos por Kiefer, Hartl e Schnell (2017). Este resultado evidencia que o ILS+SA, através do processo de perturbação (Cadeia de Kempe), foi mais eficiente na exploração do espaço de soluções que o método GRASP implementado por Segatto et al. (2015). Os autores apontam que se o algoritmo tivesse participado do ITC-2007, ele ocuparia a quarta colocação na classificação final.

Segatto (2017) apresentou melhorias para um algoritmo GRASP da literatura, visando compará-lo a resultados mais recentes. Nesse estudo, foi realizada uma análise detalhada de várias vizinhanças para ser incorporado ao GRASP na tentativa de obter resultados competitivos para as instâncias do CB-CTP. A análise experimental determinou que o GRASP com a busca local *Simulated Annealing* e as vizinhanças denominadas de *Lecture Move* e Cadeia de Kempe Estendido com Restrição de Sala são os métodos capazes de obter resultados competitivos. Os resultados do trabalho mostraram que o algoritmo alcançou uma média de melhora de 49.31% em relação a versão anterior do GRASP. Por outro lado, o algoritmo apresentou resultados superiores e inferiores quando comparados aos trabalhos da literatura.

Akkan e Gülcü (2018) modelaram o problema de CB-CTP como um problema de otimização multi-objetivo, onde um objetivo é minimizar o número de restrições flexíveis violadas e o outro é a robustez. Um cronograma é considerado robusto se um pedido de alteração de horário de um evento não resulta em um cronograma com qualidade inferior em termos de penalidade de restrições. Os autores relatam que são os primeiros a considerar a robustez como um objetivo para o problema de horários de cursos universitários. O problema foi resolvido com um Algoritmo Genético Híbrido Multi-Objetivo (MOGA, do inglês *Multi-Objective Genetic Algorithm*) que faz uso dos algoritmos *Hill Climbing* e *Simulated Annealing* com o objetivo de obter uma boa aproximação da fronteira ótima de Pareto. Os resultados experimentais, realizados com as 21 instâncias do ITC-2007, mostraram que a abordagem permite obter soluções de qualidade e com alta robustez. Os resultados coletados foram comparados com outros trabalhos que mediram apenas a penalidade de violações flexíveis para soluções viáveis. Desse modo, os autores declaram que o algoritmo precisa ser melhorado para atingir valores de penalidade próximos aos melhores resultados conhecidos na literatura.

Fajrin e Fatichah (2020) utilizou um algoritmo genético para minimizar a violação de restrições flexíveis do CB-CTP. Um mecanismo de cruzamento de ordem multi-pai foi

utilizado para modificar o cruzamento clássico e minimizar a violação das restrições. Os experimentos realizados com as 21 instâncias do ITC-2007 mostram que o mecanismo de cruzamento de ordem multi-pai no algoritmo genético fornece um melhor desempenho quando comparado com o algoritmo genético clássico, pois o mecanismo de cruzamento aumenta o poder de exploração no espaço de busca e fornece resultados eficazes. Além disso, os resultados mostraram que o algoritmo proposto foi capaz de alcançar bons resultados em comparação com o AG de Akkan e Gülcü (2018).

Song et al. (2021) propuseram um novo algoritmo de busca local multi-vizinhança guiada por competição (CMLS, do inglês *Competition-guided Multi-neighborhood Local Search*). Os autores descrevem três contribuições do algoritmo proposto, sendo elas: 1) a forma de combinar várias vizinhanças, sendo que, apenas uma vizinhança é selecionada em cada iteração. Especificamente, seis operadores de vizinhanças são adotados e uma nova forma de combinação é introduzido; 2) a concepção de duas regras heurísticas para determinar a probabilidade de seleção das vizinhanças; 3) uma estratégia de reinício baseada em competição. O algoritmo de busca local com várias vizinhanças utiliza uma estratégia de reinicialização baseada em competição para melhorar iterativamente a solução inicial. Em cada iteração do algoritmo, duas buscas locais multi-vizinhanças baseadas em *Simulated Annealing* são empregadas separadamente, cada uma com um conjunto de probabilidades diferente. Esse procedimento retorna duas soluções diferentes, aquela que apresenta menor custo de violação de restrições é selecionada como solução inicial para a iteração seguinte do algoritmo. O processo que envolve a execução de duas buscas locais de forma independente e a seleção da melhor solução resultante para a próxima rodada do algoritmo é determinado de estratégia de reinicialização baseada em competição. Os resultados obtidos para as 21 do ITC-2007 mostram que o CMLS é altamente competitivo quando comparado com outros 6 algoritmos de última geração da literatura. O CMLS foi capaz de melhorar os resultados médios de 16 instâncias.

A Tabela 2 apresenta uma descrição simplificada dos estudos retratados. Para cada estudo é especificado a Abordagem, Tempo, Max, Melhor e Média. (1) Abordagem: especifica o método utilizado, (2) Tempo: especifica se o tempo de execução do algoritmo foi definido pelo programa de *benchmark* do ITC-2007 ou foi utilizado um tempo superior, (3) Max: informa a quantidade de execuções realizadas para cada instância, (4) Melhor: informa se o estudo fornece o melhor valor da função objetivo para cada instância e (5) Média: informa se o estudo fornece o valor médio da função objetivo para as execuções do algoritmo. Os campos vazios são campos que não foram informados nos estudos.

Tabela 2 – Descrição Simplificada dos Trabalhos Relacionados

| **Autores** | **Abordagem** | **Tempo** | **Max** | **Melhor** | **Média** |
|---|---|---|---|---|---|
| Segatto et al. (2015) | Grasp com *Simulated Annealing* e *Path-Relinking* | ITC-2007 | 10 | √ | |
| Soria-Alcaraz et al. (2016) | *Iterated Local Search* com Hiper-Heurística Generativa | ITC-2007 | 10 | √ | |
| Bellio et al. (2016) | *Simulated Annealing* | ITC-2007 | 31 | | √ |
| Kiefer, Hartl e Schnell (2017) | *Adaptive Large Neighborhood Search* | ITC-2007 | 10 | √ | √ |
| Monteiro et al. (2017) | *Iterated Local Search* com *Simulated Annealing* | ITC-2007 | 10 | √ | √ |
| Segatto (2017) | Grasp com *Simulated Annealing* | ITC-2007 | 20 | | √ |
| Akkan e Gülcü (2018) | *Multi-Objective Genetic Algorithm*<br>*Hill Climbing*<br>*Simulated Annealing* | Superior | 30 | | √ |
| Fajrin e Fatichah (2020) | Algoritmo Genético | | 30 | √ | √ |
| Song et al. (2021) | *Competition-guided Multi-neighborhood Local Search* | ITC-2007 | 30 | √ | √ |

# 4 Métodos de Solução

Diversos trabalhos da literatura utilizaram, com sucesso, diferentes abordagens para solucionar o problema de horários de cursos baseado em currículo. Neste trabalho, foi implementado uma abordagem híbrida, que utiliza a metaheurística *Biased Random-Key Genetic Algorithm* com a metaheurítica *Simulated Annealing* e uma estratégia de pertubação. Esse capítulo descreve os métodos utilizados na resolução do problema.

## 4.1 *Biased Random-Key Genetic Algorithm*

Algoritmos Genéticos são metaheurísticas, inspiradas na Teoria da Evolução, capazes de lidar com diversos problemas de otimização combinatória. Os algoritmos genéticos desenvolvem uma população de indivíduos aplicando o princípio de sobrevivência de Charles Darwin, ou seja, os indivíduos evoluem ao longo do tempo através da seleção natural, sendo que, os indivíduos mais aptos apresentam maior chance de produzir descendentes. A evolução é realizada aplicando um conjunto de operadores: seleção, cruzamento (*crossover*) e mutação.

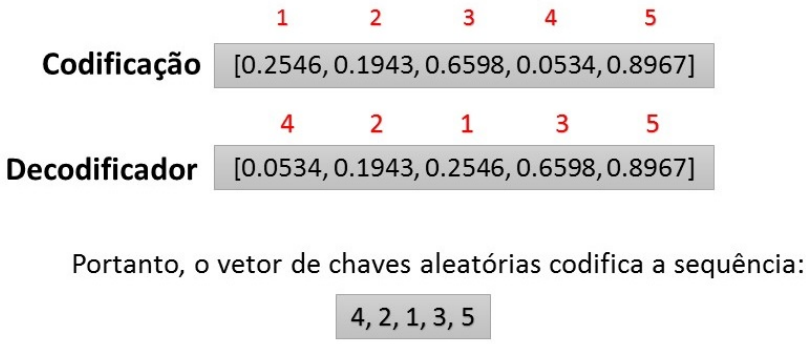
Apesar de alcançar resultados satisfatórios na resolução de problemas em várias áreas, o AG clássico possui um obstáculo que é a convergência prematura. Esse obstáculo faz com que as soluções presentes na população se estabilizem, tornando o algoritmo incapaz de manter a exploração do espaço de busca devido à falta de diversidade do material genético (ALIXANDRE; DORN, 2017).

Para evitar a convergência prematura do algoritmo genético foram propostas várias estratégias de melhoria ao longo dos anos (ALIXANDRE; DORN, 2017). Entre elas, destaca-se o Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês *Biased Random-Key Genetic Algorithm*) que é uma metaheurística variante do Algoritmo Genético de Chaves Aleatórias (RKGA, do inglês *Random-Key Genetic Algorithm*) proposto por Bean (1994). O BRKGA foi proposto por Gonçalves e Resende (2011) com o objetivo de selecionar elementos de um subconjunto específico de soluções (MOURA, 2018).

Os cromossomos em um BRKGA são representados como um vetor de números reais gerados aleatoriamente no intervalo [0,1]. A população inicial é composta por $P$ vetores de chaves aleatórias. Para obter a solução de um problema é necessário submeter os vetores de chaves aleatórias a um algoritmo determinístico, chamado decodificador. O decodificador recebe como entrada um cromossomo e associa a ele uma solução do problema de otimização. Cada solução é avaliada recebendo um valor que representa sua aptidão. O decodificador desenvolvido por Bean (1994) ordena o vetor de chaves aleatórias

e utiliza os índices das chaves para representar uma sequência (GONÇALVES; RESENDE, 2011). A Figura 4 exemplifica esse processo.

Figura 4 – Codificação e Decodificação de um RKGA




Fonte: Autoria Própria

Por ter uma fundamentação nos algoritmos genéticos, o BRKGA evolui uma população ao longo de várias gerações. Cada indivíduo da população é avaliado e classificado em dois grupos de acordo com o valor de aptidão. Esses grupos são denominados de: População Elite ($P_e$), composta pelos indivíduos que possuem os melhores valores de aptidão, e a População Não Elite ($P$ - $P_e$), composta pelo restante dos indivíduos da população. Isso é feito pra evitar a convergência prematura do método (MAINIERI, 2014; MOURA, 2018; MORO, 2017).

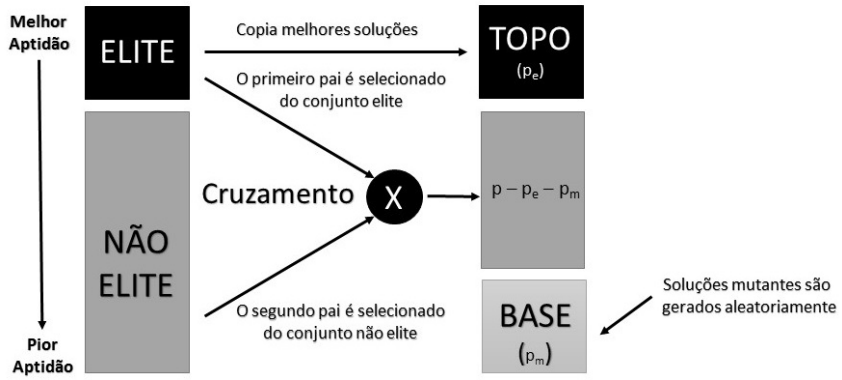
Após a classificação e separação dos cromossomos, inicia-se o processo de formação das próximas gerações (k + 1). Nesse processo, todos os indivíduos da população elite da geração k são copiados para a geração k + 1, caracterizando o princípio darwinista do elitismo. Em seguida, uma quantidade $P_m$ de mutantes é gerada para compor a próxima geração, inserindo diversidade na população. Um mutante é simplesmente um indivíduo gerado da mesma maneira que é gerada a população inicial (MAINIERI, 2014; MOURA, 2018; MORO, 2017; JUNIOR, 2017).

Com $P_e$ e $P_m$ introduzidos na população, é necessário produzir $P$ - $P_e$ - $P_m$ para completar os indivíduos da população da geração k + 1. Esses indivíduos são produzidos através do cruzamento uniforme parametrizado de Spears e Jong (1991). Quando a próxima população estiver concluída, os valores de aptidão serão calculados para todos os vetores de chave aleatória recém-criados (GONÇALVES; RESENDE, 2011).

A diferença de um BRKGA para um RKGA está na maneira como os pais são selecionados para o cruzamento. No RKGA, os pais são selecionados aleatoriamente da população inteira, enquanto que, no BRKGA, um pai é selecionado da população elite e o outro da não elite. Além disso, é permitido no BRKGA a repetição de um pai na seleção para o cruzamento, portanto, um mesmo indivíduo pode gerar mais de um filho

na mesma geração. Esse processo de seleção amplia a participação dos indivíduos mais aptos, ampliando também a atribuição dos melhores genes nos indivíduos descendentes (MAINIERI, 2014; MOURA, 2018; MORO, 2017). A Figura 5 ilustra o processo de evolução.

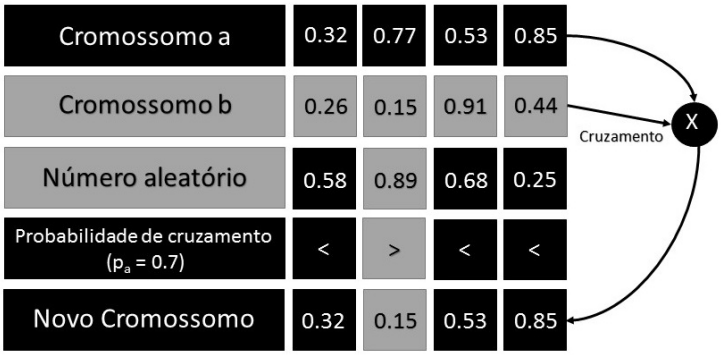
Figura 5 – Transição da geração k para a geração k+1 em um BRKGA.




Fonte: Adaptado de Gonçalves e Resende (2011)

A Figura 6 ilustra o processo de cruzamento para dois vetores de chaves aleatórias. Para cada posição do cromossomo é determinado um número aleatório, que será comparado com uma probabilidade $\rho_e$ de ser selecionado um alelo do primeiro pai ou do segundo pai. Nesse exemplo, o cromossomo $a$ se refere ao indivíduo elite, o cromossomo $b$ ao não elite e o valor de $\rho_e = 0.7$. Se o número aleatório, relacionado a cada posição do cromossomo, for menor ou igual a 0.7, o novo filho herdará o alelo do pai $a$, caso contrário, herdará o alelo do pai $b$. É importante destacar que os operadores de variação (mutação e cruzamento) são aplicados no vetor de chaves aleatórias e não no vetor decodificado (MAINIERI, 2014; MOURA, 2018; MORO, 2017; JUNIOR, 2017; GONÇALVES; RESENDE, 2011). Um pseudocódigo do funcionamento geral do BRKGA pode ser visto no Algoritmo 1.

Figura 6 – Cruzamento Uniforme Parametrizado.




Fonte: Adaptado de Gonçalves e Resende (2011)

Em um algoritmo genético de chave aleatória viciada é necessário definir alguns parâmetros. Esses parâmetros são o número de genes em um cromossomo ($N$), o tamanho

**Algoritmo 1:** Pseudocódigo do *Brkga*

```
   Entrada: P, P_e, P_m, ρ_e, critério_de_parada
   Saída: Melhor Indivíduo S
 1 início
 2     Inicializa a População
 3     Avalia a População
 4     Ordena a População
 5     para i ← 1 até critério_de_parada faça
 6         Particione a População em dois Conjuntos: Elite (P_e) e Não-Elite (P - P_e)
 7         Copie os Indivíduos do Conjunto Elite para a Próxima Geração
 8         Gere o Conjunto de Mutantes P_m
 9         para i ← 1 até P - P_e - P_m faça
10             Selecione um Indivíduo do Conjunto Elite;
11             Selecione um Indivíduo do Conjunto Não-Elite;
12             Realize o Cruzamento Uniforme Parametrizado;
13         fim
14         Atualiza a População
15         Avalia a População
16         Ordena a População
17     fim
18 fim
```

da população ($P$), o tamanho da população elite ($P_e$), o tamanho da população mutante ($P_m$), a probabilidade de herança de alelos de um indivíduo elite ($\rho_e$) e o critério de parada (tempo, número de gerações, qualidade da solução ou número de gerações sem melhoria). Na Tabela 3 são apresentados valores para os parâmetros recomendados por Gonçalves e Resende (2011).

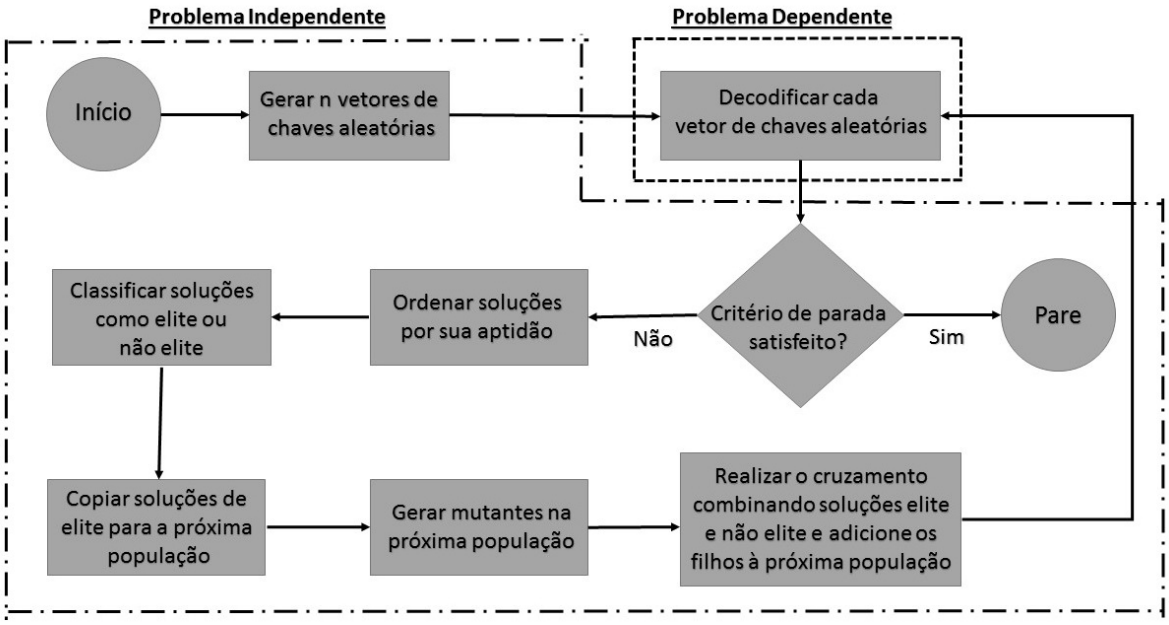
Tabela 3 – Configurações Recomendadas para os Valores dos Parâmetros

| **Descrição** | **Valor Recomendado** |
|---|---|
| Tamanho da população ($P$) | $P = aN$, onde $1 \leq a \in R$ é uma constante e $N$ é o comprimento do cromossomo |
| Tamanho da população elite ($P_e$) | $0.10P \leq P_e \leq 0.25P$ |
| Tamanho da população mutante ($P_m$) | $0.10P \leq P_m \leq 0.30P$ |
| Probabilidade de herança de alelos de elite ($\rho_e$) | $0.5 < \rho_e \leq 0.8$ |

Um algoritmo BRKGA é baseado em uma estrutura de metaheurística de uso geral. Esta estrutura, descrito na Figura 7, é dividida em duas partes: a parte independente do problema e a parte dependente do problema. A parte independente do problema não tem conhecimento do problema que está sendo resolvido. A conexão do BRKGA com o problema de otimização combinatória é feita pela parte dependente do algoritmo, na qual um decodificador produz as soluções a partir dos vetores de chaves aleatórias e calcula a aptidão destas soluções. Diante disso, para especificar um heurística BRKGA, é necessário apenas definir sua representação cromossômica e um decodificador (GONÇALVES; RESENDE, 2011).

A principal vantagem em usar um Algoritmo Genético de Chaves Aleatórias Viciadas está na possibilidade de reuso do código em implementações futuras. Como mencionado anteriormente, o BRKGA tem módulos independentes e dependentes do

Figura 7 – Fluxograma de um BRKGA




Fonte: Adaptado de Gonçalves e Resende (2011)

problema. Dessa forma, ao projetar uma nova heurística para um problema específico, é necessário implementar apenas a parte dependente do problema, ou seja, o decodificador (GONÇALVES; RESENDE, 2011). Esse fato torna o algoritmo flexível de adequação a diversos problemas de otimização. Algumas de suas desvantagens estão na quantidade de parâmetros a serem configurados e, por se tratar de um método populacional, exige um custo computacional alto, sendo mais recomendado para problemas mais complexos e multimodais (MOURA, 2018).

Há na literatura uma variedade de trabalhos que utilizaram a metaheurística BRKGA para resolver problemas complexos, alguns deles podem ser encontrados em Moura (2018), que tratou do problema de otimização global com restrições de caixa; Pereira (2018) abordou o problema de atribuição de localidades a anéis SONET/SDH; Moro (2017) utilizou um GRASP e um BRKGA para resolver o problema de diversidade máxima; Junior (2017) utilizou um BRKGA para resolver o problema de posicionamento de figuras irregulares; Mainieri (2014) utilizou a metaheurística BRKGA para resolver o problema de programação de tarefas no ambiente *flowshop* híbrido, entre outros.

## 4.2 *Simulated Annealing*

O Recozimento Simulado (SA, do inglês *Simulated Annealing*) é um algoritmo de busca local que trabalhar em um único estado atual e tenta melhora-ló movimentando-se para estados vizinhos. A estratégia de busca do SA é inspirada no processo de metalurgia que aquece um material e resfria de modo controlado visando diminuir seus defeitos. O SA

possui cinco parâmetros principais: a solução inicial do problema (S), a temperatura inicial ($T_0$), a temperatura final ($T_f$), a taxa de resfriamento ($\beta$) e o número de soluções vizinhas ($N_v$) que deverão ser geradas a cada iteração (ROCHA, 2013; MONTEIRO et al., 2017).

O SA parte de uma temperatura inicial que é resfriada até chegar a temperatura final. Em cada temperatura, são gerados $N_v$ vizinhos. Para cada vizinho gerado, é calculado o valor da função objetivo. Se o vizinho gerado for melhor que a solução atual, esta é atualizada. Se o vizinho for pior, ele é aceito com uma probabilidade igual a $P = e^{-\triangle f/T}$, em que $\triangle f$ é a diferença de valor da função objetivo do vizinho e da solução atual, e T é a temperatura atual. Quanto maior for o valor de $\triangle f$ e menor a temperatura T, menor sera a chance de aceitar a solução vizinha. Após a geração dos vizinhos, recalcula-se a temperatura atual aplicando a taxa de resfriamento ($\beta$) através da fórmula T = T * $\beta$. O algoritmo termina quando a temperatura atual é menor ou igual a temperatura final. O comportamento típico do algoritmo é aceitar mais soluções piores quando a temperatura está alta, obtendo grande diversificação. A medida que ela diminui, uma quantidade menor de soluções piores são aceitas, e como consequência, uma determinada região de busca é intensificada (ROCHA, 2013; MONTEIRO et al., 2017). O Algoritmo 2 apresenta o pseudocódigo do SA para a busca local. Esse pseudocódigo foi apresentado nos trabalhos de Segatto (2017), Monteiro et al. (2017) e Rocha (2013).

**Algoritmo 2:** Pseudocódigo do *Simulated Annealing*

```
   Entrada: Solução S, T_0, T_f, β, N_v
   Saída: Solução S_Melhor
 1 início
 2     T ← T_0
 3     S_Atual ← S
 4     S_Melhor ← S
 5     enquanto T > T_f faça
 6         para i ← 1 até N_v faça
 7             S_Vizinho ← GeraVizinho(S_Atual)
 8             △f ← f(S_Vizinho) - f(S_Atual)
 9             se △f <= 0 então
10                 S_Atual ← S_Vizinho
11                 se f(S_Vizinho) < f(S_Melhor) então
12                     S_Melhor ← S_Vizinho
13                 fim
14             fim
15             senão
16                 Gere um número aleatório p ∈ (0, 1]
17                 se p < e^{-△f/T} então
18                     S_Atual ← S_Vizinho
19                 fim
20             fim
21         fim
22         T ← T * β
23     fim
24 fim
```

## 4.3 *Movimentos/Vizinhanças da Busca Local*

Os algoritmos de busca local exploram a vizinhança de uma determinada solução na tentativa de gerar soluções melhores. Diante disso, é necessário definir quais movimentos devem ser realizados para explorar a vizinhança de uma solução inicial, na tentativa de escapar de ótimos locais. Neste trabalho, foram realizados testes computacionais com seis movimentos distintos, entre eles: *Move*, *Swap*, *Time Move*, *Room Move*, Cadeia de Kempe Estendido e Cadeia de Kempe Estendido com Restrição de Sala. A descrição de cada um deles é apresentado a seguir:

***Move***: Uma aula é movida para uma posição vazia na tabela-horário. A aula e a posição vazia são escolhidas aleatoriamente. Por exemplo, uma aula X alocada na célula (r1, p1) será movida para uma célula vazia. Logo, a aula pode ser movida para uma outra sala no mesmo período (r2, p1) ou para outro período na mesma sala (r1, p2) ou para um período e sala distintos (r2, p4). O movimento só é considerado válido se não violar restrições rígidas. A Figura 8 exemplifica esse processo.

Figura 8 – Movimento do tipo *Move*

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$AT_2$** | $TC_2$ | | | |
| **rB** | | $GT_2$ | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | | $TC_2$ | | | |
| **rB** | | $GT_2$ | $AT_3$ | | | **$AT_2$** |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

***Swap***: Duas aulas trocam de posição na tabela-horário. As aulas são escolhidas aleatoriamente, desde que, pertençam a disciplinas diferentes. Desse modo, uma aula X alocada na célula (r1, p1) pode ser trocada com uma aula Y alocada em um período e sala distintos (r2, p3) ou com uma aula Y alocada na mesma sala em outro período (r1, p3) ou com uma aula Y alocada no mesmo período em outra sala (r3, p1). Assim como no *Move*, o movimento só é válido quando não inviabiliza a solução. A Figura 9 ilustra a realização de um *Swap*.

Figura 9 – Movimento do tipo *Swap*

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$AT_2$** | $TC_2$ | | | |
| **rB** | | **$GT_2$** | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$GT_2$** | $TC_2$ | | | |
| **rB** | | **$AT_2$** | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

***Time Move***: Uma aula é movida para outro período sem alterar a sala. A aula e a posição da tabela horário são escolhidos aleatoriamente. Logo, uma aula X alocada na célula (r1, p1) pode ser movida para uma posição vazia da tabela horário (r1, p3) ou pode ser trocada com uma aula Y alocada em outro período na mesma sala (r1, p4). Nesse movimento, é realizado um *Move* caso a posição escolhida esteja vazia, caso contrário, é realizado um *Swap*. A Figura 10 apresenta um exemplo do movimento *Time Move*.

Figura 10 – Movimento do tipo *Time Move*

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$AT_2$** | $TC_2$ | | | |
| **rB** | | $GT_2$ | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | | $TC_2$ | | | **$AT_2$** |
| **rB** | | $GT_2$ | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

***Room Move***: Uma aula é movida para outra sala sem alterar o período. Logo, uma aula X alocada na célula (r2, p2) pode ser movida para uma posição vazia da tabela horário (r3, p2) ou pode ser trocada com uma aula Y alocada no mesmo período em outra sala (r4, p2). Assim como nos demais movimentos, a aula e a posição são selecionados de modo aleatório. Caso a posição esteja vazia é realizado um *Move*, caso contrário, um *Swap*. A Figura 11 apresenta um exemplo desse movimento.

Figura 11 – Movimento do tipo *Room Move*

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | $AT_2$ | **$TC_2$** | | | |
| **rB** | | $GT_2$ | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | | |

(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | $AT_2$ | | | | |
| **rB** | | $GT_2$ | $AT_3$ | | | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | | |
| **rD** | $GT_1$ | $AV_1$ | **$TC_2$** | | | |
| **rE** | | $BP_1$ | | | | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

***Cadeia de Kempe***: Uma Cadeia de Kempe é definida como um conjunto de aulas que formam uma componente conexa devido aos conflitos no subconjunto de aulas que pertencem a períodos distintos. As Cadeias de Kempe são derivadas do modelo de coloração de grafo e correspondem a subgrafos conectados contento vértices (aulas) em duas classes de cores (períodos). Para o CB-CTP, o movimento de cadeia de kempe realiza a troca de períodos entre um conjunto de aulas de forma a manter a viabilidade da solução (SEGATTO, 2017).

Considere que foram selecionados dois períodos distintos aleatoriamente: P1 (Dia 0 e H1) e P4 (Dia 2 e H4), ilustrado na Figura 12. O período P1 possui cinco aulas alocadas ($AT_2$, $GT_2$, $TC_1$, $AV_1$, $BP_1$) em cinco salas diferentes (rA, rB, rC, rD e rE). Enquanto,

o período P4 possui quatro aulas alocadas ($CT_1$, $AT_4$, $TC_3$ e $BP_2$) e um período vazio. Suponha que existem conflitos entre as aulas das disciplinas GT e AT, entre as aulas das disciplinas GT e CT e entre as aulas das disciplinas AV e BP. A Figura 13 apresenta o grafo de conflitos formado para o exemplo da Figura 12. Além disso, todas as aulas da mesma disciplina sempre recebem uma aresta, pois possuem alunos e professores em comum. Para esse exemplo são formadas três Cadeia de Kempe: a primeira contém as aulas $AT_2$, $GT_2$, $CT_1$ e $AT_4$, a segunda contém as aulas $TC_1$ e $TC_2$ e a terceira contém as aulas $AV_1$, $BP_1$ e $BP_2$.

Figura 12 – Movimento do tipo *Cadeia de Kempe*

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$AT_2$** | $TC_2$ | | **$CT_1$** | |
| **rB** | | **$GT_2$** | $AT_3$ | | **$AT_4$** | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | $TC_3$ | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | $BP_2$ | |

(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | **$CT_1$** | $TC_2$ | | **$AT_2$** | |
| **rB** | | **$AT_4$** | $AT_3$ | | **$GT_2$** | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | $TC_3$ | |
| **rD** | $GT_1$ | $AV_1$ | | | | |
| **rE** | | $BP_1$ | | | $BP_2$ | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

Figura 13 – Grafo para o Movimento de Cadeia de Kempe


(a) Grafo de conflito antes do movimento

(b) Grafo de conflito depois do movimento

Fonte: Segatto (2017)

O movimento é realizado selecionando a maior Cadeia de Kempe, ou seja, a cadeia que contém as aulas $AT_2$, $GT_2$, $CT_1$ e $AT_4$. Logo, as aulas do período P1 são trocadas com as aulas do período P4, e vice-versa. Caso a viabilidade seja mantida, obtém-se um novo vizinho conforme ilustrado na parte b da Figura 12.

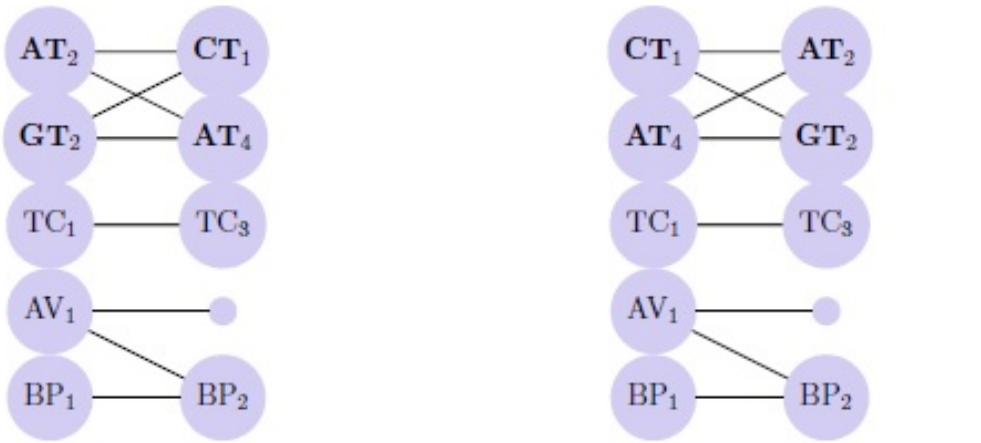
Nesse exemplo, o número de aulas trocadas do período P1 é igual ao número de aulas do período P2, assim como, o número de salas disponíveis em ambos os períodos. No entanto, em alguns casos o número de aulas a serem realocadas em um determinado período ultrapassa o número de salas disponíveis, tornando a troca das aulas inviável. Para

contornar essa situação, foi utilizado neste trabalho a Cadeia de Kempe Estendida com Restrição de Sala apresentada no trabalho de Segatto (2017).

A Cadeia de Kempe Estendida com Restrição de Sala é similar ao processo mencionado. A principal diferença está na construção do grafo e na opção de escolher, caso necessário, mais de um Cadeia de Kempe. A construção do grafo de conflitos é realizado adicionado arestas entre aulas de disciplinas que pertencem ao mesmo currículo ou professor e também adicionando arestas entre salas com períodos iguais (SEGATTO, 2017).

O grafo da Figura 14 ilustra esse processo. Nesse exemplo, três Cadeias de Kempe foram formadas: ($TC_1$, $TC_2$), ($AT_2$, $GT_2$, $CT_1$, $AT_4$) e ($AV_1$, $BP_1$, SalaVazia, $BP_2$). Observa-se que a adição de arestas entre salas com períodos iguais afeta o número de aulas trocadas de um período para outro. Suponha que a cadeia selecionada é a ($AV_1$, $BP_1$, SalaVazia, $BP_2$). Logo, o número de aulas movidas do primeiro período para o segundo será diferente, e vice-versa, pois existe a possibilidade de formar cadeias de kempe selecionando salas com períodos vazios.

Figura 14 – Grafo para o Movimento de Cadeia de Kempe com Restrição de Sala




(a) Grafo de conflito antes do movimento (b) Grafo de conflito depois do movimento

Fonte: Segatto (2017)

Por outro lado, a opção de escolher mais de uma Cadeia de Kempe é realizada para equilibrar o número de aulas com o número de salas disponíveis. O grafo da Figura 16 foi construído de acordo com a Figura 15. Nesse exemplo, foram formadas três cadeias de kempe: ($AT_2$, $GT_2$, $CT_1$, $AT_4$, $GT_4$), ($TC_1$, $TC_3$) e ($AV_1$, $BP_1$, $BP_2$). A cadeia selecionada para o movimento é sempre a maior cadeia formada ($AT_2$, $GT_2$, $CT_1$, $AT_2$, $GT_4$). Como o número de aulas do período P4 é maior que o número de aulas do período P1, o algoritmo seleciona uma segunda Cadeia de Kempe para equilibrar o número de aulas com o número de salas disponíveis em ambos os períodos. Desse modo, a cadeia que contém as aulas ($AV_1$, $BP_1$, $BP_2$) é selecionada e o movimento é realizado (ilustrado na parte b da Figura 16). Se caso as duas cadeias escolhidas não equilibrassem o número de aulas e salas, uma

outra cadeia é escolhida até que forme uma movimento válido ou até que não haja mais cadeias para serem selecionadas (SEGATTO, 2017).

Figura 15 – Movimento de Cadeia de Kempe Estendido

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | $AT_2$ | $TC_2$ | | $CT_1$ | |
| **rB** | | $GT_2$ | $AT_3$ | | $AT_4$ | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | $TC_3$ | |
| **rD** | $GT_1$ | $AV_1$ | | | $GT_4$ | |
| **rE** | | $BP_1$ | | | $BP_2$ | |

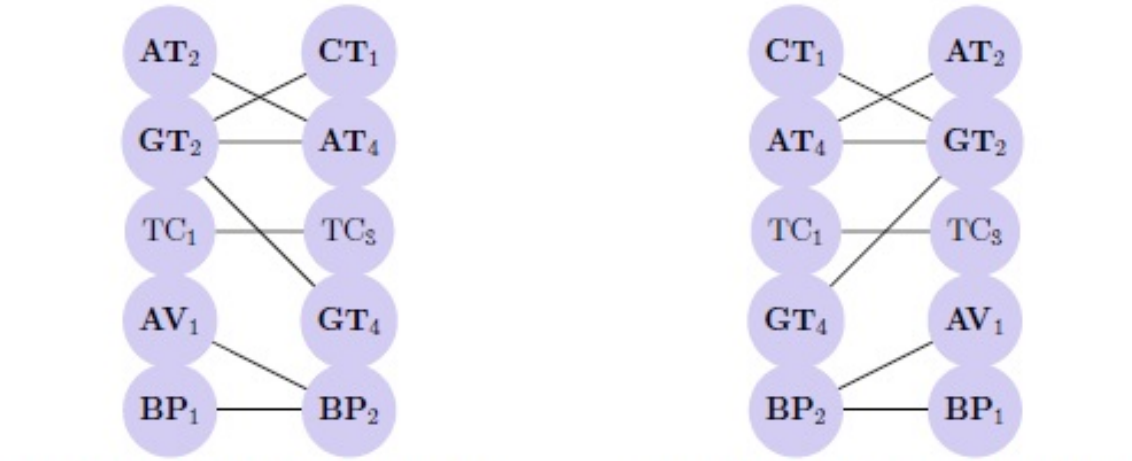
(a) Tabela-horário antes do movimento

| Sala | Dia 0 | | Dia 1 | | Dia 2 | |
|---|---|---|---|---|---|---|
| | $h_0$ | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ |
| **rA** | | $CT_1$ | $TC_2$ | | $AT_2$ | |
| **rB** | | $AT_4$ | $AT_3$ | | $GT_2$ | |
| **rC** | $AT_1$ | $TC_1$ | | $GT_3$ | $TC_3$ | |
| **rD** | $GT_1$ | $GT_4$ | | | $AV_1$ | |
| **rE** | | $BP_2$ | | | $BP_1$ | |

(b) Tabela-horário depois do movimento

Fonte: Segatto (2017)

Figura 16 – Grafo para o Movimento de Cadeia de Kempe Estendido




(a) Grafo de conflito antes do movimento

(b) Grafo de conflito depois do movimento

Fonte: Segatto (2017)

# 5 Abordagem Proposta

A concepção de uma solução para o CB-CTP ocorre em duas fases: construção e melhoria. Na fase de construção, uma tabela horário é criada iterativamente sem violar quaisquer restrições rígidas, mas sendo capaz de violar muitas restrições flexíveis. Na fase de melhoria, o algoritmo recebe uma solução inicial como entrada e tenta melhorá-la gradualmente, minimizando suas restrições flexíveis. Essas etapas são importantes, pois podem alterar a velocidade de convergência do algoritmo e também a qualidade da solução final (WAHID; HUSSIN, 2016). As seções seguintes definem detalhes da modelagem do problema, o algoritmo de construção de uma solução inicial, o algoritmo híbrido proposto e a configuração dos parâmetros das metaheurísticas.

## 5.1 Detalhes da Implementação

A modelagem descrita neste trabalho foi baseado no trabalho de Segatto (2017). As principais classes da implementação do algoritmo são descritas a seguir.

**Instance**: é a classe responsável por armazenar todas as informações do arquivo de entrada, ou seja, de cada instância do problema. Além disso, essa classe armazena uma lista de períodos, uma lista de professores, uma lista de salas, uma lista de disciplinas, uma lista de currículos, uma lista de restrições e uma Matriz Disciplina_Curriculo. Todas essas listas armazenam objetos de uma classe, com exceção da Matriz Disciplina_Curriculo que armazena números inteiros. A Figura 17 apresenta a modelagem dessas classes em um diagrama de classes.

Um período é composto por três inteiros: dia, período do dia e um id do período. Um professor possui um id e um nome. Uma sala possui um id, um nome e uma capacidade. Uma aula possui um id e uma disciplina. Uma disciplina possui um id, um nome, um professor, um número de aulas, um número mínimo de dias em que se deseja que as aulas sejam alocadas, um número de estudantes matriculados, uma lista de aulas, uma lista de períodos viáveis para a alocação das aulas, uma lista com os currículos e uma Matriz de Indisponibilidade ($N_{disciplinas} \times N_{periodos}$) que informa os horários indisponíveis para as disciplinas. Um currículo possui um id, um nome e uma lista de disciplinas. Cada restrição possui um peso e é utilizada para o cálculo da função objetivo. A Matriz Disciplina_Curriculo possui dimensão $N_{disciplinas} \times N_{curriculos}$ e é utilizada para verificar conflitos, ou seja, para verificar se duas disciplinas pertencem ao mesmo currículo.

Neste trabalho, a representação de um indivíduo foi adaptado para tratar o problema da melhor forma possível. Desse modo, cada indivíduo é representado como um objeto

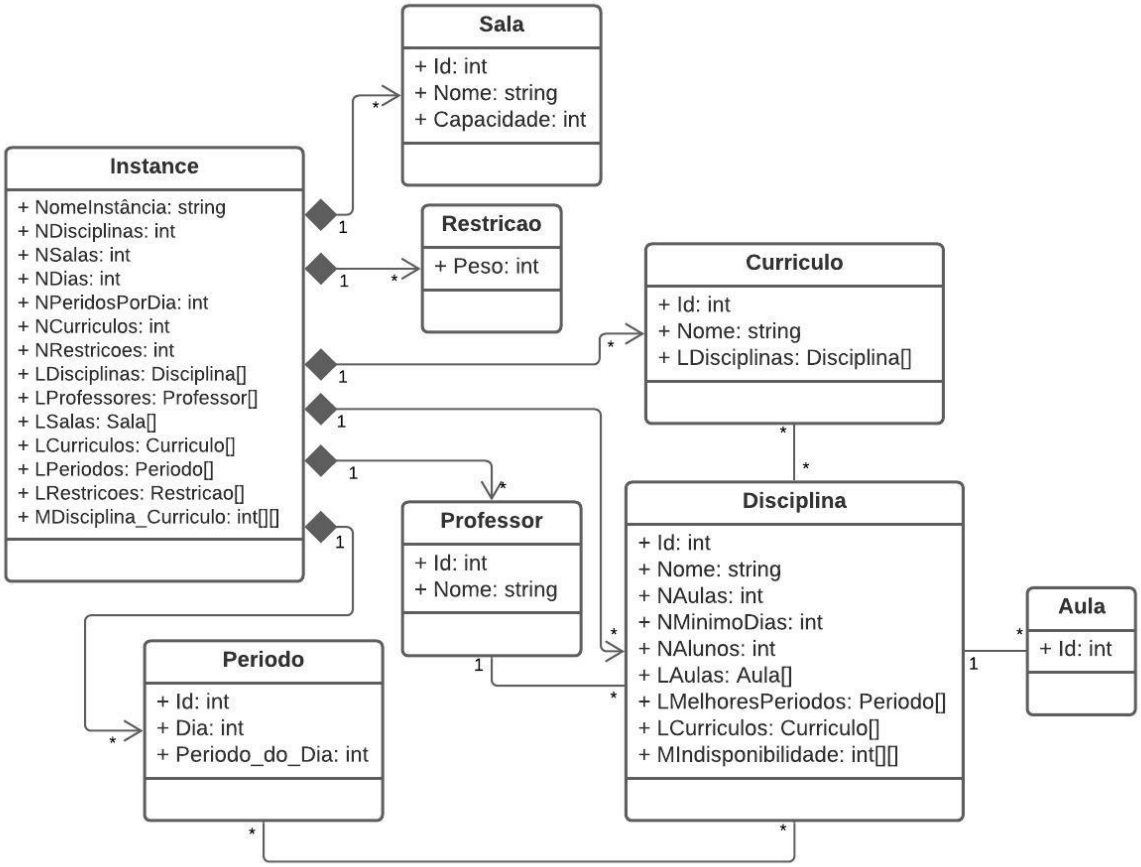



Figura 17 – Classes do Problema

da classe **Indivíduo**. A Figura 18 ilustra a composição de um indivíduo da população. Um indivíduo possui um vetor de chaves aleatórias, uma lista de períodos decodificado, um vetor que representa a Tabela Horário (Todos_Horários), um vetor que contém as aulas alocadas (Aulas_Alocadas) e um vetor que armazena os horários desocupados (Horários_Vazios). O tamanho do vetor de chaves aleatórias é definido de acordo com o número de períodos disponíveis. O tamanho dos vetores Todos_Horários, Aulas_Alocadas e Horários_Vazios é igual a $N_{salas} \times N_{periodos}$. Cada posição desses vetores é ocupada com um objeto da classe **Alocação** que contém um sala, uma aula e um período. Inicialmente, cada aula da alocação possui valor nulo.

Além desses vetores, um indivíduo possui matrizes pré-processadas. Essa estratégia foi utilizado por Segatto (2017) baseado na representação adotada por Teoh, Abdullah e Haron (2015). As matrizes pré-processadas armazenam informações da solução do problema, facilitando de maneira eficiente o acesso aos dados necessários para efetuar mudanças, calcular ou recalcular a função objetivo. Essas matrizes são inicializadas na criação de cada indivíduo e atualizadas em cada movimento durante a busca local. Segundo Segatto (2017), essas matrizes reduzem o tempo de processamento na atualização de mudanças,

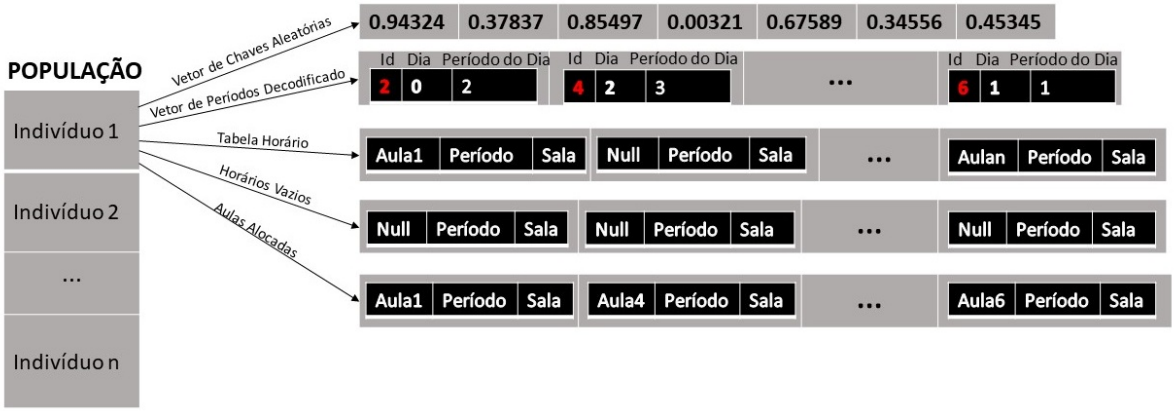



Figura 18 – Representação dos Indivíduos

pois o custo para atualizar uma solução e recalcular a função objetivo é bem menor quando comparado com a inexistência delas. Essas matrizes são descritas a seguir:

- **Salas Utilizadas**: Matriz Disciplina × Sala. É uma matriz de inteiros utilizada no cálculo da violação da restrição fraca Estabilidade da Sala.
- **Dias Utilizados**: Matriz Disciplina × Dia. É uma matriz de inteiros utilizada no cálculo da violação da restrição fraca Dias Mínimos de Trabalho.
- **Horários utilizados por Currículo**: Matriz Currículo × Período. É uma matriz de inteiros bidimensional utilizada na verificação da restrição forte Conflitos.
- **Alocação Currículo de Dias e Períodos**: Matriz Currículo × Dia × Período. É uma matriz de objetos tridimensional utilizada no cálculo da violação da restrição fraca Compacidade do Currículo.
- **Horários utilizados por Professores e Horário dos Professores**: A primeira é uma matriz de inteiros bidimensional Professor × Período e a segunda é uma matriz de objetos bidimensional Professor × Período. Ambas são também utilizadas na verificação da restrição forte Conflitos.

## 5.2 Solução Inicial

A construção de uma população de soluções iniciais no BRKGA é realizada através de uma heurística gulosa (para produzir boas soluções) e aleatória (para produzir soluções diferentes a cada execução). A heurística construtiva tem a capacidade de gerar uma população de soluções inicias (ou vários cronogramas de horários) que satisfaz todas as restrições rígidas. Caso alguma solução gerada viole alguma restrição rígida, ela é descartada da população.

Partindo de uma tabela horário vazia, uma aula é inserida por vez em um período e em uma sala viável para alocação. Para inserir uma aula no cronograma, duas etapas são realizadas: a primeira consiste em selecionar uma aula não atribuída de uma disciplina e a segunda consiste em selecionar um período viável para a aula. As etapas do processo de construção da população de soluções, assim como, o processo para selecionar uma aula, um período e uma sala são descritas a seguir:

**Inicializar**: Cria uma população $p$ de indivíduos. Os vetores e as matrizes são inicializadas, com exceção da lista de períodos decodificado que permanece vazia na inicialização dos indivíduos da população.

**Decodificar**: O decodificador recebe como entrada um vetor de chaves aleatórias, uma lista de períodos e retorna para cada indivíduo uma lista de períodos decodificado. Cada posição do primeiro vetor de chaves aleatórias está associada com a mesma posição da lista de períodos. O processo de decodificação utilizado na abordagem proposta é o mesmo apresentado por Bean (1994), ou seja, o vetor de chaves aleatórias é ordenado gerando uma sequência. Logo, a lista de períodos decodificado é uma lista que segue essa sequência. A Figura 19 ilustra o processo de decodificação utilizado neste trabalho. O processo de decodificação é utilizado para definir a ordem sequencial dos períodos em que as aulas serão alocadas.

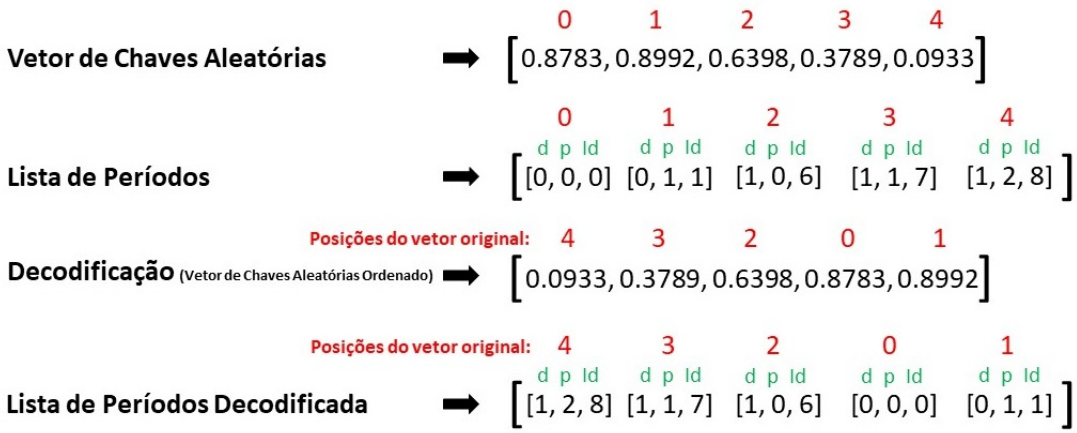



Figura 19 – Codificação e Decodificação da Lista de Períodos

Em outras palavras, o processo de decodificação consiste em gerar uma permutação da lista de períodos. Nesse ponto, poderia ser utilizado o método *shuffle*(), presente em várias linguagens de programação, para reorganizar a ordem dos períodos na lista. No entanto, qualquer decodificador implementado para o BRKGA é utilizado para decodificar os vetores de chaves aleatórias em todas as fases do algoritmo como descrito na Figura 7 da seção 4.1 do Capítulo 4. Portanto, é conveniente que o mesmo seja utilizado também na concepção das soluções inicias.

**Construir Solução Inicial**: O algoritmo 3 apresenta o processo de construção da solução inicial. Inicialmente, a lista de períodos decodificada é ordenada em ordem crescente de acordo com o número do intervalo de tempo (ou período do dia). A Figura 20 ilustra esse processo. Essa lista decodificada e ordenada é utilizada para preencher a lista de períodos viáveis de cada disciplina e assim definir o número de períodos viáveis.

**Algoritmo 3:** Pseudocódigo do Algoritmo de Construção

```
   Entrada: TodasAulas, Salas, Indivíduo
   Saída: Solução S
 1 início
 2     Preenche a Lista de Períodos Viáveis de cada Disciplina
 3     Lista de Candidatos ← TodasAulas
 4     Lista de Salas ← Salas
 5     Ordena a Lista de Candidatos em Ordem Crescente
 6     Ordena a Lista de Salas em Ordem Crescente
 7     enquanto Lista de Candidatos > 0 faça
 8         se Grau de Saturação > 0 então
 9             Aloca Aula e Atualiza a Lista de Períodos Viáveis de cada Disciplina
10             se Não Existe Salas com Capacidade >= Número de Alunos então
11                 Lista de Aulas Não Atribuídas ← Aula
12             fim
13         fim
14         senão
15             Remove uma Aula Alocada e Aloca a Aula com Grau de Saturação igual a Zero
16             se Não Existe Aula Alocada que Possa ser Removida então
17                 Excluí Lista de Candidatos e Indivíduo
18                 Cria um Novo Indivíduo e Reinicializa a Construção da Solução
19             fim
20         fim
21         Remove Aula da Lista de Candidatos
22         Ordena a Lista de Candidatos
23     fim
24     Ordena a Lista de Aulas Não Atribuídas em Ordem Crescente
25     Ordena a Lista de Salas em Ordem Decrescente
26     enquanto Lista de Aulas Não Atribuídas > 0 faça
27         Aloca Aula e Atualiza a Lista de Períodos Viáveis de cada Disciplina
28         Remover Aula da Lista de Aulas Não Atribuídas
29         Ordenar a Lista de Aulas Não Atribuídas
30     fim
31 fim
```

Para dá início ao processo de construção da tabela horário é criada uma **Lista de Candidatos** que contém as aulas de todas as disciplinas. A ordem sequencial das aulas é definida utilizando a abordagem da heurística Grau de Saturação: os eventos que tem menos períodos viáveis são agendados primeiro.

A atribuição das aulas a períodos e salas é realizada em duas fases. Na Fase 1, a lista de salas é ordenada em ordem crescente de acordo com a capacidade. Em cada iteração, a aula com menos períodos disponíveis é atribuída ao primeiro período da lista de períodos viáveis e na primeira sala com capacidade igual ou superior ao número de alunos. Esse processo ocorre até que a tabela horário esteja completamente preenchida.

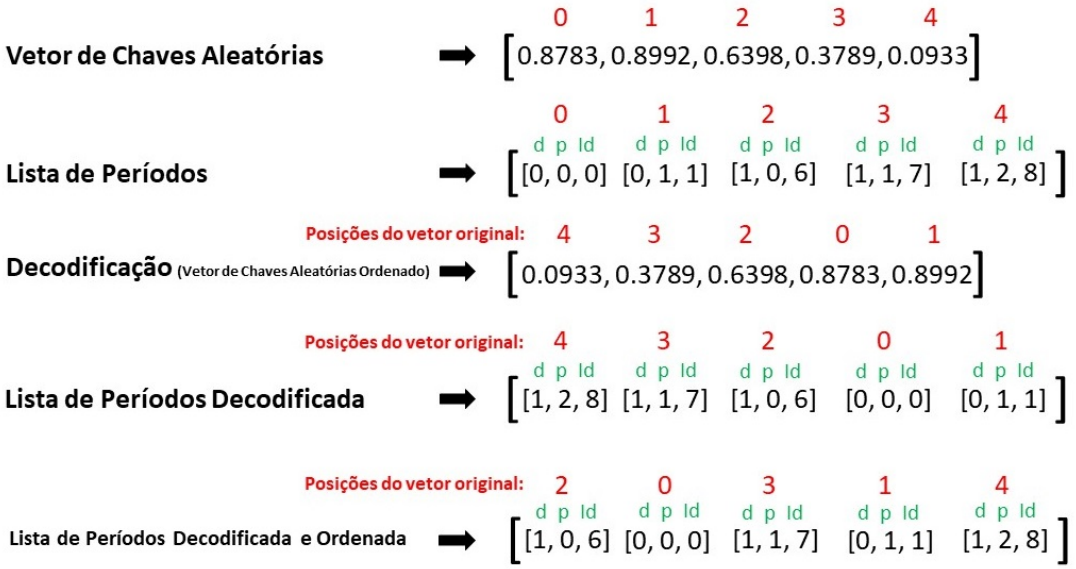



Figura 20 – Ordenação da Lista de Períodos

Quando uma aula é alocada, o período é removido da lista de períodos viáveis de todas as aulas conflitantes e a aula é removida da lista de candidatos. Após cada alocação, o grau de saturação de cada aula é atualizado e a lista de candidatos é reordenada. Nesse ponto, os vetores e as matrizes de cada indivíduo da população são atualizados. O vetor que representa a Tabela Horário e o vetor de Aulas_Alocadas recebe um objeto da classe Alocação com a respectiva aula, período e sala. Por outro lado, os períodos e salas utilizados são retirados do vetor de Horários_Vazios.

Em alguns casos, é possível que uma determinada aula não tenha períodos disponíveis (Aula A) em um ponto do processo de construção da solução. Para contornar essa situação foi implementado um procedimento para remover da tabela horário uma aula alocada (Aula B). Para remover uma aula da tabela horário, é necessário verificar algumas condições:

A) a aula a ser removida deve pertencer ao mesmo professor ou currículo da aula que não pôde ser alocada (aula conflitante);

B) a aula a ser removida deve possuir outros períodos disponíveis para realocação;

C) a disciplina da aula a ser removida deve ser diferente da aula que não pôde ser alocada;

D) o período e a sala em que a aula a ser removida está alocada devem ser viáveis para a aula que não pôde ser alocada;

E) verificar se alguma outra aula conflitante com a aula que não possui períodos disponíveis está alocada no mesmo período da aula a ser removida. Caso exista e

caso as quatro condições acima seja satisfeitas, remova da tabela horário também essa outra aula, desde que, a mesma possua outros períodos para realocação.

A primeira aula alocada que satisfaça as cinco condições listadas é removida da tabela horário, abrindo espaço para a aula que não pôde ser alocada no ponto atual da construção. A aula removida da tabela horário é adicionada no final da lista de candidatos e a lista é reordenada. No entanto, se caso nenhuma aula for encontrada para contornar o processo de alocação, a lista de candidatos e o indivíduo são excluídos. O processo de atribuição será reiniciado do zero criando um novo indivíduo. Esse processo se repete até que o tamanho definido para a população seja alcançado.

É possível que no final da Fase 1 ainda existam aulas que não foram alocadas a salas e períodos. Nesse caso, elas são colocadas na lista de aulas não atribuídas, sendo necessária a Fase 2. Nesta fase, a lista de salas é classificada em ordem decrescente. Cada aula é atribuída à primeira sala disponível, sem verificar a capacidade da sala para a aula, e também ao primeiro período da lista de períodos viáveis. Assim como na Fase 1, o período é removido da lista de períodos viáveis de todas as disciplinas conflitantes, o grau de saturação é atualizado, os vetores e matrizes de cada indivíduo são atualizadas, a aula alocada é removida da lista de aulas não atribuídas e a lista é novamente ordenada.

O processo de criar uma lista de períodos viáveis para cada disciplina e remover o período que foi utilizado da lista de todas as disciplinas conflitantes garante que cada aula contenha apenas períodos viáveis de alocação em qualquer ponto do processo de construção da solução. Logo, as restrições rígidas Conflitos e Disponibilidade são sempre satisfeitas.

**Avaliar Solução**: Cada uma das soluções construídas é avaliada recebendo um valor de aptidão. Este valor é determinado pela função objetivo, ou seja, pela penalidade atribuída as restrições flexíveis violadas.

## 5.3 Algoritmo Híbrido

O algoritmo 4 apresenta o pseudocódigo para a abordagem proposta. O algoritmo é composto de duas fases: construção e melhoria. Na fase de melhoria foi utilizado o BRKGA, o *Simulated Annealing* para uma busca local e a Cadeia de Kemepe como mecanismo de pertubação.

Após a geração de uma população de soluções inicias, o BRKGA evolui uma população de indivíduos por meio dos operadores de mutação e cruzamento. Inicialmente, a população é dividida em elite e não elite. Os indivíduos mutantes são gerados através do algoritmo de construção, ou seja, inicializa o conjunto de indivíduos $P_m$, decodifica os vetores de chaves aleatórias e utiliza o algoritmo de construção para gerar a solução (ou a

**Algoritmo 4:** Pseudocódigo do *Brkga* com SA e Cadeia de Kempe

```
   Entrada: P, P_e, P_m, ρ_e, critério_de_parada
   Saída: Melhor Solução
 1 início
 2     PopulacaoAtual ← Inicializa a População
 3     Decodifica a População
 4     Construir Solução ⟹ Algoritmo de Construção
 5     Avalia a População
 6     Ordena a População
 7     para i ← 1 até critério_de_parada faça
 8         NovaPopulacao ← Ø
 9         Particione a População em dois Conjuntos: Elite (P_e) e Não-Elite (P - P_e)
10         Gere o Conjunto de Mutantes P_m (Decodifica P_m ⟹ Algoritmo de Construção)
11         NovaPopulação ← P_m
12         para i ← 1 até P - P_e - P_m faça
13             Selecione um Indivíduo do Conjunto Elite
14             Selecione um Indivíduo do Conjunto Não-Elite
15             C ← Realize o Cruzamento Uniforme Parametrizado (I_Elite, I_NaoElite)
16             NovaPopulação ← NovaPopulação ∪ C
17         fim
18         Avalia a População
19         PopulaçãoAtual ← NovaPopulacao ∪ Elite P_e
20         Ordena a População
21         se Critério para Execução da Busca Local é Válido então
22             Melhor Indivíduo da População ← Busca Local
23             Melhor Indivíduo da População ← Pertubação
24             Melhor Indivíduo da População ← Busca Local
25             Melhor Indivíduo da População ← Aceitação
26         fim
27         se Aptidão do Melhor Indivíduo da População < Aptidão da Melhor Solução então
28             Melhor Solução ← Melhor Indivíduo da População
29         fim
30     fim
31 fim
```

tabela horário). Por outro lado, os indivíduos do cruzamento são gerados selecionando um pai do conjunto elite e outro do não elite.

O cruzamento no BRKGA é do tipo uniforme parametrizado e ocorre no vetor de chaves aleatórias, como descrito na seção 4.1. Logo, o cromossomo do novo indivíduo é constituído de características do pai elite e do pai não-elite. Para adequar o cruzamento ao problema abordado, foi aplicado um processo de mapeamento considerando o valor de cada chave aleatória do novo indivíduo. O valor da chave aleatória determina em qual dos pais irá ocorrer o cruzamento mapeado. Se o valor da chave for menor que a probabilidade de cruzamento definida, o cruzamento será realizado no pai elite, caso contrário, no pai não-elite.

Em outras palavras, o processo de cruzamento natural do BRKGA gera um novo indivíduo com um vetor de chaves aleatórias que contém valores do pai elite e do não-elite. Por outro lado, o cruzamento mapeamento utiliza o valor de cada chave do novo indivíduo para realizar mudanças na tabela de horários dos pais selecionados. Esse processo permite

gerar dois filhos com características diferentes. O filho com menor valor de penalidade de restrição é considerado como o novo indivíduo do cruzamento, sendo inserido na população da geração corrente. O cruzamento ocorre da seguinte forma:

1 Selecione aleatoriamente uma posição no cronograma de horários de ambos os pais.

2 Estabeleça uma relação entre as aulas alocadas no mesmo período das posições selecionadas, mapeando-as no local do cruzamento.

3 Troque as aulas especificadas pela relação do mapeamento.

A Figura 21 exemplifica o processo de formação de indivíduos filhos. Observa-se que o valor da primeira chave aleatória do novo indivíduo é menor que a probabilidade de cruzamento. Portanto, o cruzamento mapeado será realizado no pai elite. O processo de cruzamento começa selecionando aleatoriamente duas posições (podem ser iguais ou diferentes) no cronograma de horários. Uma relação é estabelecida entre as aulas nas posições selecionadas, mapeando-as no local do cruzamento.

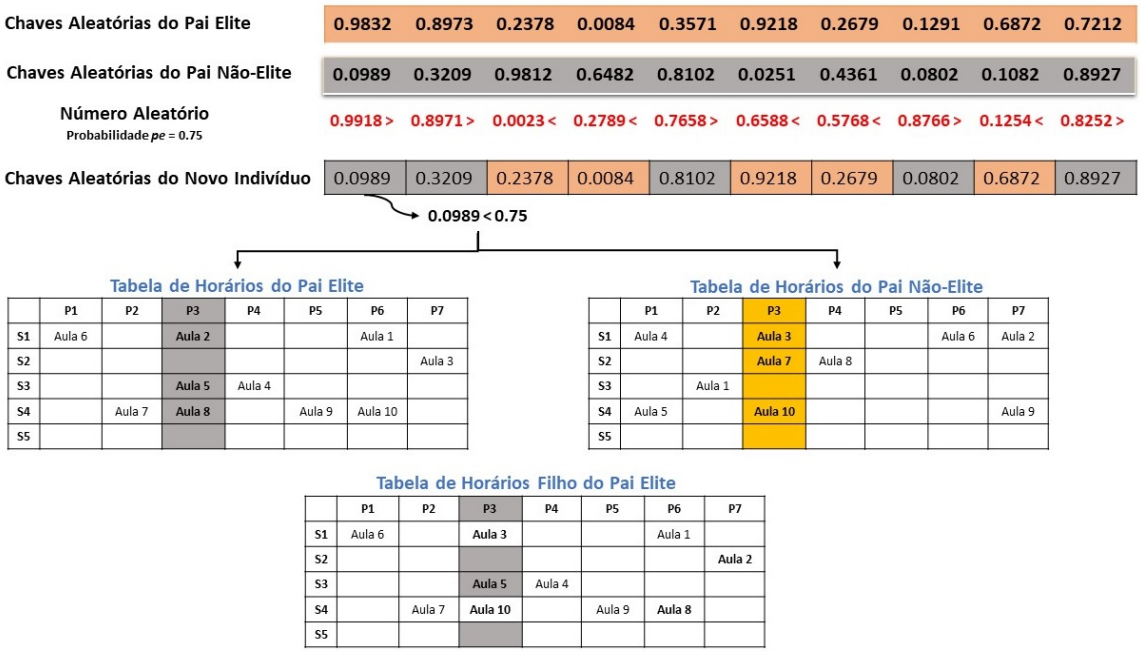



Figura 21 – Cruzamento Mapeado utilizando Chaves Aleatórias

A relação estabelecida entre as posições escolhidas determinada que as aulas: L2-L3 e L8-L10 devem trocar de lugar no cronograma de horários. Desse modo, duas aulas podem trocar de posição, ou seja, de sala/período, de acordo com essa relação, desde que, a viabilidade do cronograma de horários seja mantida. Esse tipo de cruzamento mapeamento foi baseado no trabalho de Akkan e Gülcü (2018), que utilizou um cruzamento mapeado baseado em período. As diferenças do processo utilizado neste trabalho envolve

os seguintes aspectos: 1) utilização das chaves para determinar em qual dos pais irá ocorrer o cruzamento; 2) diferentes períodos podem ser selecionados para estabelecer a relação de mapeamento; 3) a relação de mapeamento só é realizada entre aulas, não foi considerado períodos/salas vazias.

Após a execução dos operadores do BRKGA (particionamento da população, cruzamento e mutação), o melhor indivíduo da população é submetido duas vezes a busca local e a um mecanismo de pertubação. No entanto, isso só ocorre quando um critério é satisfeito. O critério utilizado para execução da busca local e da pertubação foi o tempo de execução. Nesse caso, o tempo de execução do algoritmo precisa ser superior ou igual a 20% do tempo total. Esse critério foi estabelecido na tentativa de equilibrar o tempo para as metaheurísticas utilizadas na abordagem proposta.

Quando o critério é satisfeito, o *Simulated Annealing* recebe como entrada o melhor indivíduo da população corrente e busca melhora-ló por meio de dois movimentos para geração de soluções vizinhas: *Time Move* e *Room Move*. Em seguida, o indivíduo retornado é submetido ao mecanismo de pertubação definido como Cadeia de Kempe. A Cadeia de Kempe é utilizada para explorar novas regiões no espaço de soluções. O indivíduo retornado do novo espaço de busca pode apresentar valor de aptidão superior ou igual ao indivíduo retornado da busca local. Em consequência disso, o indivíduo é submetido novamente a busca local e um critério de aceitação é aplicado. A Figura 22 apresenta o fluxo de execução do algoritmo híbrido.

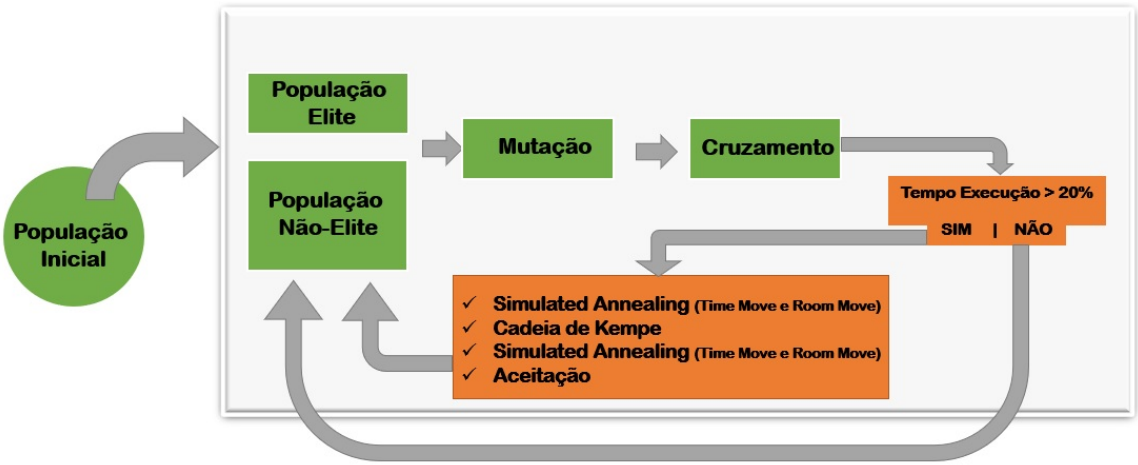



Figura 22 – Fluxo de Execução do Algoritmo Híbrido

O processo de submissão do melhor indivíduo da população a uma busca local e a um mecanismo de pertubação foi inspirado na estrutura da *Iterated Local Search*. O ILS inicia com uma solução qualquer, que em seguida é submetida a uma busca local para obter um solução de melhor qualidade. Na fase iterativa, a solução resultante é submetida a três procedimentos: pertubação, busca local e aceitação. Desse modo, a solução obtida

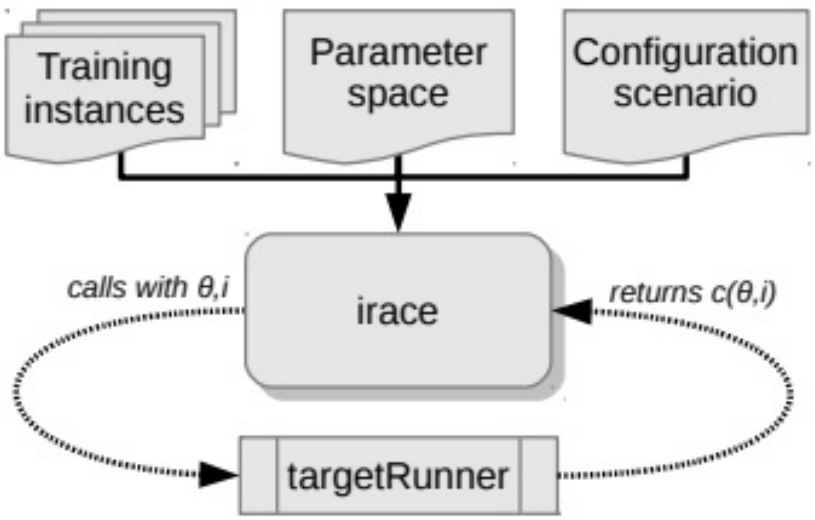
após os procedimentos é selecionada como ponto de partida para a próxima iteração, caso apresente um menor custo de penalidade de restrições.

## 5.4 Ajuste dos Parâmetros

Os algoritmos de otimização são métodos que possuem um grande número de parâmetros que devem ser configurados. Neste trabalho, utilizamos o pacote *irace* para ajustar os parâmetros da abordagem proposta. O *irace* é uma ferramenta para configuração automática de parâmetros de algoritmos de otimização e decisão. Ele implementa o método *Iterated Race*, que é uma extensão do método *Iterated F-race*, para encontrar as configurações mais adequadas para um conjunto de instâncias de um problema (LÓPEZ-IBÁÑEZ et al., 2016).

A Figura 23 apresenta o esquema de funcionamento do *irace*. O *irace* recebe como entrada um arquivo que contém a definição dos parâmetros, um conjunto de instâncias para os quais os parâmetros devem ser ajustados e um arquivo que contém a definição do cenário de configuração do *irace*. Um *targetRunner* (ou programa) atua como uma interface entre a execução do algoritmo destino e o *irace*. Ele recebe a instância e uma configuração específica e retorna a avaliação da execução do algoritmo destino (LÓPEZ-IBÁÑEZ et al., 2016).

Figura 23 – Funcionamento do Irace




Fonte: López-Ibáñez et al. (2016)

Gonçalves e Resende (2011) apresentaram um conjunto de valores recomendados para os parâmetros do BRKGA (descrito na seção 4.1). Portanto, os parâmetros foram ajustados considerando os intervalos recomendados. Os valores dos parâmetros da busca local *Simulated Annealing* foram coletados da literatura, tendo em vista, que os mesmos

valores foram replicados nos trabalhos de Segatto (2017) (GRASP e SA) e Monteiro et al. (2017) (ILS e SA). Esses autores apresentaram abordagens diferentes para resolver as instâncias do ITC-2007, sendo que, cada abordagem incluía o SA na resolução do problema.

Além disso, foi realizado o ajuste da probabilidade de seleção de cada movimento da busca local. Em cada iteração do SA um movimento do tipo *Time Move* ou *Room Move* é selecionado para geração de soluções vizinhas. Diante disso, foi definido vários valores de probabilidade de seleção para os movimentos da busca local. Por exemplo: se um determinado número randômico for maior que 0.20, o movimento a ser realizado é o *Time Move*, caso contrário, *Room Move*. A Tabela 4 apresenta os valores de todos os parâmetros utilizados neste trabalho.

Tabela 4 – Valores dos Parâmetros

| **Parâmetro** | **Descrição** | **Valores** |
|---|---|---|
| P | Tamanho da População (BRKGA) | 1 x Tamanho do Cromossomo |
| $P_e$ | Tamanho da População Elite (BRKGA) | 0.15 |
| $P_m$ | Tamanho da População Mutante (BRKGA) | 0.15 |
| $\rho_e$ | Probabilidade de Cruzamento (BRKGA) | 0.58 |
| $T_0$ | Temperatura Inicial (SA) | 1.5 |
| $T_f$ | Temperatura Final (SA) | 0.005 |
| $\beta$ | Taxa de Resfriamento (SA) | 0.999 |
| $N_v$ | Número de Soluções Vizinhas (SA) | 500 |
| T | Tempo Máximo de Execução | 234 segundos |
| TM | Movimento Time Move | $> 0.50$ |
| RM | Movimento Room Move | $< 0.50$ |

# 6 Resultados

Os testes computacionais foram realizados em uma máquina com processador Intel Core i5 1.60GHz, 8GB de memória RAM e sistema operacional Windows 10. Os algoritmos foram implementados em linguagem C++, utilizando o paradigma orientado a objetos, e compilados com GNU GCC Compiler. Neste trabalho, utilizamos as 21 instâncias do problema de horários de curso baseado em currículo disponibilizado na competição ITC-2007.

Seguindo as diretrizes da competição, foi definido um número máximo de 10 execuções para cada instância. O tempo máximo de execução foi definido pelo programa de *benchmark* disponibilizado pela organização da competição. Para a máquina utilizada neste trabalho, o tempo máximo disponibilizado foi de 234 segundos. Além disso, a execução da abordagem híbrida foi realizada com diferentes sementes para geração de números aleatórios. As seções posteriores apresentam os resultados dos experimentos realizados.

## 6.1 Execução de Experimentos

A Tabela 5 apresenta os resultados obtidos pelo algoritmo de construção para as 21 instâncias do ITC-2007. As instâncias são dividas em três níveis de dificuldade: iniciais, atrasadas e ocultas. Para avaliar o algoritmo foram realizados 50 testes de execução com cada instância e para cada execução foi delimitado uma população de 50 soluções. Logo, o número máximo de soluções viáveis que podem ser geradas para cada instância é igual a 2.500.

Os dados reportados na tabela refere-se a: (Melhor FO) o melhor valor da função objetivo obtido com o algoritmo de construção, (Pior FO) o pior valor da função objetivo, (Tempo Médio) tempo médio em segundos para se obter uma população de soluções viáveis, (Max Soluções) o número máximo de soluções viáveis geradas e (Media Soluções) o número médio de soluções geradas por população. Todos os valores dispostos na tabela são de soluções que não violaram restrições rígidas.

Analisando os dados da Tabela 5, observa-se que no geral, o tempo médio para gerar uma população com no máximo 50 soluções viáveis, para cada instância, é menor que 3 segundos. Observa-se que apenas as instâncias comp07, comp16 e comp20 consomem, em média, mais de 2 segundos para gerar uma população com 50 soluções viáveis. No geral, o número médio de soluções viáveis geradas foi condizente com o valor máximo pré-definido e o tempo médio para as demais instâncias é inferior a 2 segundos.

Não há provas de que a heurística construtiva encontre uma solução viável para

Tabela 5 – Resultados do Algoritmo de Construção para as Instâncias do ITC-2007

| Níveis | Instâncias | Melhor FO | Pior FO | Tempo Médio | Max Soluções | Média Soluções |
|---|---|---|---|---|---|---|
| **Inicias** | comp01 | 227 | 274 | 0.2392 s | 2500 | 50 |
| | comp02 | 537 | 787 | 1.05738 s | 2500 | 50 |
| | comp03 | 478 | 652 | 0.87824 s | 2500 | 50 |
| | comp04 | 479 | 604 | 1.27238 s | 2500 | 50 |
| | comp05 | 942 | 1305 | 0.44256 s | 2500 | 50 |
| | comp06 | 638 | 850 | 1.65082 s | 2500 | 50 |
| | comp07 | 680 | 915 | 2.79442 s | 2500 | 50 |
| **Atrasadas** | comp08 | 528 | 680 | 1.63142 s | 2500 | 50 |
| | comp09 | 586 | 784 | 1.17448 s | 2500 | 50 |
| | comp10 | 586 | 784 | 1.92216 s | 2500 | 50 |
| | comp11 | 53 | 83 | 0.23318 s | 2500 | 50 |
| | comp12 | 1229 | 1673 | 1.11838 s | 2500 | 50 |
| | comp13 | 500 | 668 | 1.49304 | 2500 | 50 |
| | comp14 | 471 | 717 | 1.1207 s | 2500 | 50 |
| **Ocultas** | comp15 | 461 | 656 | 0.88012 s | 2500 | 50 |
| | comp16 | 606 | 798 | 2.12452 s | 2500 | 50 |
| | comp17 | 646 | 817 | 1.5435 s | 2500 | 50 |
| | comp18 | 417 | 530 | 0.32198 s | 2500 | 50 |
| | comp19 | 493 | 643 | 1.0546 s | 2500 | 50 |
| | comp20 | 697 | 904 | 2.18626 s | 2500 | 50 |
| | comp21 | 640 | 849 | 1.5502 s | 2500 | 50 |

uma instância qualquer. No entanto, para todas as instâncias testadas neste trabalho, uma solução viável é facilmente encontrada. Portanto, é verídico que o algoritmo de construção é capaz de gerar um conjunto de soluções inicias viáveis em tempo hábil para o CB-CTP, sem violar restrições rígidas e violando várias restrições flexíveis.

## 6.2 Comparação e Análise dos Resultados

A Tabela 6 ilustra uma comparação das soluções iniciais geradas pelo algoritmo proposto com as soluções produzidas por Teoh, Abdullah e Haron (2015), Wahid e Hussin (2016), Wahid et al. (2019) e Segatto (2017). O objetivo desses trabalhos foi fornecer soluções iniciais viáveis capazes de diminuir o custo de restrições flexíveis para as instâncias do ITC-2007, com exceção de Teoh, Abdullah e Haron (2015) e Segatto (2017), que submeteram as soluções iniciais para a fase de melhoria.

A comparação das soluções iniciais foi realizada considerando apenas o valor da função objetivo para cada instância, pois o tempo de execução não foi disponibilizado nos demais trabalhos. Portanto, essas comparações visaram apontar a capacidade da abordagem proposta em produzir soluções iniciais viáveis para o CB-CTP. Os valores destacados em negrito na Tabela 6 representam os valores com menos penalidades para as restrições flexíveis e, é claro, a abordagem proposta supera todas as outras.

Para analisar o desempenho da abordagem proposta, foram realizados 30 execuções independentes para cada instância utilizando diferentes *seed* gerados aleatoriamente. A Tabela 7 apresenta os resultados computacionais obtidos pelo BRKGA híbrido (incluindo

Tabela 6 – Comparação da Solução Inicial

| **Instâncias** | **Solução Inicial** | | | |
|---|---|---|---|---|
| | Teoh, Abdullah e Haron (2015) | Segatto (2017) | Wahid e Hussin (2016) Wahid et al. (2019) | **Proposta** |
| comp01 | 412 | 623 | 330 | **227** |
| comp02 | 1678 | 1075 | 769 | **537** |
| comp03 | 2146 | 849 | 702 | **478** |
| comp04 | 1678 | 1273 | 694 | **479** |
| comp05 | 2624 | 1189 | 1466 | **942** |
| comp06 | 1493 | 2342 | 947 | **638** |
| comp07 | 1482 | 2615 | 1043 | **680** |
| comp08 | 1582 | 1522 | 790 | **528** |
| comp09 | 1294 | 1021 | 847 | **586** |
| comp10 | 1895 | 1891 | 898 | **586** |
| comp11 | 417 | 396 | 230 | **53** |
| comp12 | 1361 | 1496 | 1498 | **1229** |
| comp13 | 1964 | 1787 | 793 | **500** |
| comp14 | 809 | 1417 | 745 | **471** |
| comp15 | 1929 | 849 | 702 | **461** |
| comp16 | 1167 | 1697 | 949 | **606** |
| comp17 | 1299 | 1836 | 902 | **646** |
| comp18 | 777 | 600 | 583 | **417** |
| comp19 | 1903 | 839 | 637 | **493** |
| comp20 | 1313 | 3353 | 1042 | **697** |
| comp21 | 2288 | 1329 | 928 | **640** |

o valor mínimo de penalidade de restrição (**MIN**), o valor médio de penalidade (**M**) e o desvio padrão (**DP**)) e os resultados de outros trabalhos da literatura. Na Tabela 7 estão dispostos os valores médios da função de penalidade, ou seja, a média dos valores da violação de restrições flexíveis para soluções viáveis. Quanto menor for o valor médio, melhor será a capacidade do algoritmo correspondente.

Os trabalhos da Tabela 7, denominadas de Muller (2009), Lu e Hão (2010) e Abdullah (2012) são os mais conhecidos da literatura. A maioria dos trabalhos analisados compararam os resultados com os trabalhos de Müller (2009), Lü e Hao (2010) e Abdullah et al. (2012). Müller (2009) foi o vencedor da competição ITC-2007, apresentando os melhores resultados para as instâncias do CB-CTP, na época da competição. Os melhores resultados ou as melhores médias para cada instância estão destacadas em negrito na tabela.

A análise dos resultados aponta que a abordagem proposta é capaz de produzir cronogramas de horários viáveis para o problema de horários de cursos baseado em currículo. Os experimentos realizados mostram que o BRKGA híbrido pode competir com outros solucionadores para o CB-CTP, embora não seja o melhor para o conjunto de dados do ITC-2007. O algoritmo híbrido apresentou uma piora de 42.45% quando comparado com o melhor resultado e uma melhora de 13.23% quando comparado com o pior resultado da tabela. A abordagem proposta foi superior apenas da abordagem produzida por Monteiro et al. (2017).

Tabela 7 – Comparação de Resultados

| **Instâncias** | **Abordagem Proposta** | | | **Muller (2009)** | **Lu e Hão (2010)** | **Abdullah (2012)** | **Bellio (2016)** | **Kiefer (2017)** | **Monteiro (2017)** | **Segatto (2017)** | **Akkan (2018)** | **Fajrin (2020)** | **Song (2021)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MIN** | **M** | **DP** | **M** | **M** | **M** | **M** | **M** | **M** | **M** | **M** | **M** | **M** |
| comp01 | 5 | 5.00 | 0.00 | 5.00 | 5.00 | 5.00 | 5.23 | 5.00 | 5.00 | 5.00 | 5.07 | **4.37** | 5.00 |
| comp02 | 62 | 81.90 | 11.82 | 61.30 | 60.60 | 53.90 | 52.94 | **41.50** | 89.35 | 55.10 | 81.43 | 63.47 | 51.10 |
| comp03 | 85 | 94.50 | 4.47 | 94.80 | 86.60 | 84.20 | 79.16 | **71.70** | 117.00 | 84.95 | 98.77 | 81.03 | 78.10 |
| comp04 | 37 | 43.10 | 3.39 | 42.80 | 47.90 | 51.90 | 39.39 | **35.10** | 52.75 | 40.80 | 46.50 | 89.60 | 38.30 |
| comp05 | 411 | 450.60 | 23.89 | 343.50 | 328.50 | 339.50 | 335.13 | 305.20 | 536.75 | 328.45 | 367.80 | **304.10** | 318.50 |
| comp06 | 61 | 68.70 | 7.50 | 56.80 | 69.90 | 64.40 | 51.77 | **47.80** | 89.70 | 58.35 | 75.33 | 74.47 | 50.50 |
| comp07 | 29 | 36.60 | 5.44 | 33.90 | 28.20 | 20.20 | 26.39 | **14.50** | 52.60 | 28.70 | 47.47 | 87.47 | 20.00 |
| comp08 | 44 | 48.80 | 3.40 | 46.50 | 51.40 | 47.90 | 43.32 | **41.00** | 59.45 | 45.45 | 58.10 | 93.36 | 41.80 |
| comp09 | 115 | 121.20 | 4.02 | 113.10 | 113.20 | 113.90 | 106.10 | **102.80** | 125.55 | 107.90 | 118.83 | 117.43 | 105.50 |
| comp10 | 21 | 38.80 | 8.15 | 21.30 | 38.00 | 24.10 | 21.39 | **14.30** | 57.20 | 25.05 | 39.57 | 71.10 | 17.20 |
| comp11 | 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| comp12 | 415 | 446.60 | 28.30 | 351.60 | 365.00 | 355.90 | 336.84 | 319.40 | 480.00 | 353.30 | 367.20 | **297.80** | 347.70 |
| comp13 | 75 | 83.00 | 3.63 | 73.90 | 76.20 | 72.40 | 73.39 | **60.70** | 89.20 | 76.75 | 87.60 | 106.93 | 70.30 |
| comp14 | 64 | 67.80 | 3.43 | 61.80 | 62.90 | 63.30 | 58.16 | **54.10** | 75.40 | 61.00 | 68.93 | 89.53 | 57.70 |
| comp15 | 82 | 98.60 | 10.95 | 94.80 | 87.80 | 88.00 | 78.19 | **72.10** | 116.55 | 84.95 | 96.37 | 82.30 | 77.70 |
| comp16 | 40 | 46.60 | 3.98 | 41.20 | 53.70 | 51.70 | 38.06 | **33.80** | 70.05 | 44.40 | 61.23 | 74.97 | 37.60 |
| comp17 | 92 | 100.70 | 3.97 | 86.60 | 100.50 | 86.20 | 77.61 | **75.70** | 113.45 | 85.00 | 100.77 | 91.37 | 79.40 |
| comp18 | 82 | 93.30 | 4.75 | 91.70 | 82.60 | 85.80 | 81.10 | **66.90** | 98.10 | 84.85 | 90.80 | 84.03 | 82.00 |
| comp19 | 77 | 86.80 | 5.36 | 68.80 | 75.00 | 78.10 | 66.77 | **62.60** | 111.45 | 69.70 | 82.63 | 80.80 | 65.70 |
| comp20 | 51 | 59.30 | 6.83 | 34.30 | 58.20 | 42.90 | 46.13 | **27.20** | 92.85 | 45.95 | 62.80 | 82.16 | 36.70 |
| comp21 | 109 | 133.80 | 12.24 | 108.00 | 125.30 | 121.5 | 103.22 | **97.00** | 115.30 | 108.95 | 132.33 | 108.77 | 103.80 |
| **Médias** | | 105.03 | | 87.22 | 91.26 | 88.13 | 81.92 | **73.73** | 121.32 | 85.46 | 99.5 | 99.29 | 80.22 |

# 7 Conclusão

Este trabalho apresentou um algoritmo BRKGA hibridizado com a técnica de busca local *Simulated Annealing* para resolver o Problema de Horários de Cursos baseado em Currículo. A eficácia do algoritmo foi avaliada em um conjunto de 21 instâncias do ITC-2007, sobre o tempo de execução equivalente a uma máquina utilizada na competição. Os resultados obtidos com a utilização da meta-heurística BRKGA combinada com *Simulated Annealing* e Cadeia de Kempe provaram que o algoritmo é capaz de fornecer soluções de qualidade.

A análise dos resultados mostram que o conceito de chaves aleatórias do BRKGA, combinado com outras heurísticas, é capaz de produzir uma população de soluções inicias em tempo computacional viável. Os resultados do algoritmo de construção apontam não apenas a capacidade em produzir soluções de qualidade, mas também de produzir soluções melhores, ou seja, que minimiza a violação de restrições flexíveis, quando comparados com outros trabalhos da literatura.

Por fim pode-se concluir que o BRKGA híbrido, apesar de não alcançar os melhores resultados da literatura, apresenta resultados viáveis e de qualidade. O algoritmo híbrido apresentou resultados superiores apenas para um dos trabalhos da literatura. Isso indica, que um estudo mais aprofundado da estrutura da abordagem proposta precisa ser realizada, como também, a inserção de outras estruturas de vizinhanças na busca local. Este trabalho tem sua importância, pois na prática, foi o primeiro a utilizar o BRKGA para resolver o CB-CTP.

Existem várias perspectivas para pesquisas futuras, entre elas: investigar novas estruturas de vizinhanças para a busca local, investigar outros algoritmos de busca local que podem ser hibridizados com o BRKGA, executar a abordagem por mais tempo, ou seja, usar uma condição de parada diferente da permitida no ITC-2007, na tentativa de explorar ainda mais os resultados.

# Referências

ABDULLAH, S.; TURABIEH, H.; MCCOLLUM, B.; MCMULLAN, P. A hybrid metaheuristic approach to the university course timetabling problem. *Journal of Heuristics*, Springer, v. 18, n. 1, p. 1–23, 2012. Citado na página 64.

AKKAN, C.; GüLCü, A. A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem. *Computers  Operations Research*, v. 90, p. 22–32, 2018. ISSN 0305-0548. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054817302307>. Citado 4 vezes nas páginas 36, 37, 38 e 58.

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 39.

ALTUNAY, H.; EREN, T. A literature review for course scheduling problem. *Pamukkale Univ Muh Bilim Derg*, v. 23, n. 1, p. 55–70, 2017. Doi: 10.5505/pajes.2016.37233. Disponível em: <https://dx.doi.org/10.5505/pajes.2016.37233>. Citado na página 33.

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 29.

BABAEI, H.; KARIMPOUR, J.; HADIDI, A. A survey of approaches for university course timetabling problem. *Computers  Industrial Engineering*, v. 86, p. 43 – 59, 2015. ISSN 0360-8352. Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835214003714>. Citado na página 32.

BARBOSA, S. H. D.; SOUZA, S. R. d. Resolução do problema de programação de cursos universitários baseada em currículos via uma meta-heurística híbrida grasp-ils-relaxado. *XLIII Simpósio Brasileiro de Pesquisa Operacional*, p. 1976–1987, 2011. Citado na página 22.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, v. 6, n. 2, p. 154–160, 1994. Disponível em: <https://doi.org/10.1287/ijoc.6.2.154>. Citado 2 vezes nas páginas 39 e 53.

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 2 vezes nas páginas 29 e 30.

BELLIO, R.; CESCHIA, S.; GASPERO, L. D.; SCHAERF, A.; URLI, T. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research*, Elsevier, v. 65, p. 83–92, 2016. Citado 2 vezes nas páginas 35 e 38.

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 29.

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 29.

CARVALHO, A. S.; MARIANO, G. P.; KAMPKE, E. H.; MAURI, G. R. Simulated annealing aplicado ao problema de programação de hoários do cca-ufes. *Blucher Marine Engineering Proceedings*, v. 2, n. 1, p. 341–352, 2016. Citado 2 vezes nas páginas 14 e 18.

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 18.

FAJRIN, A.; FATICHAH, C. Multi-parent order crossover mechanism of genetic algorithm for minimizing violation of soft constraint on course timetabling problem. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, v. 6, n. 1, p. 43–51, 2020. ISSN 2502-3357. Disponível em: <http://journal.unipdu.ac.id:8080/index.php/register/article/view/1663>. Citado 2 vezes nas páginas 36 e 38.

GASPERO, L. D.; MCCOLLUM, B.; SCHAERF, A. *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*. [S.l.], 2007. Citado 2 vezes nas páginas 19 e 27.

GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, v. 17, n. 5, p. 487–525, Oct 2011. ISSN 1572-9397. Disponível em: <https://doi.org/10.1007/s10732-010-9143-1>. Citado 6 vezes nas páginas 39, 40, 41, 42, 43 e 60.

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 18.

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 4 vezes nas páginas 14, 15, 30 e 31.

HOSNY, M. Metaheuristic approaches for solving university timetabling problems: A review and case studies from middle eastern universities. In: ROCHA, Á.; SERRHINI, M. (Ed.). *Information Systems and Technologies to Support Learning*. Cham: Springer International Publishing, 2019. p. 10–20. ISBN 978-3-030-03577-8. Citado na página 34.

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 3 vezes nas páginas 16, 19 e 22.

JARDIM, A. M.; SEMAAN, G. S.; PENNA, P. H. V. Um algoritmo para o problema de programação de horários: Um estudo de caso. *XXII Simpóósio de Engenharia de Produção (SIMPEP)*, 2015. Citado 2 vezes nas páginas 15 e 30.

JUNIOR, B. A. *μBRKGA: Um algoritmo genético paralelo de chaves aleatórias tendenciosas aplicado ao problema de posicionamento de figuras irregulares*. Tese (Doutorado) — Universidade de Fortaleza, Fortaleza - CE, 2017. Citado 3 vezes nas páginas 40, 41 e 43.

KIEFER, A.; HARTL, R. F.; SCHNELL, A. Adaptive large neighborhood search for the curriculum-based course timetabling problem. *Annals of Operations Research*, v. 252, n. 2, p. 255–282, May 2017. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/s10479-016-2151-2>. Citado 3 vezes nas páginas 35, 36 e 38.

LEWIS, R.; PAECHTER, B.; MCCOLLUM, B. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff University, 2007. Citado na página 19.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STüTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58, 2016. ISSN 2214-7160. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214716015300270>. Citado na página 60.

LÜ, Z.; HAO, J.-K. Adaptive tabu search for course timetabling. *European journal of operational research*, Elsevier, v. 200, n. 1, p. 235–244, 2010. Citado na página 64.

MAINIERI, G. B. *Meta-heurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido*. Tese (Doutorado) — Universidade de São Paulo, São Paulo - SP, 2014. Citado 3 vezes nas páginas 40, 41 e 43.

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 2 vezes nas páginas 30 e 31.

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 29.

MONTEIRO, R. C.; KAMPKE, E. H.; BISSOLI, D. d. C.; MAURI, G. R. Algoritmo híbrido iterated local search e simulated annealing para o problema de tabela-horário de universidades. *Anais do XLIX Simpósio Brasileiro de Pesquisa Operacional*, p. 1867–1878, 2017. Citado 7 vezes nas páginas 16, 21, 35, 38, 44, 61 e 64.

MOREIRA, L. V.; MONTEIRO, R. C.; KAMPKE, E. H.; MAURI, G. R. Meta-heurística grasp para o problema de tabela-horário de disciplinas do departamento de computação do cca-ufes. *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, p. 2171–2182, 2016. Citado na página 14.

MORO, M. A. *Meta-heurísticas GRASP e BRKGA Aplicadas ao Problema da Diversidade Máxima*. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas - SP, 2017. Citado 3 vezes nas páginas 40, 41 e 43.

MOURA, M. A. *Algoritmo Genético de Chaves Aleatórias segundo Distribuição de Levy para Otimização Global*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife - PE, 2018. Citado 4 vezes nas páginas 39, 40, 41 e 43.

MÜLLER, T. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, Springer, v. 172, n. 1, p. 429–446, 2009. Citado na página 64.

NETO, R. F. d. O.; MARIANO, C. C. L.; FARIAS, M. S. R. Horário universitário personalizado por meta-heurıstica: Um estudo de caso. *XLIX Simpósio Brasileiro de Pesquisa Operacional*, p. 744–754, 2017. Citado na página 30.

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 2 vezes nas páginas 14 e 29.

NOGAREDA, A.-M.; CAMACHO, D. Optimizing satisfaction in a multi-courses allocation problem combined with a timetabling problem. *Soft Computing*, v. 21, n. 17, p. 4873–4882, Sep 2017. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-016-2375-8>. Citado na página 19.

PEREIRA, J. Q. *Uma Abordagem Via Metaheurística Híbrida para o Problema de Atribuição de Localidades a Anéis SONET/SDH*. Dissertação (Mestrado) — Universidade do Estado do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró - RN, 2018. Citado na página 43.

PILLAY, N. Evolving construction heuristics for the curriculum based university course timetabling problem. In: . [S.l.: s.n.], 2016. p. 4437–4443. Cited By 3. Citado 2 vezes nas páginas 28 e 29.

PILLAY, N. A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, v. 239, n. 1, p. 3–38, Apr 2016. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/s10479-014-1688-1>. 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 2 vezes nas páginas 28 e 29.

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 3 vezes nas páginas 14, 20 e 29.

QUEIROZ, D. L. d.; NEPOMUCENO, N. V. Um modelo em programação linear inteira para alocação de disciplinas: Um estudo de caso no curso de ciância da computação da universidade de fortaleza. *XLIX Simpósio Brasileiro de Pesquisa Operacional*, p. 2914–2925, 2017. Citado 4 vezes nas páginas 14, 18, 19 e 29.

ROCHA, W. d. S. *Algoritmo GRASP para o Problema de Tabela-horário de Universidades*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Vitória - ES, 2013. Citado 4 vezes nas páginas 15, 16, 23 e 44.

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 4 vezes nas páginas 18, 19, 28 e 29.

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. Citado 14 vezes nas páginas 16, 36, 38, 44, 45, 46, 47, 48, 49, 50, 51, 61, 63 e 64.

SEGATTO, E. d. A.; BOERES, M. C. S.; RANGEL, M. C.; KAMPKE, E. H. Um algoritmo grasp com cadeia de kempe aplicado ao problema de tabela-horário para universidades. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, p. 2643–2654, 2015. Citado 4 vezes nas páginas 21, 35, 36 e 38.

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 29.

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 29.

SONG, T.; CHEN, M.; XU, Y.; WANG, D.; SONG, X.; TANG, X. Competition-guided multi-neighborhood local search algorithm for the university course timetabling problem. *Applied Soft Computing*, v. 110, p. 107624, 2021. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494621005457>. Citado 2 vezes nas páginas 37 e 38.

SORIA-ALCARAZ, J. A.; ÖZCAN, E.; SWAN, J.; KENDALL, G.; CARPIO, M. Iterated local search using an add and delete hyper-heuristic for university course timetabling. *Applied Soft Computing*, v. 40, p. 581–593, 2016. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494615007760>. Citado 2 vezes nas páginas 35 e 38.

SOUZA, M. J. F. *Programação de horários em escolas: uma aproximação por metaheurísticas.* Tese (Doutorado) — Universidade Federal de Rio de Janeiro, Rio de Janeiro - RJ, 2000. Citado 2 vezes nas páginas 18 e 19.

SPEARS, V. M.; JONG, K. A. D. On the virtues of parameterized uniform crossover. In: *In Proceedings of the Fourth International Conference on Genetic Algorithms*. [S.l.: s.n.], 1991. p. 230–236. Citado na página 40.

SPINDLER, M.; CHIWIACOWSKY, L. D. Uma proposta de solução para problemas de horário educacional utilizando busca dispersa e reconexão por caminhos. *XLII Simpósio Brasileiro de Pesquisa Operacional*, p. 1722–1733, 2010. Citado na página 22.

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 4 vezes nas páginas 14, 15, 16 e 31.

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 6 vezes nas páginas 14, 22, 33, 51, 63 e 64.

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 2 vezes nas páginas 30 e 31.

WAHID, J.; ABDUL-RAHMAN, S.; DIN, A. M.; MOHD-HUSSIN, N. Constructing population of initial university timetable: Design and analysis. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 15, n. 2, p. 1109–1118, 2019. Citado 2 vezes nas páginas 63 e 64.

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 3 vezes nas páginas 50, 63 e 64.