Explorando Diferentes Mutações de Um Algoritmo Genético para o Curriculum-Based Course Timetabling Problem

João E. R. Manica[1]

[1]Universidade de Passo Fundo (UPF) – Passo Fundo – RS – Brasil

joaoedisonmanica@gmail.com

Abstract. *In this work, two mutations of a genetic algorithm for the Curriculum-Based Course Timetabling (CB-CTT) problem are proposed, implemented, and validated. One of the mutations is based on a forward checking mechanism and the other on a repair mechanism. It is concluded that, compared to existing mutations, the number of conflicts in the main constraints was reduced.*

Resumo. *Neste trabalho, duas mutações de um algoritmo genético para o Curriculum-Based Course Timetabling (CB-CTT) problem são propostas, implementadas e validadas. Uma das mutações é baseada em um mecanismo de avaliação adiante e a outra num mecanismo reparador. Conclue-se que, em comparação às mutações existentes, foi reduzido o número de conflitos das principais restrições.*

1. Introdução

Organizar o cronograma de universidades é um problema pertencente à classe de complexidade computacional NP-completo. Mesmo em algumas versões bem simplificadas do problema, considerando somente os horários de disponibilidade dos professores, os horários das aulas e as horas que devem ser atribuídas para cada professor durante a semana [Even et al. 1976]. Logo, encontrar um agendamento válido por meio de um algoritmo, para qualquer configuração do problema, é necessariamente custoso em relação ao tempo.

Programar o cronograma consiste, basicamente, em alocar professores a turmas e turmas a horários, visto que isso se trata de um problema que pode ser expresso em termos de encontrar o melhor conjunto de valores para um grupo de variáveis, então é um problema de otimização combinatória [Coppin 2015].

As meta-heurísticas são os métodos usados pelas técnicas de busca local para solucionar problemas de otimização combinatória. Técnicas de busca local funcionam partindo de uma configuração inicial, fazendo-se pequenas alterações nesta configuração até encontrar um estado limite considerado ótimo. Dentro das meta-heurísticas, se encontram os algoritmos genéticos, os quais usam técnicas que imitam a teoria da evolução dos seres vivos [Coppin 2015].

Algoritmos genéticos empregam os três princípios de Darwin: variação genética, hereditariedade e seleção natural. Os indivíduos evoluem de acordo com regras de seleção e operadores genéticos, tais como a recombinação e a mutação [Goldbarg et al. 2015].

Atendendo à quantidade de cursos existentes na universidade, é preciso solucionar o problema em tempo hábil, para tanto foi proposto um algoritmo genético. O algoritmo

considera vários critérios que o cronograma deve seguir, tais como as diversas modalidades de cursos existentes: presencial, educação a distância (EaD) ou híbridos.

No algoritmo implementado, a mutação é a grande propulsora da evolução, porém ainda não resolve rapidamente todas as restrições mais importantes. Com o intuito de melhorar a evolução para gerar melhores indivíduos, o presente trabalho projeta, implementa e valida duas novas opções de mutações.

Nas seções seguintes, é explicado do que se trata o *Curriculum-Based Course Timetabling* (CB-CTT) *problem*, são abordados os conceitos fundamentais de algoritmos genéticos, os trabalhos relacionados, metodologia, que explica o estado atual do algoritmo e os novos operadores genéticos, recursos utilizados no trabalho, resultados obtidos nos testes realizados e, no final, a conclusão do estudo e possíveis trabalhos futuros.

2. Fundamentação Teórica

2.1. CB-CTT

O *Curriculum-Based Course Timetabling* (CB-CTT) *problem* [Bettinelli et al. 2015] ou *university curriculum-based course timetabling problem* (CCTP) [Mansour and El-Jazzar 2013] consiste em montar o cronograma semanal dos cursos universitários de maneira que satisfaça as restrições mais rígidas (*hard constraints*) e as mais flexíveis (*soft constraints*).

Durante uma semana há um número fixo de dias de trabalho e cada dia é dividido em intervalos de tempo. As disciplinas têm um número pré-determinado de aulas e devem ter um professor, também devem acontecer num período em que estão disponíveis. Os cursos universitários são formados por uma coleção de disciplinas que certamente têm estudantes em comum [Mansour and El-Jazzar 2013]. Na descrição original, as salas de aula fazem parte do problema.

As *hard constraints* são as restrições que precisam ser cumpridas para que o cronograma seja válido, por exemplo: todas as aulas de uma disciplina devem ser atribuídas a diferentes períodos, aqui um período denomina um horário de um dia; aulas ministradas pelo mesmo professor ou que pertencem ao mesmo currículo devem estar em períodos diferentes; e cada aula deve estar dentro de um período disponível para a sua disciplina [Bettinelli et al. 2015].

As *soft constraints* otimizam aspectos que não são obrigatórios de um cronograma, tais como: para cada disciplina, as aulas devem ser distribuídas em um mínimo de dias; e para cada currículo, as aulas devem estar contiguamente atribuídas [Mansour and El-Jazzar 2013].

2.2. Algoritmos Genéticos

Os algoritmos genéticos trabalham com uma população de indivíduos durante o processo de busca, um indivíduo é uma solução em potencial, que é representado univocamente pelo seu cromossomo. A primeira população normalmente nasce de maneira randômica. Cada população pertence a uma geração e o comportamento dos indivíduos é traduzido por meio de uma função matemática, chamada de Função de Avaliação [Filho 2023].

O algoritmo termina depois de um determinado número de gerações ou quando alguma população atende a um critério de parada. Se não, são selecionados quais indivíduos

darão origem à nova geração, que são chamados de pais, e são gerados indivíduos-filho para completar a nova geração e substituir indivíduos repetidos que surgirem na fase de seleção, por meio da reprodução [Filho 2023].

Nesse contexto, a mutação é aplicada para alterar o material genético dos cromossomos da nova geração e torná-la mais diversificada. Se trata, geralmente, de um processo estocástico de busca que envolve três elementos: a escolha dos cromossomos que serão candidatos a sofrer mutação, a probabilidade de a mutação acontecer e a extensão da mutação. A extensão é normalmente definida pela taxa de mutação, que determina o percentual de posições que são alteradas no cromossomo [Goldbarg et al. 2015].

3. Trabalhos Relacionados

Vários trabalhos exploraram usar mutações mais intrincadas, em algoritmos genéticos, para resolver o agendamento dos cursos universitários. Raghavjee e Pillay implementaram quatro operadores de mutação na implementação deles de um algoritmo genético para solucionar a programação de horários escolares. Um dos operadores se baseia em procurar uma tupla de disciplina e professor com alguma violação de restrições e trocar com uma tupla aleatória (1V). Outro operador faz a busca de duas tuplas com violações de restrições e faz a troca entre elas (2V). Os outros dois operadores são baseados nesses, mas implementam a técnica de subida de colina. Eles concluíram que, dentre todos esses operadores, o 2V foi o que produziu o maior número de horários viáveis [Raghavjee and Pillay 2012].

Gozali et al. definiram um o operador de mutação que faz três tipos de trocas em sequência: a primeira (M1) move uma turma para um horário livre; a segunda (M2) busca uma turma que apresenta violações de restrições e escolhe aleatoriamente uma outra turma, depois é feita a troca entre as duas e a alteração só é mantida se diminuir o número de violações, muito similar ao operador (1V); a última (M3) encontra uma turma com violações de restrições e busca outras duas turmas de quaisquer disciplinas que tem salas do mesmo tamanho, a troca é feita com aquela que produzir menos conflitos e só é efetivada se diminuir o número total de conflitos do indivíduo [Gozali et al. 2020].

No trabalho de Matias et al, após o *crossover* e a mutação, são aplicados *neighborhood operators*, que são determinados usando-se um mecanismo de auto-adaptação e, depois, um operador reparador, se os melhores indivíduos não forem viáveis. O mecanismo de auto-adaptação escolhe entre quatro *neighborhood structures*, que modificam os genes (aulas) de maneiras distinas. O operador de reparação é aplicado nos melhores indivíduos e faz a correção das salas e horários das disciplinas que não são possíveis no cronograma, atribuindo-as a salas e horários livres [Matias et al. 2018].

4. Materiais e Métodos

4.1. A Implementação Atual

Neste algoritmo genético, os cromossomos são representados pelo conjunto das grades horárias de todos os cursos/níveis. Uma grade horária comporta 64 períodos, nos quais as disciplinas (genes) podem ser atribuídas. Um crédito equivale a um horário e cada turno, de um dia da semana, é formado por quatro horários consecutivos. Cada um dos créditos pode ser interpretado como sendo um alelo e os créditos das disciplinas fora de horário são gerenciados em estruturas separadas. O presente cronograma formulado para as disciplinas da universidade não faz a atribuição de aulas à salas.

Os cursos da universidade têm disciplinas que podem ocorrer de manhã, de tarde ou de noite de segunda a sexta e sábado de manhã. No total, há 31 combinações de turnos de funcionamento. A implementação trata disciplinas que devem ser oferecidas nos mesmos horários para todos os cursos/níveis que participam; essas disciplinas são as disciplinas compartilhadas. Também existem disciplinas que ocorrem em modalidade EaD e disciplinas com créditos que não devem ocupar lugar nas grades de horários: são as disciplinas sem horário. Independente do tipo, todas as disciplinas contam na carga horária dos professores ministrantes. Se uma disciplina tiver várias turmas no mesmo período, então essas turmas devem ser lecionadas por professores diferentes.

No Algoritmo 1, está exemplificado o funcionamento do algoritmo genético. As constantes numéricas que especificam o tamanho da população, a parcela da população que é inicializada heuristicamente, o número de gerações e a porção da população que sofre mutação são parametrizadas, aqui se apresentam como constantes numéricas de forma a facilitar o entendimento e estão ajustadas de acordo com as configurações dos testes.

Algoritmo 1 Algoritmo genético

```
1:  DIM cromossomos(1000)
2:  i ← 0
3:  enquanto i < 200 faça
4:      INICIALIZAÇÃOHEURÍSTICA(cromossomos[i])
5:      i ← i + 1
6:  fim enquanto
7:  enquanto i < 1000 faça
8:      INICIALIZAÇÃOALEATÓRIA(cromossomos[i])
9:      i ← i + 1
10: fim enquanto
11: geração ← 0
12: enquanto geração < 500 faça
13:     Avalie cromossomos
14:     cromossomos ← Selecione nova população por torneio com 2 rondas e K=10
15:     i ← 0
16:     enquanto i < 600 faça
17:         MUTAÇÃO(cromossomos[i])
18:         i ← i + 1
19:     fim enquanto
20:     geração ← geração +1
21: fim enquanto
```

Os cromossomos são inicializados da seguinte forma (linhas 1 a 10): os níveis são ordenados em ordem descendente pela quantidade de disciplinas compartilhadas. Dentro de uma grade horária de um nível, primeiro são postas as disciplinas compartilhadas, em ordem das disciplinas que possuem aula sábado de manhã, que têm mais disciplinas no seu grupo compartilhado, que mais disciplinas do grupo foram atribuídas ou que têm mais créditos; seguido das disciplinas convencionais, em ordem das que têm mais créditos nas grades. Os horários de uma disciplina compartilhada que já fora escolhida na alocação são reaproveitados para todos os níveis de cursos do mesmo compartilhamento, se houver algum conflito, então é escolhido um outro horário, causando conflitos de compartilhamento.

A inicialização heurística (linhas 2 a 6) faz com que primeiro os professores com a maior diferença entre as horas definidas das horas que estão sendo atribuídas, durante o processo de inicialização, sejam escolhidos para ministrarem as disciplinas.

Na implementação atual, o processo de seleção é híbrido, parte da população

sendo selecionada seguindo o critério elitista e o restante é selecionado por torneio. O operador genético de recombinação (*crossover*) é usado em parte dos indivíduos selecionados no torneio e no restante é aplicada a mutação. Porém, no presente estudo, toda a população é selecionada por torneio com duas rondas, escolhendo-se sempre o melhor entre os dez candidatos, e o único operador genético empregado é a mutação, que está sendo aplicada a 60% dos indivíduos (linhas 14 a 19). Então, o presente trabalho segue uma estratégia de programação evolucionária, com o intuito de isolar os efeitos da mutação ao longo das gerações.

4.1.1. Função de Avaliação

A Função de Avaliação está definida na Equação 1. Atualmente, 13 restrições são utilizadas para direcionar as soluções, elas se encontram na Tabela 1. [Fabian 2024].

$$f = \sum_{i=1}^{n} w_i \times \frac{l_i - v_i}{l_i} \tag{1}$$

f	Fitness do indivíduo
n	Número de restrições
w_i	Peso da i-ésima restrição
l_i	Limite da i-ésima restrição
v_i	Violações da i-ésima restrição

Na Equação 1, o peso da restrição (w) e o limite do número de conflitos (l) determinam quão importante uma restrição é. O número de violações (v) se refere a soma do número de conflitos para dada restrição de um cromossomo inteiro.

Tabela 1. Restrições do algoritmo genético

Restrição	Peso	Limite	Tipo
Oferecimento	500000	20	Hard
Compartilhamento	300000	2000	Hard
Clash	200000	1600	Hard
Indisponibilidade	100000	350	Hard
Titulação Doutor	40000	200	Soft
Regime de Trabalho	20000	400	Soft
Titulação Mestre	20000	200	Soft
Horas Diárias	10000	400	Soft
Máximo de Horas	10000	200	Soft
Prioridade	1000	200	Soft
Horas Professor	200	4000	Soft
Excesso de Turnos	200	20	Soft
Excesso de Dias	20	200	Soft

A Tabela 1 contém quatro restrições do tipo *hard* e nove restrições do tipo *soft*. A restrição *Oferecimento* conta cada curso/nível de um indivíduo que ofertou um número

incorreto de créditos. Esse tipo de conflito é causado por inconsistências na base de dados ou erros na inicialização dos cromossomos e não é resolvido durante a evolução dos indivíduos.

A restrição de *Compartilhamento* é o número de professores ou horários diferentes para cada crédito das disciplinas que são do mesmo grupo compartilhado. Compara-se cada par de disciplinas (d_1 e d_2) do mesmo grupo compartilhado, que são identificadas pela tupla (grupo compartilhado, horário, professor e créditos), entre todos os cursos/níveis. A soma do total de conflitos é feita seguindo as seguintes regras: se o professor, o horário ou o tipo de crédito (teórico ou prático) for diferente entre as duas disciplinas, então soma-se o total de créditos de d_1, se não se o número de créditos for diferente, então soma-se a diferença absoluta entre os créditos de d_1 e d_2. Por exemplo, se houverem quatro disciplinas que pertencem ao mesmo grupo compartilhado: $(g, x, p_1, 4)$, $(g, x, p_1, 2)$, $(g, x, p_2, 4)$ e $(g, y, p_1, 4)$. Para este grupo, o total de erros de compartilhamento seria igual a diferença absoluta do número de créditos da primeira disciplina com a segunda disciplina (2), mais os quatro créditos com o professor diferente da primeira (p_1) com a terceira (p_2), mais os quatro créditos no horário diferente da primeira (x) com a última (y); depois, comparando-se a segunda com a terceira, somaria dois créditos, pois o professor (p_1) é diferente na terceira disciplina (p_2) e mais dois créditos, porque o horário é diferente entre a segunda (x) e a quarta (y); por fim, comparando a terceira com a quarta somaria mais quatro créditos, que estão em diferentes horários (x e y). Então, a soma total de conflitos para esse grupo compartilhado seria igual a 18.

Neste estudo, um *Clash* acontece quando um professor é posto para lecionar mais de uma disciplina, que não é compartilhada, no mesmo período. Logo, para cada crédito das disciplinas lecionadas pelo mesmo professor, no mesmo horário, está contando uma violação com as demais do mesmo horário. Se a disciplina for compartilhada, só um crédito entre todos os cursos/níveis do mesmo compartilhamento é comparado com os demais. Se um professor p_1 vai ministrar quatro disciplinas d_1, d_2, d_3 e d_4, d_1 não é compartilhada, inicia no horário x e tem três créditos, d_2 é do mesmo grupo compartilhado que d_3, ambas tem 2 créditos e iniciam no mesmo horário que d_1 e d_4 não é compartilhada inicia no horário $x + 1$ e tem dois créditos. O total de conflitos de *clashes* presentes nessa atribuição é calculado desta forma: no horário x, há um conflito entre d_1 contra d_2 e d_3; no horário $x + 1$, há um conflito entre d_1 e d_2 e d_3 mais um conflito entre d_1 e d_4 e mais um conflito entre d_2 e d_3 e d_4, totalizando três conflitos; e no horário $x + 2$ há um conflito entre d_1 e d_4. Logo, essa atribuição resultou em cinco conflitos de clashes para o professor p_1.

A restrição de *Indisponibilidade* soma todos os créditos de disciplinas que estão nos turnos de indisponibilidade dos respectivos professores.

As restrições *Titulação Doutor*, *Titulação Mestre* e *Regime de Trabalho* priorizam a escolha dos professores nos cursos que fazem parte do Exame Nacional de Desempenho dos Estudantes (ENADE). A primeira restrição conta o total de professores que não tem o título de doutorado em todos os níveis de todos os cursos que fazem parte do ENADE naquele ano. A segunda restrição conta o total de professores que não tem título de mestre ou doutorado nos mesmos cursos. A terceira restrição conta quantos professores são horistas ou possuem regime de trabalho indefinido nesses cursos [Fabian 2024].

Exitem três restrições que têm como propósito ajustar a jornada de trabalho dos professores. A restrição de *Horas Diárias* conta o total dos excessos de horas alocadas a um professor nos dias da semana para todos os professores. A restrição de *Máximo de Horas* é o resto da diferença entre o total de horas alocadas e o total de horas máximo para todos os professores. Caso as horas alocadas a um professor forem menores que o total de horas máximo, a diferença não é calculada. A restrição de *Horas Professor* é dada pela Equação 2. Tem como objetivo aproximar o máximo possível as horas alocadas a um professor das horas definidas àquele. Se forem reservadas mais horas do que as definidas, então é contabilizado mais do que se forem reservadas menos horas ao professor.

$$v = \begin{cases} (h_a - h_d)^{w_h+2} & \text{se } h_a > h_d, \\ (h_d - h_a) \times w_h & \text{se } h_a \leq h_d \end{cases} \tag{2}$$

v	Violações de *Horas Professor* para um professor
h_a	Número de horas atribuídas a esse professor
h_d	Número de horas definidas para esse professor
w_h	Peso das horas desse professor $\in \{0, 1, 2\}$

A restrição de *Prioridade* foi feita para favorecer a escolha de certos professores. Computa a diferença entre o professor com a maior prioridade possível e o total de pontos de prioridade que o professor alocado para lecionar uma disciplina tem. Para as disciplinas compartilhadas, é comparado com o professor mais prioritário de cada curso do compartilhamento.

Duas restrições controlam as atribuições das disciplinas para que fiquem adequadamente distribuídas nos dias da semana. O *Excesso de Turnos* conta o total de dias da semana com mais do que dois turnos com aulas para todos os cursos/níveis. A restrição de *Excesso de Dias* conta a diferença entre o total de dias com aulas e o mínimo de dias possível para todos os níveis de cursos.

4.1.2. Mutação

O operador de mutação faz com que os algoritmos genéticos escapem de soluções subótimas, ou máximos locais. Trata-se de um operador que é aplicado a um único gene e altera o seu valor [Coppin 2015]. No presente estudo, as disciplinas são formadas por créditos consecutivos, logo, para que o operador não gere disciplinas lecionadas por mais de um professor ou disciplinas com créditos descontínuos, o grupo inteiro de horários que representa uma disciplina deve ser operado. A função de mutação utilizada pelo algoritmo genético está definida no Algoritmo 2.

Algoritmo 2 Mutação original

```
1: procedimento MUTAÇÃO(cromossomo)
2:     pMut ← 0,005
3:     pComp ← pMut
4:     se conflitos de compartilhamento > 0 então
5:         pComp ← 0,01
6:     fim se
7:     PROFESSORES(pMut, pComp)
8:     HORÁRIOS(pComp)
9: fim procedimento
```

No Algoritmo 2, os parâmetros *pMut* e *pComp* definem as taxas de mutação. O primeiro é aplicado sobre disciplinas convencionais e sem horário e o segundo somente em disciplinas que são compartilhadas. Os procedimentos *professores* (Algoritmo 5 - Apêndice A) e *horários* (Algoritmo 6 - Apêndice B) são utilizados em conjunto para realizar a mutação.

Quando são utilizadas em conjunto as mutações sobre os professores e sobre os horários faz com que os indivíduos tenham uma melhor pontuação final, isso dá-se por possibilitar mais combinações de horários e professores por turma.

O procedimento *professores*, que é descrito no Algoritmo 5, começa escolhendo compartilhamentos de acordo com a probabilidade de *pComp* sobre todos os compartilhamentos existentes na base de dados (linhas 2 e 3). Depois, sorteia-se novos professores aptos a ministrarem a disciplina segundo o curso/nível atual, para cada uma das turmas (linhas 4 a 23). Se não há professores aptos e é permitido escolher um professor indefinido, este é posto no lugar. Se a disciplina tem créditos EaD, o professor é substituído neles também (linha 20). Isso só é feito nas disciplinas compartilhadas presentes em $mutaComp$ e é feita a troca com esses professores sorteados nas outras disciplinas que pertencem ao mesmo grupo compartilhado. Nas linhas 24 a 31, está definido que as disciplinas sem horário são escolhidas segundo a probabilidade de *pMut* sobre todas as disciplinas sem horário do curso/nível atual. Pois então, o professor é substituído por um outro professor apto a ministrar a disciplina, o critério é também aleatório.

A mutação de *horários*, que está definida no Algoritmo 6, escolhe aleatoriamente quais compartilhamentos serão alterados de todos os compartilhamentos (linhas 2 e 3), da mesma forma que no Algoritmo 5. Na linha 5, é escolhido randomicamente um curso/nível para, depois, trocar o horário de alguma disciplina. Se a disciplina escolhida for compartilhada, então o horário é trocado por um outro horário aleatório que seja comum a todas as disciplinas desse compartilhamento. As demais disciplinas desse grupo compartilhado são realocadas para esse horário escolhido. Caso contrário, quando a disciplina não é compartilhada, se o curso/nível for igual ao $selecionado$, segue-se com a escolha de um horário dentro dos que são permitidos pelo curso em questão (linhas 6 a 27).

Na linha 23, acontece a troca entre as disciplinas seguindo algumas regras: quando houver uma disciplina no horário escolhido (x) e os créditos forem diferentes entre as duas, ou elas estiverem em turnos diferentes, ou d_i ocupar um turno inteiro, a troca é feita entre os turnos inteiros em que estão os horários das disciplinas. Quando o horário estiver livre e a disciplina d_i for compartilhada, mas é a primeira vez que está sendo movida alguma disciplina do compartilhamento $comp$, também troca-se o turno inteiro. Se não,

troca-se apenas as disciplinas d_i e d_j em questão. Quando não tem nenhuma disciplina em x, d_i é simplesmente realocada para x.

Em resumo, as disciplinas que são compartilhadas estão sendo reposicionadas de acordo com uma taxa de *pComp* e, obrigatoriamente, uma disciplina convencional ou mais uma disciplina compartilhada é transferida para um novo horário válido. Vale ressaltar que, se a disciplina escolhida for compartilhada no curso/nível selecionado e não estiver em $mutaComp$, então a troca vai causar conflitos de compartilhamento.

Há outro erro quando muda-se o horário de uma disciplina de um grupo compartilhado g_i, num curso/nível n_1, e, depois, muda-se o horário de uma disciplina pertencente a outro compartilhamento g_j em n_2, por existir mais de um compartilhamento em $mutaComp$. Se g_i e g_j tiverem disciplinas num mesmo curso/nível n_3, então o horário de só uma disciplina compartilhada será ajustado. Isso também pode ocorrer se $selecionado$ contiver alguma disciplina compartilhada dos grupos de $mutaComp$, mas uma disciplina convencional for escolhida. Se for removido o comando *Pare* na linha 24 do Algoritmo 6, o erro pode ser corrigido, mas no curso/nível selecionado tem de se alterar obrigatoriamente uma disciplina convencional.

4.2. Novos Operadores de Mutação

Ao todo, são propostos dois novos operadores de mutação, um que age sobre os professores (*profSClash*) e outro sobre os horários (*horCRepComp*). Também são definidas versões alternativas desses dois, nas quais os mecanismos principais estão desativados e servem para comprovar a consequência do uso deles.

O procedimento *profSClash* (Algoritmo 3) se baseia na avaliação adiante como mecanismo principal. A avaliação adiante consiste em eliminar do conjunto de possíveis escolhas futuras qualquer uma que se revele não válida [Coppin 2015]. Adaptando a ideia para o contexto do algoritmo genético estudado, pode-se fazer com que as trocas de professores considerem somente aqueles que estão livres nos horários em questão, isto é, troca-se por professores que não causem novos conflitos de *clash* quando forem atribuídos.

Algoritmo 3 Nova mutação de professores - profSClash

```
1: procedimento PROFSCLASH(pMut)
2:     compAlt ← ∅
3:     para Todos os Cursos/Níveis faça
4:         para Todas as Disciplinas faça
5:             d ← Disciplina Atual
6:             se d é compartilhada então
7:                 comp ← Código compartilhado de d
8:                 se comp ∈ compAlt então
9:                     Continue
10:                fim se
11:            fim se
12:            se d foi sorteada por pMut então
13:                professores ← Professores aptos a ministrarem d
14:                x ← Horário de d
15:                professores ← professores que podem lecionar nos turnos de x
16:                professores ← professores que não estão lecionando em x
17:                novoP ← Seleciona um professor por uma roleta simples da prioridade dos professores
18:                p_i ← Professor de d
19:                prioridade_i ← Prioridade de p_i
20:                extras ← Professores das demais turmas de d
21:                p_j ← Professor menos prioritário entre os extras
22:                prioridade_j ← Prioridade de p_j
23:                se prioridade_j < prioridade_i então
24:                    p_j ← p_i
25:                fim se
26:                p_i ← novoP
27:                se d é compartilhada então
28:                    para ∀ Curso/Nível ∈ grupo compartilhado de comp faça
29:                        Alterações das linhas 18 a 26 com novoP
30:                    fim para
31:                    compAlt ← compAlt ∪ comp
32:                fim se
33:            fim se
34:        fim para
35:    fim para
36: fim procedimento
```

Nesta nova versão do procedimento que opera sobre os professores, as disciplinas que são compartilhadas são tratadas da mesma forma que as disciplinas normais. Os pontos alterados podem estar em quaisquer níveis de cursos (linhas 3, 4 e 12). Escolhe-se os professores que são aptos a ministrarem a disciplina, inclusive o professor indefinido, que não estão lecionando nos horários da disciplina escolhida, para que a troca não cause novos *clashes*. Um professor é selecionado por uma roleta simples entre a prioridade dos professores (linhas 13 a 17).

Então o professor ganhador passa a lecionar a turma atual. Se tiver um professor menos prioritário do que o professor que estava lecionando a primeira turma, ele é substituído pelo primeiro (linhas 18 a 26). Quando a disciplina é compartilhada, as trocas acontecem da mesma forma, com o mesmo professor ganhador, mas nos outros cursos/níveis do compartilhamento. O grupo compartilhado é marcada como modificado para não ser escolhida uma outra disciplina do mesmo grupo (linhas 27 a 32). Na linha 26, acontecem também as atribuições de p_i para $novoP$ nos créditos sem horário e EaD.

Com base nesta nova implementação, para verificar as consequências do mecanismo empregado, foi derivada uma outra versão desse procedimento equivalente, mas sem o filtro dos professores que estavam ministrando as disciplinas e é feito um sorteio aleatório entre todos os professores ao invés de usar uma roleta simples pela prioridade (Algoritmo 3, linhas 13 a 17). Este outro procedimento foi chamado de *profAleCorr*, pois

faz a troca de professores das disciplinas comuns, se for comparado ao Algoritmo 5, isso corrige uma falha da implementação anterior.

A mutação de horários das disciplinas (Algoritmo 6) pode, em vez de mover todas as disciplinas de um compartilhamento para um novo horário (linhas 12 a 19), agrupar todas as disciplinas do mesmo grupo compartilhado no mesmo horário de alguma disciplina pertencente a esse grupo (d_i), estas etapas estão descritas nas linhas 15 a 28 do Algoritmo 4. Posteriormente, todas elas são movidas para um novo horário (Algoritmo 4, linhas 29 a 35). Entretanto, se a disciplina d_i estiver fragmentada, a reparação vai falhar para todos os cursos/níveis. Enfim, executa-se uma reparação e depois uma troca.

Além disso, é possível usar somente a taxa de mutação *pMut* para escolher as disciplinas que serão comutadas (Algoritmo 4, linha 14), independente se são compartilhadas ou não, permitindo que sejam trocados os horários de mais de uma disciplina por curso/nível. Essas mudanças constituem outro operador de mutação, *horCRepComp*, o qual está definido no Algoritmo 4.

Algoritmo 4 Nova mutação de horários - horCRepComp

```
1:  procedimento HORCREPCOMP(pMut)
2:      compAlt ← ∅
3:      para Todos os Cursos/Níveis faça
4:          nível ← (curso, nível)
5:          para Todas as Disciplinas faça
6:              d_i ← Disciplina Atual
7:              x ← Horário de d_i
8:              se d_i é compartilhada então
9:                  comp_i ← Código compartilhado de d_i
10:                 se comp_i ∈ compAlt então
11:                     Continue
12:                 fim se
13:             fim se
14:             se d_i foi sorteada por pMut então
15:                 se d_i é compartilhada então
16:                     para ∀ Curso/Nível ∈ grupo compartilhado de comp_i faça
17:                         outroNível ← (curso, nível)
18:                         se outroNível = nível então
19:                             Continue
20:                         fim se
21:                         d_j ← Disciplina em x do outroNível
22:                         comp_j ← Código compartilhado de d_j
23:                         se ∄comp_j OU comp_j ≠ comp_i então
24:                             d_k ←
25:                             Disciplina do outroNível que ∈ grupo compartilhado de comp_i
26:                             Troque d_j e d_k
27:                         fim se
28:                     fim para
29:                     y ← Horário em comum a todos os Cursos/Níveis do grupo compartilhado de comp_i que comporta d_i
30:                     para ∀ Curso/Nível ∈ grupo compartilhado de comp_i faça
31:                         outroNível ← (curso, nível)
32:                         d_j ← Disciplinas em x do outroNível
33:                         d_k ← Disciplinas em y do outroNível
34:                         Troque d_j e d_k
35:                     fim para
36:                     compAlt ← compAlt ∪ comp_i
37:                 se não
38:                     y ← Horário dentro dos turnos de nível que comporta d_i
39:                     d_j ← Disciplinas em y de nível
40:                     Troque d_i e d_j
41:                 fim se
42:             fim se
43:         fim para
44:     fim para
45: fim procedimento
```

No momento em que são trocadas as disciplinas na linha 26, a disciplina d_j pode acabar se fragmentando, pois o algoritmo realiza a troca independente do número de créditos. Enquanto que as trocas nas linhas 34 e 40 nunca fragmentam as disciplinas, porque y é escolhido entre os horários que obrigatoriamente têm disciplinas que somadas possuem o mesmo número de créditos ou menos que d_i.

Inclusive foi derivada uma versão alternativa do Algoritmo 4, na qual o trecho de código das linhas 16 a 28, que agrupa as disciplinas compartilhadas para depois movê-las para um novo horário y, não é executado. Essa versão foi denominada de *horáriosCorr*, nos resultados, e tem como objetivo comparar a eficácia do mecanismo reparador de compartilhamentos.

5. Implementação

5.1. Recursos

O algoritmo genético foi implementado na linguagem de programação C++, compilado com o *GNU Compiler Collection* (versão 11.4.0). Todos os experimentos foram conduzidos num computador com processador Intel® Core™ i7-7700 @ 3.60 GHz (8 núcleos), memória RAM de 8 GB e sistema operacional Ubuntu 22.04.5 LTS. As estatítiscas e os diagramas foram feitos usando a linguagem Python na versão 3.10.15 e as bibliotecas SciPy[1] (1.14.1), Pandas[2] (2.2.3), Seaborn[3] (0.13.2) e Matplotlib[4] (3.9.2).

5.2. Resultados

O algoritmo genético gera a grade de horário para 308 níveis de 158 cursos com um total de 1597 disciplinas e 607 professores distintos. Dentre as disciplinas ofertadas cerca de 15,34% (245) são compartilhadas, que pertencem a 210 grupos de compartilhamento.

Foram realizadas 32 execuções para cada uma das mutações e o algoritmo foi configurado com os parâmetros da Tabela 2 nos testes. O melhor indivíduo entre os 1000 da 500ª geração está sendo comparado nas seguintes análises.

Tabela 2. Configurações das execuções

Parâmetro	Valor
População	1000
Gerações	500
Parcela da população que sofre mutação	60%
Taxa de mutação	0,5%
Inicialização	20% Heuristicamente
Método de Seleção	Torneio

[1]https://scipy.org/
[2]https://pandas.pydata.org/
[3]https://seaborn.pydata.org/
[4]https://matplotlib.org/

Nas execuções, as mutações são identificados pelos nomes a seguir:

- **original**: algoritmo de mutação original;
- **horários**: algoritmo de mutação de horários original;
- **horáriosCorr**: algoritmo de mutação de horários corrigido;
- **profAleCorr**: algoritmo de mutação de professores corrigido;
- **profSClash**: algoritmo de mutação de professores sem criar novos *clashes* **(novo)**;
- **horCRepComp**: algoritmo de mutação de horários com reparação de disciplinas compartilhadas **(novo)**.

Na Tabela 3, encontram-se os resultados dos testes de Shapiro-Wilk para cada um dos conflitos do tipo *hard* e para a pontuação de cada um dos operadores de mutação trabalhados. Todos os conflitos de indisponibilidade foram resolvidos na mutação profS-Clash+horCRepComp e na profSClash+horáriosCorr.

Tabela 3. Testes de normalidade de Shapiro-Wilk para cada variável analisada

Variável	Mutação	Estatística	Valor-p	Hipótese ($\alpha = 0,05$)
Pontuação	original	0,82140	0,00011	Anormal (H_1)
	profAleCorr+horários	0,80351	0,00005	Anormal (H_1)
	profSClash+horários	0,81331	0,00007	Anormal (H_1)
	profSClash+horCRepComp	0,84146	0,00027	Anormal (H_1)
	profSClash+horáriosCorr	0,86933	0,00112	Anormal (H_1)
	Conclusão:	Anormal		
Clash	original	0,96875	0,46567	Normal (H_0)
	profAleCorr+horários	0,95849	0,24955	Normal (H_0)
	profSClash+horários	0,92016	0,02100	Anormal (H_1)
	profSClash+horCRepComp	0,98082	0,82309	Normal (H_0)
	profSClash+horáriosCorr	0,98465	0,91757	Normal (H_0)
	Conclusão:	Anormal		
Compartilhamento	original	0,97339	0,59786	Normal (H_0)
	profAleCorr+horários	0,95182	0,16227	Normal (H_0)
	profSClash+horários	0,92124	0,02247	Anormal (H_1)
	profSClash+horCRepComp	0,97948	0,78422	Normal (H_0)
	profSClash+horáriosCorr	0,91833	0,01874	Anormal (H_1)
	Conclusão:	Anormal		
Indisponibilidade	original	0,93026	0,03981	Anormal (H_1)
	profAleCorr+horários	0,68699	$< 10^{-5}$	Anormal (H_1)
	profSClash+horários	0,61573	$< 10^{-5}$	Anormal (H_1)
	profSClash+horCRepComp	-	-	-
	profSClash+horáriosCorr	-	-	-
	Conclusão:	Anormal		

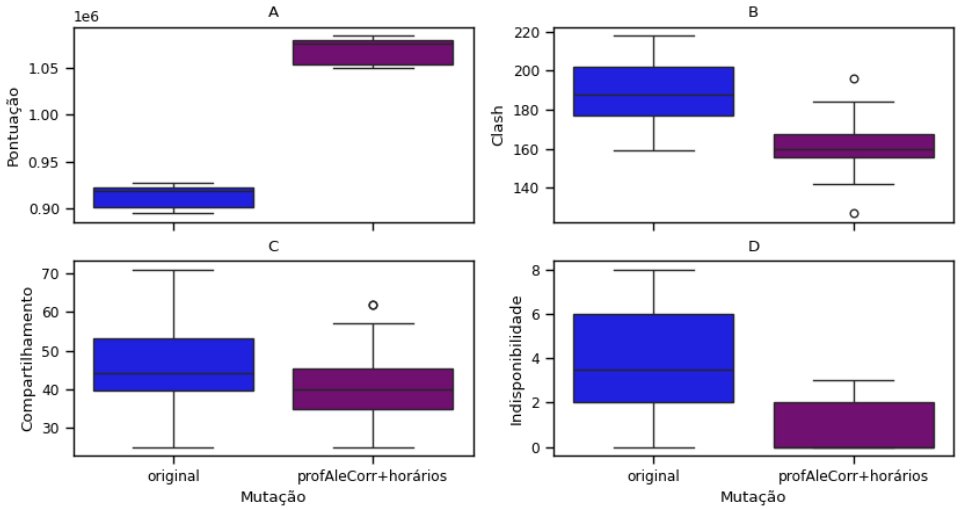
Ao nível de significância de 0,05, os testes revelaram que as distribuições das variáveis: pontuação, conflitos de *clash*, conflitos de compartilhamento e conflitos de indisponibilidade não seguiram uma distribuição normal em todas as execuções. Portanto, foram feitos testes estatísticos livres de distribuição para as comparações a seguir.

Foram feitos testes U de Mann-Whitney com a hipótese alternativa das distribuições serem diferentes, com correção de continuidade e de empates. Dessa forma, obtém-se evidência de que as distribuições são diferentes e, então, pode-se visualizar nos diagramas de caixas qual das duas obteve o melhor desempenho.

A mutação original (Algoritmo 2) é confrontada contra a sua versão equivalente: *profAleCorr+horários* (Algoritmo 5 modificado e Algoritmo 4), mas corrigida em relação a troca dos professores das disciplinas normais nos diagramas de caixa modificados da

pontuação (A), conflitos de *clashes* (B), conflitos de compartilhamento (C) e conflitos de indisponibilidade (D) presentes na Figura 1. Valores que estão a mais de 1,5 vezes a amplitude interquartílica do quartil superior ou que estão a menos disso do quartil inferior são considerados atípicos.

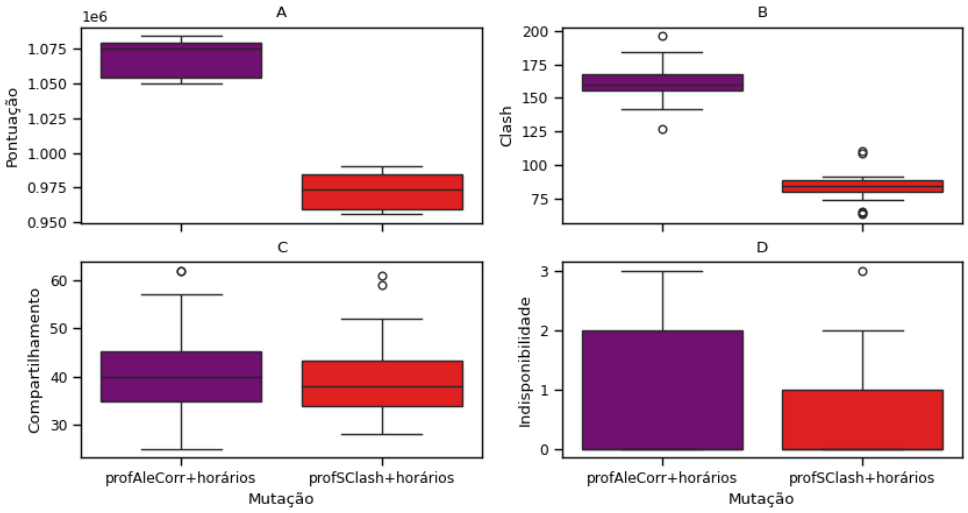
Figura 1. Diagramas de caixas modificados da pontuação, *clashes*, conflitos de compartilhamento e conflitos de indisponibilidade de orig x profAleCorr+horários



Percebe-se que a versão corrigida melhorou a pontuação (estatística U = 0,0 e valor-p $< 10^{-5}$), reduziu os conflitos de *clashes* (estatística U = 87,0 e valor-p $< 10^{-5}$) e de indisponibilidade (estatística U = 187,0 e valor-p $< 10^{-5}$) em relação a versão original. Portanto a *profAleCorr+horários* explorou melhor o espaço de busca e trouxe melhores soluções para o agendamento.

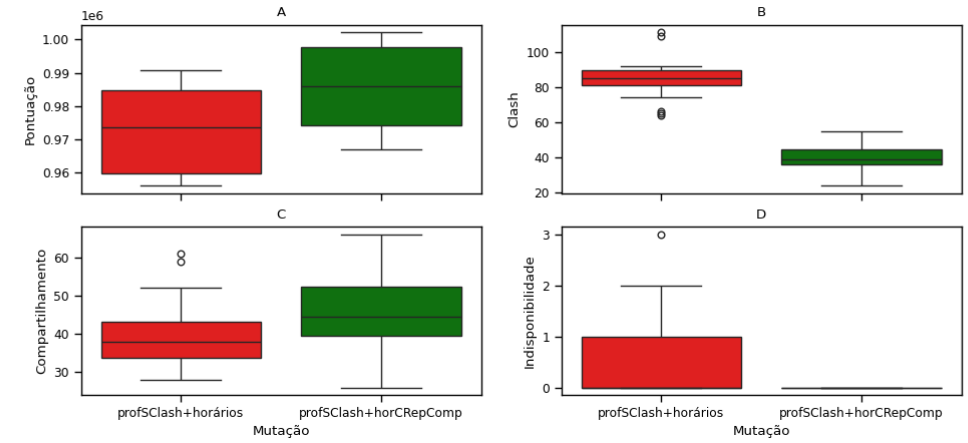
Ao confrontar a versão corrigida contra o mecanismo de avaliação adiante implementado para a escolha dos professores: *profSClash+horários* (Algoritmo 3 e Algoritmo 6), verifica-se que foi reduzido o total de conflitos de *clashes* (estatística U = 0,0 e valor-p $< 10^{-5}$). Contudo, a pontuação da versão corrigida ainda assim foi maior na Figura 2 (estatística U = 0,0 e valor-p $< 10^{-5}$). Isso se deve, provavelmente, a um ajuste melhor das *soft constraints*, mas a resolução dos conflitos do tipo *hard* importa mais para o agendamento.

Figura 2. Diagramas de caixas modificados da pontuação, *clashes*, conflitos de compartilhamento e conflitos de indisponibilidade de profAleCorr+horários x profSClash+horários



A eficácia do mecanismo reparador de *profSClash+horCRepComp* (Algoritmo 3 e Algoritmo 4) é averiguada nos resultados dos diagramas de caixas da Figura 3. A reparação aumentou a pontuação (estatística U = 256,0 e valor-p = 0,00060), resolveu mais conflitos de *clashes* (estatística U = 0,0 e valor-p $< 10^{-5}$) e anulou os conflitos de indisponibilidade, mas, inesperadamente piorou em relação aos conflitos de compartilhamento (estatística U = 345,0 e valor-p = 0,02525).

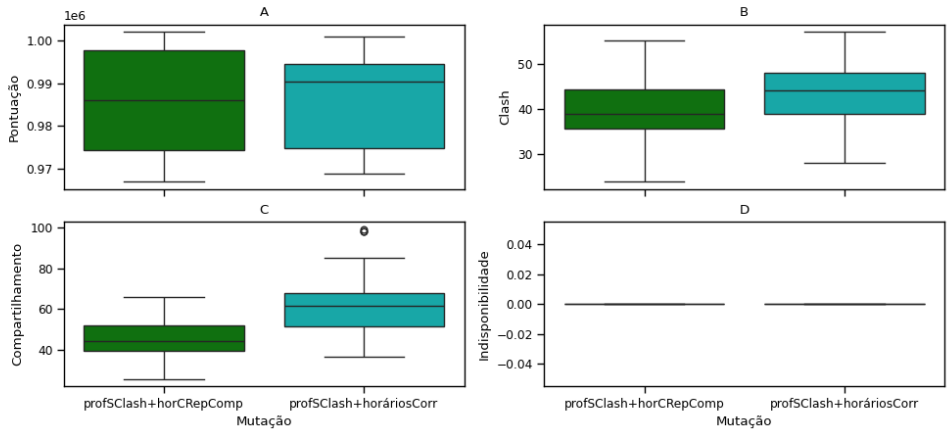
Figura 3. Diagramas de caixas modificados da pontuação, *clashes*, conflitos de compartilhamento e conflitos de indisponibilidade de profSClash+horários x profSClash+horCRepComp



Por fim, os resultados da versão derivada da *profSClash+horCRepComp*, mas sem a reparação: *profSClash+horáriosCorr* (Algoritmo 3 e Algoritmo 4 modificado) se en-

contram na Figura 4. Conclue-se que a reparação reduziu os *clashes* (estatística U = 333,0 e valor-p = 0,01634) e os conflitos de compartilhamento (estatística U = 148,5 e valor-p $< 10^{-5}$), Figura 4, B e C.

Figura 4. Diagramas de caixas modificados da pontuação, *clashes*, conflitos de compartilhamento e conflitos de indisponibilidade de profSClash+horCRepComp x profSClash+horáriosCorr



Na tabela 4, se encontram a média e a mediana para cada um dos conflitos discutidos e a média para os tempos de execução. Estão em destaque os melhores valores para as medidas de tendência central.

Tabela 4. Medidas resumo das amostras

Mutação	Pontuação		*Clash*		Compartilhamento		Indisponibilidade		Minutos
	Média	Mediana	Média	Mediana	Média	Mediana	Média	Mediana	Média
original	913792,536	919492,500	189,3	188,0	45,5	44,0	3,6	3,5	**29,32**
profSClash+horários	972669,774	973756,940	84,6	85,0	**39,4**	**38,0**	0,5	**0,0**	31,25
profSClash+horCRepComp	985920,875	986059,875	**39,4**	**39,0**	44,7	44,5	**0,0**	**0,0**	31,12
profSClash+horáriosCorr	986402,039	990299,155	43,9	44,0	63,3	61,5	**0,0**	**0,0**	31,17
profAleCorr+horários	**1068300,825**	**1075001,940**	161,9	160,0	41,4	40,0	0,8	**0,0**	29,68

Dentre todas, a mutação que obteve a melhor pontuação média e mediana amostral foi a profAleCorr+horários. Quanto a média e mediana amostral dos conflitos de *clash* foi a profSClash+horCRepComp, quanto aos compartilhamentos foi a profSClash+horários e quanto a indisponibilidade dos professores a profSClash+horCRepComp e a profSClash+horáriosCorr zeraram esta restrição. A mutação original foi a mais rápida, mas em comparação a mais lenta (profSClash+horários) tem aproximadamente uma diferença de tempo médio de dois minutos.

6. Considerações Finais

Recapitulando, foi discutido o sistema que gera o cronograma dos cursos universitários, duas novas mutações foram propostas e, depois, foram comparadas por meio de uma análise quantitativa das principais regras existentes.

Em suma, dentre todas as versões da mutação àquela que gerou um agendamento com menos conflitos do tipo *hard* foi a combinação dos Algoritmos 3 e 4: profSClash+horCRepComp. A mutação foi capaz de reduzir em aproximadamente 80% a média dos conflitos de *clash*, manteve a média de conflitos de compartilhamento, anulou os conflitos de indisponibilidade e aumentou em 8% a média da pontuação.

Entretanto, nenhum indivíduo gerado pelo algoritmo solucionou todos os conflitos do tipo *hard* para as configurações testadas, então o problema ainda não foi solucionado. Inclusive, certos indivíduos com menos conflitos do tipo *hard* apresentaram pontuações inferiores a outros indivíduos com um maior número destes conflitos, logo é sugerido aumentar o peso das *hard constraints* para que a pontuação represente melhor as soluções que são mais válidas.

Nos próximos estudos, pode-se, além de buscar outras formas de mutação que superem as que foram implementadas, encontrar o valor adequado para a taxa de mutação ou, talvez, implementar um operador de mutação que seja guiado pela aptidão dos indivíduos, como em [Libelli and Alba 2000]. Seria possível, também, verificar se escolher horários que não contém disciplinas compartilhadas, na linha 38 do Algoritmo 4, reduz o número de conflitos de compartilhamento em relação a mutação de horários original (Algoritmo 6).

Referências

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). An overview of curriculum-based course timetabling. TOP, 23:313–349.

Coppin, B. (2015). Inteligência Artificial. LTC, 1. edition.

Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing, 5:691–703.

Fabian, M. (2024). Solução de timetabling para universidade com foco na otimização para avaliação dos cursos pelo enade. pages 29–47.

Filho, O. (2023). Inteligência artificial e aprendizagem de máquina: aspectos teóricos e aplicações. Blucher, 1 edition.

Goldbarg, M., Goldbarg, E., and Luna, H. (2015). Otimização Combinatória e Meta-heurísticas - Algoritmos e Aplicações. Elsevier, 1 edition.

Gozali, A. A., Kurniawan, B., Weng, W., and Fujimura, S. (2020). Solving university course timetabling problem using localized island model genetic algorithm with dual dynamic migration policy. IEEJ Transactions on Electrical and Electronic Engineering, 15:389–400.

Libelli, S. M. and Alba, P. (2000). Adaptive mutation in genetic algorithms. Soft Computing, 4:76–80.

Mansour, N. and El-Jazzar, H. (2013). Curriculum based course timetabling. In 2013 Ninth International Conference on Natural Computation (ICNC), pages 787–792. IEEE.

Matias, J. B., Fajardo, A. C., and Medina, R. M. (2018). 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), pages 1–6. IEEE.

Raghavjee, R. and Pillay, N. (2012). A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. In 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), pages 98–103. IEEE.

Apêndices

A. Algoritmo da Mutação de Professores Original

Algoritmo 5 Mutação original - professores

```
1:  procedimento PROFESSORES(pMut, pComp)
2:      mutaComp ← Compartilhamento aleatório
3:      mutaComp ← mutaComp ∪ Sorteia compartilhamentos de acordo com pComp
4:      horáriosComp ← ∅
5:      para Todos os Cursos/Níveis faça
6:          para Todas as disciplinas faça
7:              d ← Disciplina atual
8:              p ← Professor de d
9:              extras ← Professores das demais turmas de d
10:             comp ← Código compartilhado de d
11:             se comp ∈ mutaComp então
12:                 se ∃horáriosComp[comp] então
13:                     novoPD ← horáriosComp[comp][0]
14:                     novosExtras ← horáriosComp[comp][1]
15:                 se não
16:                     novoPD ← Sorteia um professor apto
17:                     novosExtras ← Sorteia novos professores aptos para as outras turmas
18:                     horáriosComp[comp] ← (novoPD, novosExtras)
19:                 fim se
20:                 p ← novoPD
21:                 extras ← novosExtras
22:             fim se
23:         fim para
24:         para Todas as disciplinas sem horário faça
25:             d ← Disciplina atual
26:             se d foi sorteada por pMut então
27:                 p ← Professor de d
28:                 novoPD ← Sorteia um professor apto
29:                 p ← novoPD
30:             fim se
31:         fim para
32:     fim para
33: fim procedimento
```

B. Algoritmo da Mutação de Horários Original

Algoritmo 6 Mutação original - horários

1: **procedimento** HORÁRIOS($pComp$)
2: $mutaComp \leftarrow$ Compartilhamento aleatório
3: $mutaComp \leftarrow mutaComp \cup$ Sorteia compartilhamentos de acordo com $pComp$
4: *horáriosComp* $\leftarrow \emptyset$
5: $selecionado \leftarrow$ Seleciona um Curso/Nível
6: **para** Todos os Cursos/Níveis **faça**
7: *nível* $\leftarrow$ *(curso, nível)*
8: $disciplinas \leftarrow$ Embaralha todas as disciplinas de *nível*
9: **para** $\forall d_i \in disciplinas$ **faça**
10: $comp \leftarrow$ Código compartilhado de d_i
11: **se** $comp \in mutaComp$ OU *nível* $= selecionado$ **então**
12: **se** d_i é compartilhada **então**
13: **se** $\exists$*horáriosComp[comp]* **então**
14: $x \leftarrow$ *horáriosComp[comp]*
15: **se não**
16: $x \leftarrow$ Horário dentro dos horários comuns a todos os Cursos/Níveis do grupo compartilhado de $comp$
17: *horáriosComp[comp]* $\leftarrow x$
18: **fim se**
19: **se não**
20: $x \leftarrow$ Um horário livre que comporta d_i OU um horário de uma disciplina não compartilhada dentro dos horários de *nível*
21: **fim se**
22: $d_j \leftarrow$ Disciplina em x
23: Troque d_i e d_j
24: `Pare`
25: **fim se**
26: **fim para**
27: **fim para**
28: **fim procedimento**