Algoritmo Genético distribuído para problema de *Timetabling*

Iago R. Bianquini[1], **Osmar J. Hofman da Silva**[1], **Wilson Castello Branco Neto**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina
88506-400 – Lages – SC – Brasil

iagobianquini@gmail.com, hofman.osmar@gmail.com,

wilson.castello@ifsc.edu.br

Abstract. *This paper presents a Distributed Genetic Algorithm for solving the Timetabling problem of the Federal Institute of Santa Catarina Campus Lages, which aims to build a timetable without violating the restrictions imposed by the institution. To this end, an algorithm that can be run in a centralized or distributed environment was developed. Such algorithm has a pre-processing step which is responsible for reducing the search space, in addition to a depth-first search for the distributed environment, aimming to solve the remaining conflicts. It was verified that the algorithm reached, in both environments, the perfect solution and, in addition, the solution in the distributed environment reached the results in 26 seconds on average, while the centralized solution took 70 seconds.*

Resumo. *Este artigo apresenta um Algoritmo Genético Distribuído para resolução do problema de Timetabling do Instituto Federal de Santa Catarina Câmpus Lages, que visa gerar um quadro de horários sem violar as restrições impostas pela instituição. Para tal, foi elaborado um algoritmo que pode ser executado em ambiente centralizado ou distribuído, que conta com um pré-processamento que é responsável por diminuir o espaço de busca, além de um algoritmo de árvore de busca em profundidade limitada para o ambiente distribuído, com o objetivo de resolver os conflitos restantes. Foi constatado que o algoritmo alcançou, em ambos os ambientes, a solução perfeita e, além disso, a solução em ambiente distribuído chegou nos resultados, em média, 26 segundos, enquanto a centralizada levou 70 segundos.*

1. Introdução

A cada início de semestre em uma instituição de ensino, surge um problema recorrente: a elaboração dos quadros de horários de professores e estudantes. Esse problema é conhecido como *timetabling*, que segundo Schaerf (1999), consiste no agendamento de uma sequência de disciplinas e espaços físicos, como salas de aula e laboratórios, entre professores e estudantes num período de tempo previamente estabelecido, satisfazendo um conjunto de restrições. De acordo com Cruz et al. (2019), a elaboração desses horários causa um grande desgaste nos profissionais responsáveis por esta atividade, o que pode levar a erros causados por uma grande quantidade de variáveis, restrições e necessidades relacionadas aos recursos físicos e humanos. Essa complexidade é causada pelo fato de que não existe um algoritmo capaz de encontrar uma solução ótima em tempo polinomial para o problema de *timetabling*, o qual é categorizado como um problema NP-completo (Colorni et al., 1999).

Dentro desse contexto, as técnicas mais utilizadas em problemas de *timetabling*, são meta-heurísticas, tais como: *Greedy Randomized Adaptive Search Procedure* (GRASP), *Artifi-*

cial Bee Colony (ABC), busca tabu e Algoritmos Genéticos (AG). Esse último possui resultados expressivos em problemas de *timetabling*, sendo abordado nesse trabalho.

Algoritmos genéticos é uma subárea da computação evolucionária, que segundo Concilio (2000) é classificada como uma estratégia para resolver problemas genéricos, capaz de atuar em espaços não-lineares e não-estacionários. Apesar de não garantir o melhor resultado, geralmente chega-se em uma boa aproximação para a solução ótima em um tempo não-exponencial. A partir dessa afirmação, é possível identificar que para problemas NP-Completo, como o *timetabling*, AG são capazes de encontrar boas soluções em um tempo computacional viável.

Para Costa et al. (2010), pode-se definir algoritmos genéticos como uma função estocástica de busca global, que é baseada no princípio de funcionamento da teoria da evolução e seleção natural de populações. De acordo com Linden (2012):

> "Nos algoritmos genéticos, populações de indivíduos são criados e submetidos aos operadores genéticos de seleção, *crossover* e mutação. Estes operadores utilizam uma caracterização da qualidade de cada indivíduo como solução do problema em questão chamada de avaliação deste indivíduo e vão gerar um processo de evolução natural destes indivíduos, que eventualmente gerará um indivíduo que caracterizará uma boa solução (talvez até a melhor possível) para o nosso problema."

No Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina (IFSC), Câmpus Lages, o problema de *timetabling* também é uma realidade, pois a instituição utiliza um sistema terceirizado que não é capaz de gerar um horário completo, sem violar muitas restrições. Por consequência, é necessário a intervenção de servidores técnico-administrativos e coordenadores de curso para realizar ajustes nos horários gerados.

O objetivo deste trabalho é elaborar um AG para resolução de *timetabling* para o contexto do IFSC Câmpus Lages, visto que cada instituição de ensino tem suas próprias restrições e recursos, o que dificulta a reutilização de soluções já desenvolvidas ou genéricas.

Para alcançar este objetivo, foram traçados os seguintes objetivos específicos:

- Modelar um AG capaz de encontrar a solução para o problema de *timetabling* do IFSC Câmpus Lages;
- Implementar o AG modelado de diferentes maneiras, como *threads* e objetos distribuídos;
- Avaliar o desempenho das diferentes implementações do AG.

Visando melhorar o desempenho do processamento do AG foi implementado um módulo que tem o papel de processar os dados provenientes do IFSC, dividindo os cursos em conjuntos menores para posteriormente serem processados no AG. Após a conclusão dessa etapa, inicia-se de fato a execução do AG, que pode acontecer de duas formas, a primeira é realizando o processamento de maneira centralizada, assim o AG é executado em um único computador, gerando a solução final. A segunda forma de execução é o processamento distribuído, que utiliza-se dos conjuntos gerados no pré-processamento para enviá-los a diferentes computadores e assim tornar o processo mais rápido. Entretanto, ao final da execução de cada computador obtém-se uma solução para cada conjunto processado. Devido a isso, foi implementada uma etapa posterior a execução do AG que utiliza um algoritmo de árvore de busca em profundidade limitada para solucionar os possíveis conflitos gerados.

Este trabalho está divido em cinco seções. A primeira seção contém uma introdução ao tema e os objetivos do trabalho. Na segunda seção apresenta-se o referencial teórico sobre *timetabling* e alguns trabalhos similares. A terceira seção descreve a metodologia utilizada incluindo a apresentação da modelagem e implementação do AG. A seção 4 apresenta os resultados encontrados. Por fim, na quinta seção são descritas as considerações finais deste trabalho.

2. Trabalhos Relacionados

Na visão de Wren (1995), *timetabling* pode ser descrito como a alocação de recursos, sujeito a restrições, de certos objetos alocados no espaço-tempo, de maneira a satisfazer o máximo possível um conjunto de objetivos. Problemas de *timetabling* estão presentes em diferentes contextos, incluindo escalas de plantão médico, eventos esportivos, transporte e instituições educacionais (Burke et al., 2007). Ainda sobre a abordagem em escolas e universidades, este problema pode ser dividido em três classes principais (Schaerf, 1999):

- *Timetabling* de Escola: também conhecido como modelo turma/professor, no qual o agendamento é feito para todas as turmas de uma escola, evitando que o professor tenha duas turmas ao mesmo tempo. Este modelo é o mais utilizado em escolas de ensino fundamental e médio do Brasil;
- *Timetabling* de Curso: também conhecido como *timetabling* de cursos de universidade, que é o agendamento para todas as disciplinas de um conjunto de cursos da universidade, com o intuito de minimizar a sobreposição de diferentes matérias que tenham alunos em comum;
- *Timetabling* de Exame: agendamento de testes de um conjunto de cursos, com o objetivo de evitar que os alunos tenham duas provas em um mesmo horário, e minimizar o acúmulo de testes em um curto período de tempo.

De maneira geral, *timetabling* educacional possui dois tipos de restrições: *hard* e *soft*. Restrições *hard* devem necessariamente ser atendidas para que a solução seja viável. Algumas restrições *hard* comuns são: um professor não pode lecionar duas disciplinas em um mesmo horário, uma sala de aula não pode ser utilizada por duas turmas ao mesmo tempo e uma turma não pode ter duas disciplinas no mesmo horário. As retrições *soft* são definidas como desejáveis, porém não essenciais, como, por exemplo, um professor lecionar uma disciplina em um horário de sua preferência, ou não haver somente uma matéria durante um único turno. Os dois tipos de restrições são definidas pela instituição de ensino, levando em consideração suas necessidades.

Segundo Daskalaki e Birbas (2005), existem na literatura um número significativo de técnicas e abordagens utilizadas para a resolução de *timetabling*. Estes autores pontuam, ainda, que problemas pequenos podem ser resolvidos através de algoritmos exatos. Porém, conforme seu tamanho aumenta, a solução ótima fica mais distante de ser encontrada, devido à sua complexidade computacional. A dificuldade de resolução deste tipo de problema também está associada a discrepância de restrições, recursos e necessidades entre instituições. Em função disto, meta-heurísticas vem sendo aplicadas com o objetivo de encontrar uma solução satisfatória em um tempo aceitável.

GRASP é uma das técnicas utilizadas para resolver problemas de *timetabling*. De acordo com Resende e Ribeiro (2010), ela é constituída basicamente de duas fases: a construção e a busca local. A fase de construção gera uma solução viável, a fase de busca local investiga

sua vizinhança até que o mínimo local seja encontrado e a melhor solução geral é mantida como resultado.

O trabalho de Burke et al. (2009) apresenta um algoritmo híbrido baseado em GRASP, em conjunto com as heurísticas de baixo nível de grafos, *Saturation Degree* (SD) e *Largest Weighted Degree* (LDW), para um problema de *timetabling* de exame adaptativo. A heurística SD ordena os exames de acordo com os espaços de tempo, priorizando os mais restritivos, e a heurística LWD ordena os exames que possuem mais alunos com possíveis choques com outros exames. De maneira geral, o SD gera uma solução viável, porém não apresenta um bom desempenho nos estágios iniciais de construção, o que levou a utilização de LWD até um certo ponto dessa etapa.

O algoritmo foi aplicado nas duas primeiras versões do conjunto de dados para testes de desempenho da Universidade de Toronto, concluindo que o desempenho do SD em conjunto com LWD foi superior à execução individual dessas heurísticas. O algoritmo híbrido apresentou os melhores resultados, alcançando ao menos uma solução factível em 100% dos conjuntos de testes. Além disso, obteve um aproveitamento superior em relação aos algoritmos individuais, referente ao percentual de testes em que foi capaz de encontrar a solução em cada um dos conjuntos.

Outro tipo de técnica utilizada é a ABC. Essa técnica é baseada na estrutura de uma colônia de abelhas e tem como objetivo encontrar a fonte de alimento com maior qualidade, sendo essa fonte a provável solução do problema. Schiezaro e Pedrini (2013) classificam as abelhas em três tipos:

- Empregada: encontra uma possível fonte e armazena a informação de sua qualidade, além de compartilhar essas informações com as outras abelhas da colmeia;
- Oportunista: recebe a informação da empregada, e escolhe a fonte de alimento com maior qualidade para explorar a vizinhança, e no momento que escolhe essa fonte, torna-se uma abelha empregada;
- Exploradora: são abelhas empregadas que trocam sua função quando a fonte de alimento é exaurida, assim tentando encontrar novas fontes.

O trabalho de Bolaji et al. (2014) apresenta um algoritmo híbrido de ABC e um otimizador de subida de encosta (HCO), para um problema de *timetabling* de curso. O algoritmo é baseado na estrutura do ABC, a inicialização das fontes de alimento é feita através de LWD e *backtracking algorithm* (BA) e o processo de busca local é feito pelas empregadas que foi otimizado através do HCO. O testes foram realizados no conjunto de instâncias de Socha et al. (2002), com problemas de tamanho pequeno, médio e grande. Em comparação com vários outros algoritmos, apresentou os melhores resultados gerais, principalmente em problemas de médio e grande escala.

Na Tabela 1, o algoritmo ABC-Híbrido, produzido por Bolaji et al. (2014), é comparado com as outras técnicas. Os autores consideram necessário que nenhuma das restrições *hard* sejam violadas para uma solução factível ser encontrada, por isso o parâmetro utilizado para comparar o desempenho dos algoritmos foi a quantidade de restrições *soft* violadas, que devem ser minimizadas. Os valores em negrito representam os melhores resultados para determinada instância.

A técnica de busca tabu também é uma alternativa interessante. Segundo Edelkamp e Schrödl (2012), este é um algoritmo de busca local que restringe a vizinhança factível através

Tabela 1. Os melhores resultados alcançados pelo ABC-Híbrido e outras técnicas de inteligência de enxame.

Técnica	Pequeno 1	Médio 1	Médio 2	Grande
ABC-Híbrido	**0**	**73**	**79**	**462**
Sistema de Formiga MAX-MIN	1	195	184	851.5
Algoritmo de Otimização de Acasalamento de Abelhas	**0**	75	88	523
Abordagem Híbrida Evolucionária	**0**	221	147	529
Algoritmo Great Deluge não-linear Evolucionário	**0**	126	123	821
Algoritmo Great Deluge com Mecanismo de Eletromagnétismo	**0**	96	96	683
Formigas Cooperativas Obstinadas	5	176	154	798
Algoritmo Genético de Busca Guiada Extendida	**0**	139	92	615
Algoritmo Genético Guiado	**0**	240	160	801
Sistema de Formiga Elitista	**0**	84	82	690
Sistema Híbrido de Colônia de Formigas	**0**	117	121	647
Abordagem Metaheurística Híbrida	**0**	96	96	683

Fonte: adaptado Bolaji et al. (2014)

dos vizinhos que são excluídos. Para isto, existe uma estrutura de dados chamada lista tabu, que armazena os estados inalcançáveis, evitando que a busca fique presa em máximos locais. Caso todos os vizinhos estejam na lista tabu, é permitido um movimento que piore o valor da função objetivo.

O trabalho de Di Gaspero e Schaerf (2001) utiliza um algoritmo híbrido que combina busca tabu e busca tandem (mecanismo simples para combinar duas técnicas de busca local), para um problema de *timetabling* de exame, em diferentes conjuntos de dados da comunidade de *timetabling* de Toronto e Nottingham. Foram utilizados dois algoritmos baseados em tabu, o primeiro consiste em uma busca tabu simples utilizando uma lista completa de violações, e segundo é um algoritmo tandem, combinando duas buscas tabu. No algoritmo baseado em tandem, cada uma das buscas é chamada de *runner*, onde um *runner* foca em buscar qualquer tipo de melhoria e o segundo foca nas restrições *hard*, levando em conta somente os movimentos que as afetam.

Os valores apresentados na Tabela 2 representam a função objetivo a ser minimizada, que foi normalizada com base no número de violações por estudante, permitindo a comparação de resultados de instâncias com tamanhos diferentes. O algoritmo tandem de Di Gaspero e Schaerf (2001) foi comparado a dois algoritmos meméticos, produzidos por Burke e Newall (1999). Os autores concluem que a solução tandem obteve melhores resultados que MA2, e apesar dos resultados inferiores em comparação a MA2+D, esse tipo de abordagem baseada em decomposição geralmente é sensível às instâncias do problema.

Além dessas técnicas, AG podem ser considerados para a busca de uma solução viável. De acordo com Montana et al. (1998), a razão de AG ser um sucesso e ter uma vasta utilização para problema de escalonamento deve-se a combinação de poder e flexibilidade. O poder deriva da prova empírica que algoritmos evolucionários encontram de maneira eficiente a solução ótima em espaços de pesquisa grandes e complexos, característica típica dos problemas do mundo real. A flexibilidade de AG tem múltiplas facetas, já que podem efetivamente tratar problemas que muitos algoritmos de otimização tradicionais não conseguem, como espaços discretos e não-lineares.

Tabela 2. Comparação com resultados de Burke e Newall (1999).

Conjunto de Dados	Exames	Espaço de Tempo	Solução Tandem		Burke and Newall	
			Melhor	Média	MA2	MA2+D
CAR-F-92	543	36	3048	3377	12167	1765
KFU-S-93	461	21	2135	2825	3883	1608
NOTT	800	23	751	810	1168	736
PUR-S-93	2419	30	123935	126046	219371	65461

Fonte: adaptado Di Gaspero e Schaerf (2001)

O trabalho de Sutar e Bichkar (2017) é direcionado ao *timetabling* escolar. Ele utiliza um algoritmo híbrido baseado em AG, combinado com busca tabu, como uma solução para o hdtt4, que é um conjuntos de dados para *timetabling* presente em uma coleção de conjunto de dados de testes para pesquisa operacional, chamada *OR-Library*. O funcionamento do algoritmo é baseado na estrutura fundamental do AG, a inicialização da população é feita através de busca tabu e a mutação e *crossover* foram modificados a fim de satisfazer os requisitos de carga de trabalho. Os autores concluíram que a solução produzida satisfez todas as restrições para o problema, gerando o melhor resultado em 7 segundos, enquanto um AG simples alcançou esse resultado em 12 segundos. Além disso, técnicas baseadas em redes neurais, têmpera simulada, busca tabu e gulosa, obtiveram seus resultados em tempos ainda maiores.

O trabalho de Alves et al. (2017) apresenta um AG recursivo para resolver o problema de *timetabling* de curso da Universidade Federal do Ceará. Além da recursividade, o AG tem como objetivo ser escalável e parametrizado a fim de encontrar uma solução factível para uma maior gama de cursos. O AG recebe como parâmetros a lista de cursos, a indisponibilidade dos agentes - que incluem professores e disciplinas, e os parâmetros do AG, como as taxas de mutação e cruzamento. Em cada execução do AG, busca-se por uma solução factível para um curso na lista, e uma vez que a solução é encontrada, essa lista é atualizada criando uma nova designação para o agente. Essa designação é utilizada para atualizar a indisponibilidade dos agentes para a próxima execução. Esse processo é repetido até ser encontrada a solução geral do problema. Os autores ressaltaram que houve dificuldade em comparar os resultados com outros presentes na literatura, devido à complexidade das divergências do ambiente. No entanto, os trabalhos de Borges (2007) e Ramos (2002) apresentam diferenças mínimas em classes, agentes e restrições, que possibilita a comparação. Os autores concluíram que na maioria das tentativas, o modelo encontrou a solução factível e também gerou mais de uma solução possível.

Como mostra a Tabela 3, os resultados indicaram que o algoritmo de Alves et al. (2017) levou menos tempo que os outros, mesmo assim, vale ressaltar a divergência da modelagem e parâmetros entre os trabalhos. Os autores também reforçam que quanto maior a população, mais custosa é a execução, assim como o número de gerações em relação ao tempo de execução.

O trabalho de Nourmohammadi-Khiarak et al. (2017) utiliza métodos de controle multiagente combinado com AG para resolver um problema de *timetabling*, focando no gerenciamento de eventos interdepartamentais dos cursos da universidade de Tabriz, no Irã. Cada departamento é representado como um agente, com procedimentos e restrições únicas. O *surveyor agent*, extrai eventos em comum entre esses múltiplos departamentos associados, com suas respectivas restrições e recursos adicionais disponíveis. Por fim, o *interface agent* obtém

Tabela 3. Comparação com outros trabalhos similares.

Parâmetros	Alves	Borges	Ramos
Número de Classes	16	8	14
Número de Agentes	21	33	21
Número de Salas	x	x	10
Tamanho da População	75	1000	100
Operador de Crossover	OX	PMX	Ponto Único
Taxa de Crossover	50%	50%	70%
Taxa de Mutação	1/25	1/20	0,002
Estratégia de Solução	Aleatório	Roleta	Torneio
Elitismo	x	10%	x
Monitor de estagnação	x	Sim	x
Gerações	493	201	≈15000
Tempo de Execução (em minutos)	≈3,7	x	35-90
Aptidão	1	1	0.5

Fonte: adaptado Alves et al. (2017)

as informações do *surveyor agent* e utiliza um algoritmo competitivo imperialista [1]para retirar interferências e aumentar a satisfação dos eventos em comum.

Dentre os resultados obtidos pelo trabalho, pode-se destacar dois pontos: 1) a dissipação de excesso de recursos em cada departamento foi minimizada, o que representa uma melhora na alocação de eventos em comum; 2) a abordagem proposta organizou prioridades de forma decrescente sobre os eventos comuns entre os departamentos, para a alocação de recursos em excesso. Além disso, métodos baseados em sistema multiagente levam a uma melhora da independência do *timetabling* de cada departamento.

Um trabalho que tem como objetivo implementar novas abordagens para o uso de AG na nuvem, utilizando *softwares* de conteinerização e também *web services*, é descrito por Salza e Ferrucci (2018). A aplicação adota a arquitetura mestre/escravo, sendo o nodo mestre responsável pela execução do AG, exceto a função de avaliação que é delegada aos nodos escravos. A comunicação entre esses nodos é feita através do *RabbitMQ*, um software *open source* utilizado para intermediar mensagens. Cada escravo atua sob o *CoreOS*, que é um sistema operacional focado em ambientes distribuídos.

Quanto aos resultados, segundo os autores, foi possível acelerar a execução do AG, com um total de 128 escravos. Notou-se, também, uma grande dependência entre a carga computacional e o custo de comunicação, ou seja, quanto maior a quantidade de informação associada ao cromossomo, maior é o custo da troca de mensagens entre os nodos. O desempenho e o tempo de configuração posicionam a abordagem em nuvem positivamente entre outras tecnologias empregadas à paralelização de AG disponíveis na literatura.

Os trabalhos citados nesta seção possibilitaram a compreensão das definições gerais das técnicas, conceitos, diferentes formas de resolução e tipos de problemas de *timetabling*.

[1]É um algoritmo que modela os indivíduos da população como imperialistas e colônias, baseado na competição imperialista, o qual impérios fracos colapsam e os mais fortes dominam mais colônias, com o objetivo final de gerar uma estado que haja somente um império.

Embora não seja possível uma comparação quantitativa dos seus resultados, em função das divergências dos ambientes citada por Alves et al. (2017), eles contribuíram para o entendimento sobre como a otimização do processamento geral pode ser feita a partir de pontos específicos e como a divisão do processo em partes menores pode impactar em um ganho de *performance*.

É visível dentre os trabalhos apresentados que a combinação de diferentes técnicas para buscas locais apresentam resultados mais satisfatórios do que quando comparados a técnicas puras, como mostra Burke et al. (2009), Bolaji et al. (2014) e outros. A priorização de alocação dos elementos mais restritivos primeiro, como feita por Burke et al. (2009), e a resolução do problema em etapas, como demonstrada por Alves et al. (2017), foram elementos importantes para a concepção da solução proposta neste artigo. Além disso, os trabalhos citados validam a utilização de AG como uma técnica viável para o problema abordado nesse artigo, como demonstra Montana et al. (1998) e Sutar e Bichkar (2017), apresentando flexibilidade na escolha da arquitetura de implementação e a possibilidade de utilização de agentes distribuídos, como destaca Salza e Ferrucci (2018).

3. Materiais e Métodos

Este trabalho é de natureza aplicada, pois objetiva o desenvolvimento de conhecimento para uma aplicação prática, que visa a resolução de um problema de *timetabling* escolar. Para a abordagem do problema, a análise das informações é quantitativa, a partir de recursos e técnicas estatísticas. Do ponto de vista dos objetivos, apresenta um viés exploratório, aprofundando-se no problema para apresentar hipóteses sobre o mesmo, através do estudo de outros exemplos. Livros, artigos de periódicos e materiais disponíveis da Internet, fundamentaram os procedimentos técnicos, o que caracteriza uma pesquisa bibliográfica. Além disto, adotou-se uma abordagem experimental, a fim de definir a estratégia com desempenho mais significativo em relação ao AG, e um estudo de caso com os dados disponibilizados pelo IFSC foi realizado.

O levantamento sobre os temas AG, *timetabling* e trabalhos relacionados foi desenvolvido por meio da pesquisa em sites acadêmicos como *google scholar, scielo* e *sciencedirect*.

Após essa etapa, a modelagem do AG foi realizada tomando como base o estudo de caso dos trabalhos de Borges (2007) e da Silva (2010), a qual foi adaptada para as restrições do IFSC. A principal mudança é a flexibilização do número de aulas que determinada disciplina pode ter, possibilitando que as disciplinas tenham duas ou quatro aulas em um dia.

Na sequência, para a realização dos testes de desempenho do sistema, foi criado um ambiente para simular alguns cenários, como programação centralizada, com e sem a utilização de *threads*, e programação distribuída. Posteriormente, foi implementado o AG, de acordo com o método que gerou os melhores resultados nos testes. Para esse desenvolvimento foi utilizada a linguagem de programação *Java*, aliada ao *framework Spring* para o *backend*. Uma aplicação *Web* foi desenvolvida em relação ao *frontend* e para melhor integração de ambos, foi utilizado o padrão de projeto *Model-View-Controller* (MVC).

Para testar os resultados obtidos com o algoritmo desenvolvido na etapa anterior, dados de semestres anteriores do IFSC foram utilizados. Um destes testes, avaliou a influência de parâmetros como taxa de mutação, cruzamento, elitismo e tamanho da população nos resultados do AG, a partir da análise de variância e de testes de comparação de médias. Já o outro teste, calculou a qualidade dos resultados e o tempo médio de execução, por meio do valor da média e desvio padrão, uma vez que AGs usam valores aleatórios em sua execução e podem gerar

resultados muito diferentes para os mesmos dados de entrada, em diferentes execuções.

As subseções seguintes tratam da leitura dos dados, o pré-processamento dos dados, a modelagem e implementação do AG.

3.1. Leitura dos Dados

O sistema utiliza um conjunto de dados gerado por um sistema do IFSC Câmpus Lages que está disponível através de um arquivo no padrão XML. A partir desse conjunto é gerada uma estrutura equivalente mais concisa, já que os dados provenientes do IFSC estão organizados de maneira que todas as informações estão fortemente acopladas a uma classe que é utilizada somente como uma tabela de associação, fazendo-se necessário a consulta a múltiplas classes para obtenção de dados, o que impacta diretamente no processamento do algoritmo.

A Figura 1 apresenta a estrutura dos dados presentes no XML do IFSC e a Figura 2 a estrutura da Competição Internacional de *Timetabling* (ITC) de 2007[2]. Na sequência, a Figura 3 representa o modelo adaptado utilizado pelo algoritmo, que tem como base a estrutura do ITC, visando trabalhos futuros, nos quais pretende-se testar o sistema desenvolvido com os conjuntos de testes disponibilizados nesta competição.

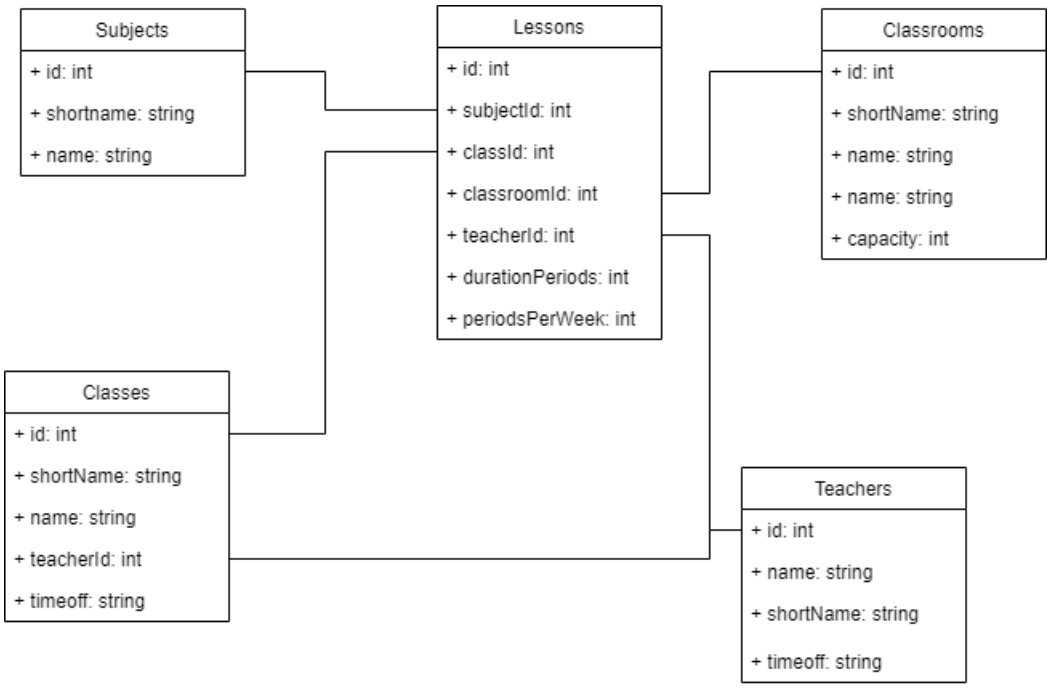


Figura 1. Relação do modelo do IFSC

As principais diferenças entre esses modelos são:

- No modelo adaptado as restrições estão representadas em uma classe separada chamada *Unavailability Constraints* e estão ligadas somente a classe *Lesson*, ou seja, as restrições

[2]A Competição Internacional de *Timetabling* (ITC) de 2007 foi um evento organizado por um grupo de pesquisa da Universidade do *Queen's* em conjunto de outras universidades, composta por três trilhas com diferentes problemas de *timetabling* educacional. O objetivo do ITC era reunir pessoas de diferentes áreas, a fim de encontrar novas abordagens e também melhorar a qualidade das pesquisas na área de *timetabling*. Os dados presentes nessa competição ainda são utilizados como referência para comparação de soluções de *timetabling* até hoje.

estão associadas as disciplinas de um curso. O modelo do IFSC não possui uma classe específica sobre essas restrições;
- No modelo adaptado a informação da entidade do professor está incluída como um atributo na classe *Lesson*. Já no modelo do IFSC existe uma classe para representar essa entidade;
- Alguns atributos da classe *Lesson* do modelo adaptado são obtidos dos identificadores das classes *Subjects* e *Classes* do modelo do IFSC;
- A classe *Course* do modelo adaptado representa um único curso com suas respectivas disciplinas. No modelo do IFSC não há uma classe com esta função, assim esses dados são obtidos através do cruzamento de informações da classe *Lessons*.

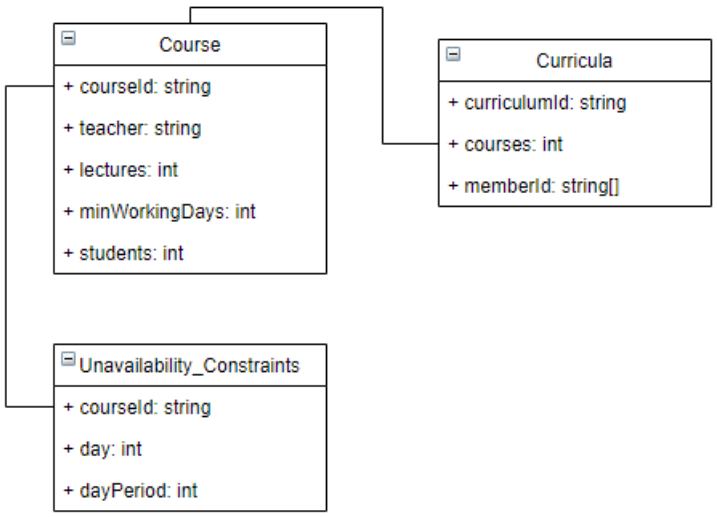


Figura 2. Relação do modelo do ITC

Visando compatibilizar os dois modelos, foram realizados alguns ajustes, sendo eles:

- Na classe *Lesson* do modelo adaptado, o atributo *minWorkingDays* é obtido através da divisão do atributo *periodsPerWeek* por *durationPeriods*, referentes ao modelo do IFSC;

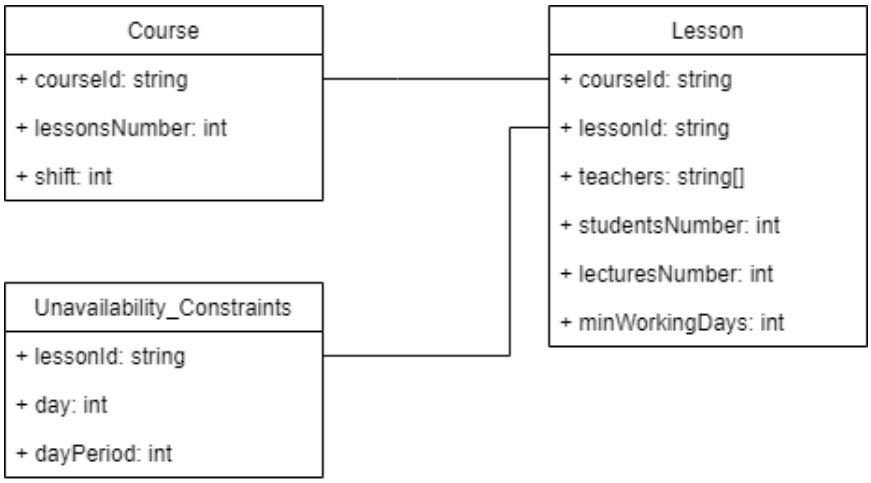


Figura 3. Relação do modelo adaptado

- O atributo *studentsNumber* da classe *Lesson* não possui correspondente no pelo modelo do IFSC, assim foi atribuído um valor constante a ele que não interfere na função de avaliação do AG;
- Na classe *Course* o atributo *shift* é obtido através da análise do atributo *timeoff* da classe *Classes*;
- Na classe *Unavailability Constraints* do modelo adaptado, os atributos *day* e *dayPeriod* são obtidos através da análise do atributo *timeoff* da classe *Teachers*, referente ao modelo do IFSC.

3.2. Modelagem do Algoritmo Genético

O objetivo do AG é gerar horários para os cursos que não violem as seguintes restrições:

- Turmas com duas aulas no mesmo período;
- Turmas com aulas de outras turmas;
- Disciplinas com aulas excedentes ou faltantes;
- Conflito de horários entre professores;
- Indisponibilidade dos professores.

As três primeiras restrições foram resolvidas durante a modelagem do AG e a implementação de seus operadores, como é explicado no decorrer desta seção. Desta forma, a função de avaliação não precisa verificar a existência das mesmas, o que representa um ganho de tempo na execução do AG.

As únicas duas restrições que podem ocorrer com a modelagem e operadores adotados e que precisam ser verificadas pela função de avaliação são:

- Conflito de horários: Essa restrição descreve os casos em que um professor ministra duas disciplinas no mesmo período letivo e é classificada como uma restrição *hard*, ou seja, o horário torna-se inviável caso ocorra;
- Indisponibilidade dos professores: Representa os cenários em que uma aula é atribuída à um professor num período que não pode lecionar. A classificação dessa restrição é *soft*, já que caso ocorra, o horário não será inviável, porém é desejável reduzir o seu número de ocorrências;

O cromossomo é uma estrutura de dados que tem a capacidade de representar a solução final do problema, que usualmente é representada por um vetor.

Neste trabalho, o tamanho do cromossomo é dado pela equação 1 :

$$tamanho = n * (q/2) \tag{1}$$

Na equação 1, n representa o número de turmas do câmpus e q a quantidade de aulas das turmas por semana. Essa quantidade é dividida por dois porque cada gene do cromossomo representa duas aulas consecutivas, pois esta é situação mais comum no IFSC. Desta forma, o AG precisa alocar duas aulas por turno para cada turma, o que reduz significativamente o espaço de busca, se comparado à uma solução modelada com quatro aulas individuais.

A Figura 4 representa a estrutura das aulas presentes no cromossomo. Essa estrutura contém as informações dos códigos das disciplinas, seus nomes e carga horária semanal. Como mostra a Figura 5, cada turma é representada por dez posições do cromossomo, e tais posições

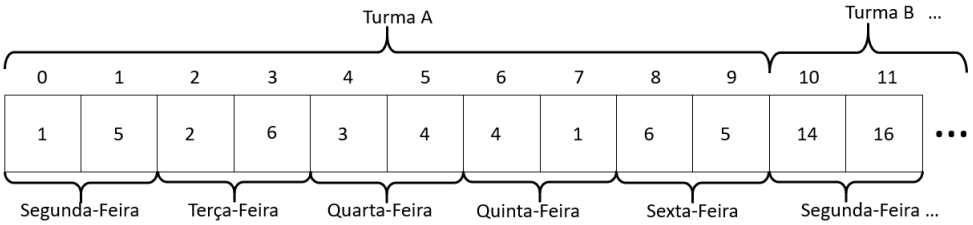
ID	Disciplina	Carga Horária (Horas semanais)
1	Cálculo Numérico	4
2	Grafos	2
3	Informática e Sociedade	2
4	Fundamentos de Banco de Dados	4
5	Introdução a Engenharia de Software	4
6	Teoria da Computação	4

Figura 4. Representação das disciplinas de uma turma

simbolizam o período e o dia da semana de cada uma das aulas. Por exemplo, os valores das posições 0 e 1 (1 e 5) representam, respectivamente, os códigos das aulas de cálculo numérico e introdução a engenharia de software, de acordo com a Figura 4.

Como citado anteriormente, cada gene representa duas aulas, pois esta é a situação da maioria dos casos no IFSC, porém existem cursos que não se encaixam nesse padrão, tendo disciplinas com quantidade ímpar de aulas semanais. Nestes casos, as aulas ímpares de cada disciplina são agrupadas para formar uma única aula par. Assim, como ocorre atualmente, esta aula representa que uma das disciplinas terá duas aulas na primeira metade do semestre e a outra disciplina duas aulas na segunda metade.

O processamento do AG segue o modelo apresentado na Figura 6 e todos os parâmetros citados nas próximas etapas foram ajustados durante a fase de testes. As etapas do processamento do AG são: inicialização, avaliação, seleção, cruzamento e mutação.



	Segunda-Feira	Terça-Feira	Quarta-Feira	Quinta-Feira	Sexta-Feira
Período 1	1	2	3	4	6
Período 2	1	2	3	4	6
Período 3	5	6	4	1	5
Período 4	5	6	4	1	5

Figura 5. Representação da estrutura do cromossomo

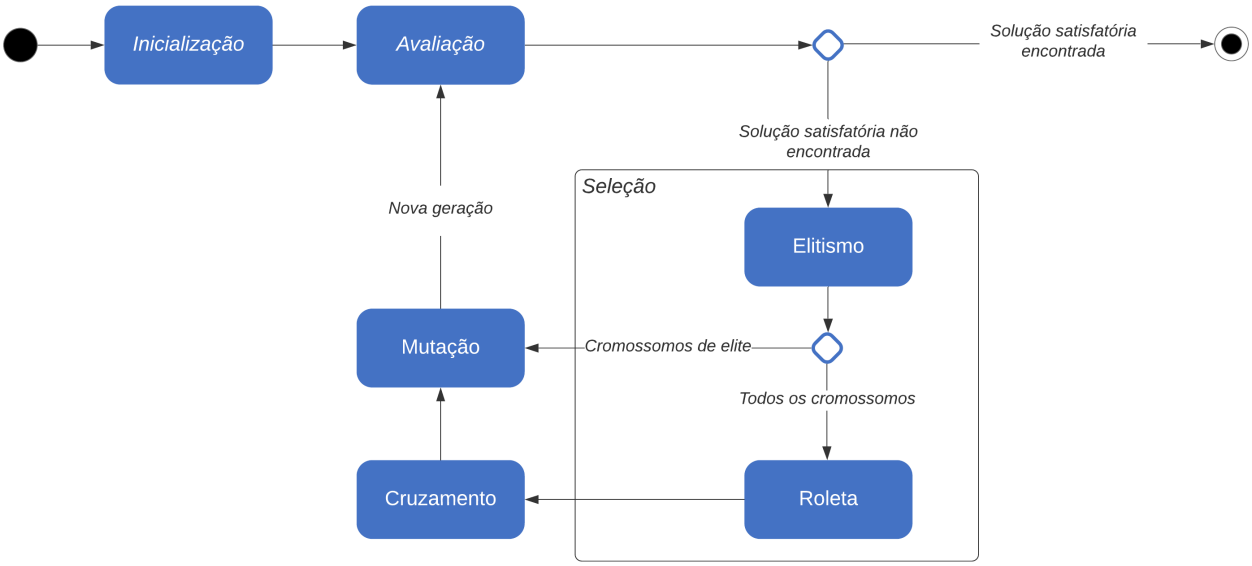


Figura 6. Diagrama de atividades do Algoritmo Genético

A etapa de inicialização tem como objetivo preencher as posições do cromossomo com valores aleatórios de acordo com as disciplinas de cada curso. Como cada cromossomo representa uma solução para o problema, a inicialização dos mesmos com valores aleatórios leva à geração de diferentes soluções, fornecendo a diversidade genética necessária para que as melhores partes de cada solução sejam combinadas ao longo do processo, até que a solução seja encontrada.

O processo consiste em percorrer cada turma e obter todas as disciplinas relacionadas a ela. A partir dessa informação, no intervalo de posições pertencentes à turma, são inseridos os identificadores dessas disciplinas e, após isso, são feitas trocas entre os identificadores em posições aleatórias desse intervalo. Tal processo é feito a fim de garantir que não existam disciplinas em excesso ou em falta em uma turma, assim como evitar que disciplinas sejam atribuídas a outras turmas.

A avaliação quantifica a qualidade da solução de cada cromossomo da população e também verifica se o resultado ideal foi alcançado. Como citado no início desta seção, há duas restrições presentes nesse processo, a primeira diz respeito aos conflitos de horários de professores, ou seja, períodos em que um mesmo professor leciona para mais de uma turma (*hard constraint*). A segunda restrição refere-se a indisponibilidade dos professores, quando é verificado se os professores lecionam em dias e horários que não gostariam (*soft constraint*). Cada cromossomo é inicializado com uma pontuação pré-definida, e tal pontuação é decrementada a cada restrição violada, ou seja, quanto maior a pontuação, melhor a avaliação do cromossomo. Conforme mencionado anteriormente, os problemas de turmas com duas disciplinas no mesmo horário ou com matérias faltantes não precisam ser verificados, pois a modelagem do cromossomo e inicialização evitam essas situações. Da mesma forma, o problema de turmas com matérias de outras turmas não precisa ser verificado pela função de avaliação, devido à forma como os operadores de cruzamento e mutação foram modelados, reduzindo o tempo de processamento da mesma.

Na seleção é feita a escolha dos cromossomos que são enviados para a próxima geração do processamento do AG. Essa escolha é feita por meio de dois sub-processos, levando em conta um parâmetro que indica a proporção da população que deve ser selecionada em cada um deles. Os sub-processos citados são:

- Elitismo: No elitismo os cromossomos melhores avaliados são selecionados de forma determinística e enviados diretamente para a etapa de mutação e, posteriormente, para a nova geração, sem passar pela etapa de cruzamento.
- Roleta: A roleta seleciona aleatoriamente pares de pais (cromossomos utilizados para gerar novos cromossomos), que são escolhidos a partir de valores gerados aleatoriamente, para a próxima etapa. A probabilidade de um cromossomo ser selecionado é diretamente proporcional a sua qualidade.

Na etapa de cruzamento, os pares de cromossomos gerados na roleta são usados para gerar novos cromossomos a fim de criar uma nova geração. Para o cruzamento foi utilizado o algoritmo descrito por Luger (2013), denominado *Recombinação Ordenada*. Neste algoritmo, a construção dos descendentes é feita a partir de uma subsequência de valores de um dos genitores, preservando também a ordem dos valores do outro genitor.

O processamento ocorre da seguinte forma: são gerados dois números aleatórios para definir os pontos de corte (3 e 7 no exemplo da Figura 7). Após, os valores entre esses pontos são copiados dos genitores (P1 e P2) e inseridos nos seus descendentes (C1 e C2) nas mesmas posições. A seguir, para se obter o primeiro conjunto (R1) obtém-se os valores a partir do

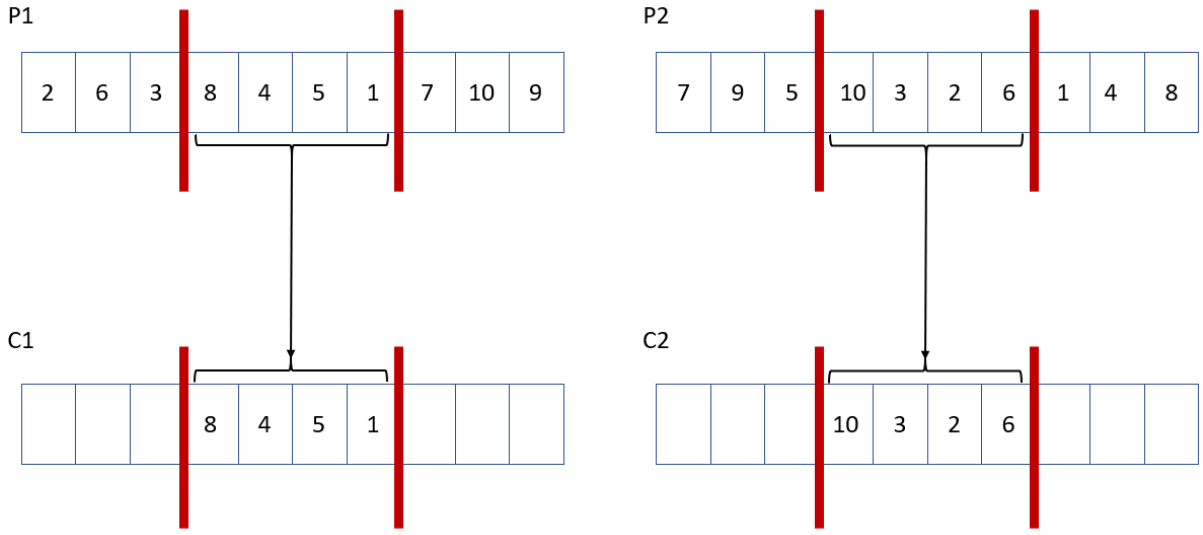


Figura 7. Exemplo de cruzamento parte 1

segundo ponto de corte do genitor P2. Quando o final do cromossomo é atingido deve-se voltar ao início formando, neste exemplo, a sequência $1|4|8|7|9|5|10|3|2|6$. Em seguida, são removidos dessa sequência os valores presentes entre os pontos de corte do P1, resultando em $7|9|10|3|2|6$, como mostra a Figura 8.

O mesmo processo é feito para definir o segundo conjunto (R2). Por fim, na Figura 9, os valores obtidos nos conjuntos R1 e R2 são inseridos nos descendentes C1 e C2 respectivamente, preservando os valores previamente inseridos.

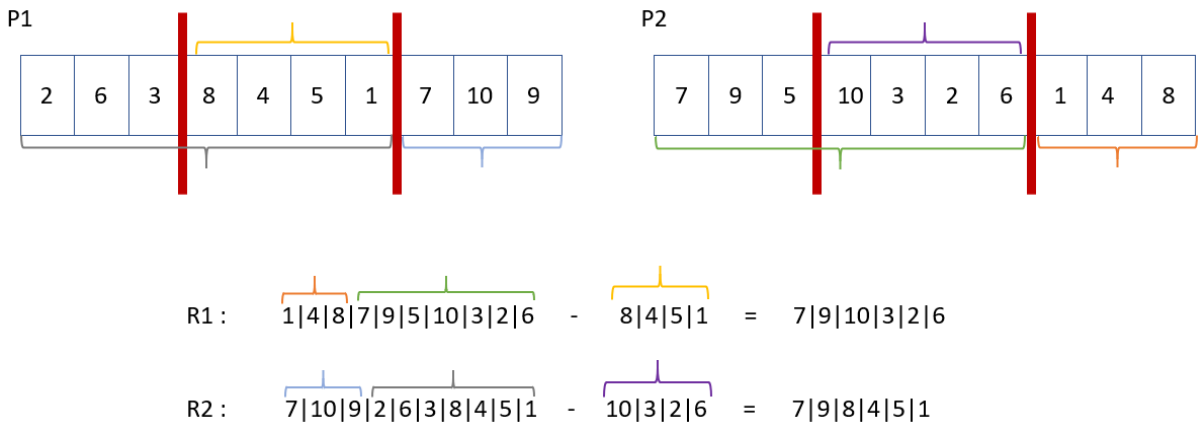


Figura 8. Exemplo de Cruzamento parte 2

Na mutação é realizada uma alteração de dois genes de forma aleatória a partir de um parâmetro, que representa a porcentagem de chance dessa etapa ocorrer. Isso é feito através da troca dos valores desses genes em posições aleatórias do cromossomo dentro de uma mesma turma, da mesma forma como na randomização dos elementos da etapa de inicialização, porém essa troca é feita somente uma vez.

Ao final desse processamento, cria-se uma nova geração a partir dos cromossomos que foram selecionados, cruzados e modificados e reinicia-se o processo até que se encontre uma solução satisfatória ou atinja um número limite de iterações.

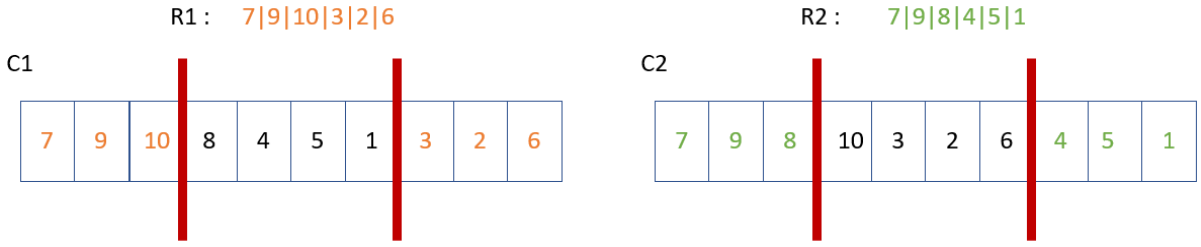


Figura 9. Exemplo de Cruzamento parte 3

3.3. Implementação

O Algoritmo Genético desenvolvido pode ser utilizado para a elaboração dos horários de um único curso ou de diversos cursos juntos. A segunda opção é mais custosa em termos de tempo de processamento, pois o espaço de busca cresce de forma exponencial com o aumento do número de cursos. No entanto, uma vez que um resultado é encontrado, ele representa a solução final do problema. A elaboração dos horários de cursos individuais é mais simples e rápida devido ao espaço de busca reduzido e possibilita o processamento em ambiente distribuído. Porém, esse tipo de solução pode gerar conflitos nos horários de aula de professores que ministram aulas em mais de um curso. Neste artigo, uma solução intermediária foi elaborada visando aproveitar as melhores características dessas duas abordagens, através da criação de conjuntos que agrupam os cursos com muitos professores em comum. Esta abordagem apresenta um espaço de busca menor que o existente quando se tenta elaborar os horários de todos

os cursos juntos, e minimiza o número de conflitos entre os professores que ministram aulas em diversos cursos.

O Algoritmo 1 apresenta todas etapas necessárias para elaborar os horários, independente da abordagem escolhida e da forma de execução do AG (centralizada e distribuída).

Os dados necessários para a execução do algoritmo são:

- Arquivo de configuração do AG: arquivo que contém os parâmetros dos operadores do AG, como porcentagem de elitismo, mutação, etc.;
- XML com dados do IFSC: arquivo que possui todos as informações do domínio da aplicação, como dados dos professores, aulas, turmas, etc.;
- Arquivo com IPs dos computadores: arquivo com o endereço IP de cada máquina disponível para processar o AG.

A execução do Algoritmo 1 é iniciada pela leitura de cada um dos dados dos arquivos providos como entrada, como apresentado nas linhas 1, 2 e 3. Na linha 4, os dados do IFSC são convertidos para o modelo adaptado, que são efetivamente utilizados no decorrer do processamento do AG. A partir desses dados, na linha 5, é feito o pré-processamento como descrito na seção 3.2, e como resultado são obtidos os conjuntos que o AG processa de forma individual.

Algoritmo 1: Algorítmo de processamento distribuído do AG

Entrada: Arquivo de configuração do AG, XML com dados do IFSC, Arquivo com IPs dos computadores

início

```
1   configuracoesAG ← leArquivoConfiguracoes(Arquivo de configuração do AG);
2   dadosIFSC ← leDadosIFSC(XML com dados do IFSC);
3   listaIPsComputadores[] ← leArquivoIPs(Arquivo com IPs dos computadores);
4   modeloAdaptado ← converteParaModeloAdaptado(dadosIFSC);
5   conjuntos[] ← preProcessamento(modeloAdaptado);
6   cromossomos[conjuntos.tamanho()];
7   para IPAtual ← listaIPsComputadores ate ultimoIP faça
8       conjuntosComputador ← obterConjuntosDoComputador(conjuntos, IPAtual);
9       listaMelhoresCromossomos ← processamentoDoAG(conjuntosComputador, modeloAdaptado, configuracoesAG);
10      cromossomos.adicionaTodosCromossomos(listaMelhoresCromossomos);
    fim
11  retorna cromossomos;
```

fim

Saída: Lista com os melhores cromossomos gerados pelo AG

Logo após, os conjuntos são atribuídos aos computadores disponíveis de forma a distribuir a quantidade de conjuntos por máquina, que é dada pela divisão simples do número de conjuntos pela quantidade de computadores, como demonstrado na linha 8. Por exemplo, caso sejam gerados cinco conjuntos e existam dois computadores disponíveis, os conjuntos são distribuídos de forma que um computador processe três conjuntos e o outro processe os dois conjuntos restantes. Essa distribuição é possível porque os computadores possuem configurações

iguais e a implementação de um algoritmo que realize o balanceamento de carga considerando fatores como o tamanho e a complexidade dos conjuntos fica em aberto para trabalhos futuros.

Na linha 9, é enviada a requisição de processamento do AG, que retorna como resposta os melhores cromossomos gerados a partir dos conjuntos enviados. A saída do algoritmo é a lista dos melhores cromossomos retornados por cada um dos processamentos ao final do laço de repetição. Caso o usuário opte por elaborar os horários de todos os cursos conjuntamente, basta definir o percentual de intersecção como 0. Neste caso, todos os cursos serão unidos em um único conjunto e apenas uma instância do AG será acionada para gerar os horários.

3.3.1. Pré-processamento

A etapa de pré-processamento, invocada na linha 5 do Algoritmo 1, agrupa cursos que possuem uma porcentagem pré-determinada de professores em comum (intersecção). Ela tem por objetivo diminuir o tempo de processamento do algoritmo, dividindo seu espaço de busca em frações menores a partir dos conjuntos gerados e processando essas frações individualmente. Isso diminui a quantidade de choques entre os professores dos cursos, pois os cursos que possuem uma determinada proporção de professores em comum são processados como um único conjunto. Essa etapa também abre a possibilidade de utilizar agentes distribuídos para realizar o processamento paralelo dos conjuntos criados.

Como apresentado na Figura 10, o agrupamento é feito a partir da análise da quantidade de intersecções entre os cursos e assim é criado um novo conjunto formado pela união das informações de ambos os cursos.

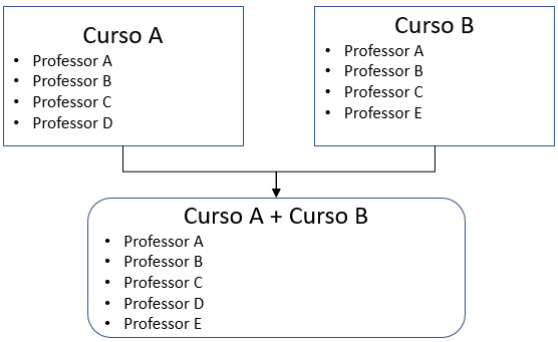


Figura 10. Agrupamento de cursos referente ao pré-processamento

Para realizar o pré-processamento é necessário definir o parâmetro de porcentagem de agrupamento, que representa a proporção de professores em comum de um curso em relação a outro. A partir de todos os cursos disponibilizados no conjunto de dados, o agrupamento acontece caso a intersecção entre o curso da iteração atual e um dos cursos restantes seja maior ou igual a porcentagem de agrupamento. No momento que um novo agrupamento é formado, o processo é reiniciado, até que não seja mais possível agrupar nenhum curso, como mostra a Figura 11.

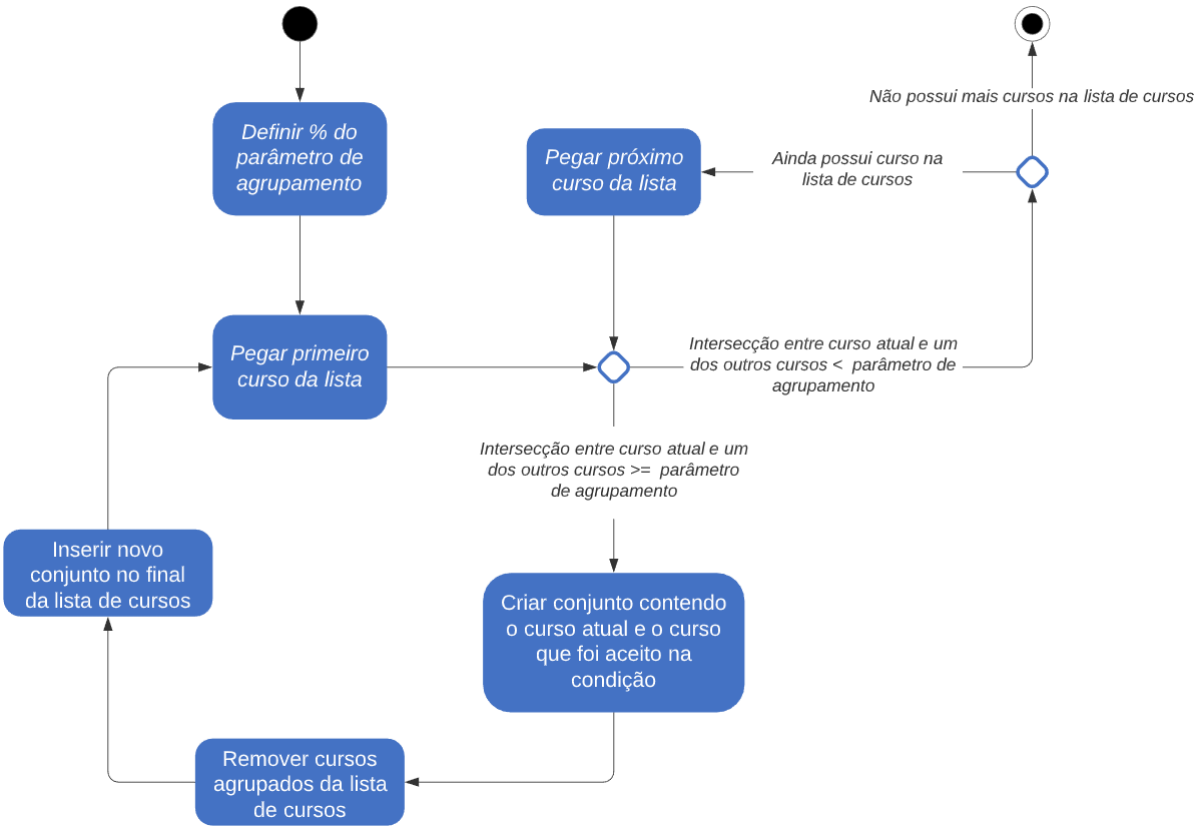


Figura 11. Diagrama de atividades do pré-processamento

3.3.2. Algoritmo Genético

O Algoritmo 2 refere-se ao processamento do método "processamentoDoAG", na linha 9 do algoritmo 1. Ele inicia com uma estrutura de repetição para cada um dos conjuntos que o computador é responsável por processar. A seguir na linha 2, é definido qual o valor máximo da função de avaliação. Esse valor é proporcional ao número de cursos do conjunto. Nas linhas 3 e 4 são realizadas as etapas de inicialização e avaliação da população, respectivamente. É importante destacar que são utilizadas *threads* durante a avaliação para paralelizar e otimizar esse processamento, visto que é um das etapas mais custosas em relação ao tempo de execução do AG. O melhor cromossomo da primeira geração é definido na linha 5. Na linha 6, é utilizado um laço de repetição que será executado até que um dos seguintes critérios de parada sejam atendidos:

- Caso o limite de iterações seja atingido;
- Caso encontre uma solução com a maior avaliação possível;

A seleção por elitismo encontra-se na linha 7, e na linha 8 o cálculo da função de avaliação acumulada é realizada, que é necessária para a seleção através do método da roleta, apresentada na linha seguinte. O cruzamento, na linha 10, é feito a partir dos cromossomos selecionados pela roleta. Para a criação de uma nova população, são utilizados os cromossomos cruzados em conjunto com os selecionados pelo elitismo, como mostra a linha 11. Após a criação dessa nova população, a mesma passa pelo processo de mutação que consta na linha 12. Nas linhas seguintes, os processos de avaliação e obtenção do melhor cromossomo são refeitos para a nova

população.

Após um dos critérios de parada serem aceitos, o melhor cromossomo é armazenado em seu respectivo conjunto. No final do processamento, é retornado o melhor cromossomo de cada conjunto.

Algoritmo 2: Processamento do AG

```
Entrada: conjuntos[], modeloAdaptado, configuracoesAG
início
1   para conjuntoAtual ← conjuntos ate ultimoConjunto faça
2       melhorAvaliacao ←
          defineMelhorAvaliacao(conjuntoAtual.obtemNumeroCursos());
3       populacao ← inicializaPopulacao(modeloAdaptado);
4       avaliaPopulacao(populacao);
5       melhorCromossomo ← obtemMelhorCromossomo(populacao);
6       enquanto não atender critérios de parada faça
7           cromossomosDeElite ←
              elitismo(configuracoesAG.porcentagemElitismo, populacao);
8           faA ← calculaFaA(populacao);
9           cromossomosDaRoleta ← roleta(configuracoesAG.porcentagemRoleta,
              faA, populacao);
10          cromossomosCruzados ←
              cruzamento(configuracoesAG.porcentagemCruzamento,
              cromossomosDaRoleta);
11          populacao ← criaNovaGeracao(cromossomosCruzados,
              cromossomosDeElite);
12          mutacao(configuracoesAG.porcentagemMutacao, populacao);
13          avaliaPopulacao(populacao);
14          melhorCromossomo ← obtemMelhorCromossomo(populacao);
        fim
15      conjuntoAtual.atribuiMelhorCromossomo(melhorCromossomo);
    fim
16  retorna conjuntos.obtemMelhoresCromossomos();
fim
Saída: Melhor cromossomo de cada um dos conjuntos processados
```

3.3.3. Pós-Processamento

Quando o usuário opta por separar os cursos em mais de um conjunto, após a execução do algoritmo genético de maneira distribuída, o resultado obtido pode violar algumas restrições *hard*. Isto ocorre porque ao unir os horários gerados para cada conjunto de cursos pode-se gerar restrições de professores que lecionam em disciplinas em um mesmo horário em cursos pertencentes aos diferentes conjuntos. Com o objetivo de resolver tais restrições, foi criado um algoritmo responsável por ajustar a solução e gerar um horário sem restrições *hard*.

O processamento do algoritmo é baseado em um algoritmo de busca em profundidade limitada. Cada nodo da árvore gerada representa uma possível solução, ou seja, um cromos-

somo diferente. A raiz representa o cromossomo criado a partir da união das soluções dos AG distribuídos e a cada nível gerado na árvore significa que uma restrição foi resolvida, consequentemente, o ultimo nível da árvore representa a solução com menos restrições possíveis.

O Algoritmo 3 descreve a chamada para o pós-processamento, que ocorre apenas se o AG foi executado de forma distribuída e caso existam conflitos de horário no cromossomo resultante, como mostra a linha 1. Caso esses critérios sejam aceitos, a lista com todos os conflitos do cromossomo é obtida na linha 2. Em seguida, nas linhas 3 e 4, a pilha usada para controlar o processo de busca em profundidade é criada e o cromossomo é inserido na mesma. A linha 5 invoca o método que executa o pós-processamento e que retorna o cromossomo com o menor número de restrições possível.

Algoritmo 3: Chamada do Algoritmo de Pós-Processamento

```
   Entrada: cromossomoFinal
   início
1      se processamentoFoiDistribuido() & cromossomoFinal.existemConflitos()
        então
2          listaConflitos[] ← cromossomoFinal.obtemConflitos();
3          pilha ← Ø;
4          pilha.empilha(cromossomoFinal);
5          retorna arvoreProfundidadeLimitada(pilha,listaConflitos);
       fim
   fim
   Saída: Melhor cromossomo possível
```

O algoritmo recursivo 4 refere-se ao método chamado no algoritmo 3, na linha 5. Na linha 1, o cromossomo do topo da pilha é obtido, logo após, na linha 2 é verificado se o cromossomo que foi passado para o método é a melhor solução possível, caso a condição seja satisfeita o cromossomo é retornado como mostra a linha 3, dando fim à recursividade. Na linha 4, o método retorna a lista de conflitos ordenada pelo seu grau de dificuldade, ou seja, quanto menor o tempo livre para troca dos horários dos professores de uma turma com conflito, maior é a sua dificuldade. Após esta definição, a lista é iterada na linha 5. Na linha 6 obtem-se a turma em que o conflito está presente e, na linha 7, é feito o processo de geração de um nodo filho. Esse processo parte do cromossomo atual e tenta fazer trocas entre as disciplinas da turma com conflito, após uma troca ser feita verifica-se se o conflito foi resolvido e também se um novo conflito não foi gerado. Caso esses critérios sejam atendidos, um novo nodo é gerado a partir do cromossomo alterado, caso contrário, nenhum filho é gerado. Na linha 8, verifica-se se o método anterior gerou um filho:

- Caso tenha gerado, esse filho é inserido na pilha e o conflito atual é removido da lista de conflitos, como mostram as linhas 9 e 10.
- Caso não tenha gerado nenhum filho, o cromossomo atual é removido da pilha, como apresentado na linha 12, o que possibilita a realização do *backtracking* para que outros ramos da árvore sejam explorados.

Por fim, na linha 13, é aplicada a recursividade a esse método a qual levará a solução do próximo conflito.

Algoritmo 4: Árvore de profundidade limitada

```
Entrada: pilha,listaConflitos
início
1   | cromossomoAtual ← pilha.topo();
2   | se cromossomoAtual.melhorAvaliacaoPossivel() então
3   |   | retorna cromossomoAtual;
    | fim
4   | restricoesOrdenadas ← listaConflitos.obtemRestricoesOrdenadas();
5   | para restricaoAtual ← restricoesOrdenadas ate ultimaRestricaoOrdenada
    |  faça
6   |   | turmaConflito ← cromossomoAtual.obtemTurmaConflito();
7   |   | filho ← geraFilho(turmaConflito,cromossomoAtual,restricaoAtual);
8   |   | se filhoExiste(filho) então
9   |   |   | pilha.empilha(filho);
10  |   |   | listaConflitos.removeConflito(restricaoAtual);
    |   | fim
11  |   | senão
12  |   |   | pilha.desempilha();
    |   | fim
13  |   | retorna arvoreProfundidadeLimitada(pilha,listaConflitos);
    | fim
fim
Saída: Melhor cromossomo possível
```

3.4. Apresentação do sistema

Antes da execução do AG, o sistema faz a leitura de um arquivo contendo as configurações apresentadas na Tabela 4. Essa Tabela contém os parâmetros do AG, além das demais configurações para as execuções centralizada e distribuída. Utilizando a Tabela 4 e também o arquivo com dados gerados por um sistema já existente no câmpus Lages do IFSC (Quadro 1), o AG é executado, gerando os resultados apresentados na Figura 12.

Tabela 4. Representação das Configurações do Sistema

Nome do parâmetro	Valor do parâmetro
Tamanho da Turma	10
Tamanho da População	200
Porcentagem de Elitismo	10
Porcentagem de Mutação	10
Porcentagem de Cruzamento	60
Porcentagem de Intersecção entre Cursos	60
Intervalo de Verificação de Gerações	1000
Total de Gerações	10000

Quadro 1. Amostra do arquivo XML

```
<?xml version=”1.0”encoding=”UTF-8”?>

<lesson id=”5”classid=”321”periodsperweek=”4”durationperiods=”2”classroomids=”54”
teacherids=”67,”subjectid=”272”/>

<lesson id=”6”classid=”321”
periodsperweek=”2”durationperiods=”2”classroomids=”54”teacherids=”8,”
subjectid=”273”/>
...
<teacher short=”ailton.durigon”name=”Ailton Durigon”
id=”27”timeoff=”.000000000000,.111111111111,.111111111111,
.111111111111,.111111111111,.000000000000”/>

<teacher short=”alexandre.perin”
name=”Alexandre Perin de Souza”
id=”10”timeoff=”.111111111111,.111111111111,.11111111111,
.111111111111,.000011111111,.000000000000”/>
...
```

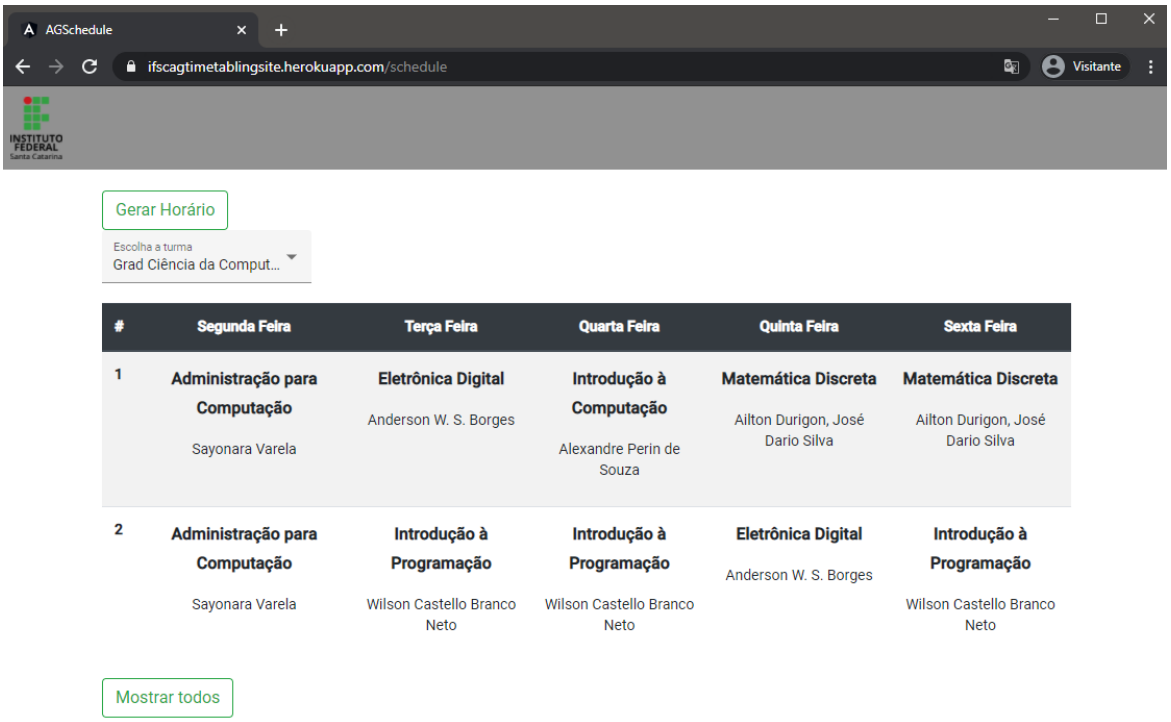


Figura 12. Amostra da interface web de um curso específico

4. Resultados

Nessa seção são apresentados os testes realizados e seus respectivos resultados, com o objetivo de determinar a melhor combinação de parâmetros para a execução do AG, tanto em ambiente centralizado, quanto no ambiente distribuído.

4.1. Configuração dos computadores

Para a execução dos testes centralizados e distribuídos, foram utilizados de maneira remota seis computadores alocados no Laboratório de Informática 115 do IFSC. Nos testes centralizados, cada máquina executou de maneira independente um algoritmo genético, já nos testes distribuídos, foi adotada a arquitetura cliente-servidor, no qual uma máquina atuou como cliente e as demais como servidor. Também é importante ressaltar que tanto no ambiente centralizado, quanto no ambiente distribuído, foram utilizadas *threads* de maneira dinâmica, ou seja, o sistema utiliza o número de núcleos disponíveis de cada computador para otimizar o desempenho, como citado anteriormente na seção 3.3.2. Todas as máquinas possuem processador Intel Core 3 6100T 3.2 GHz de 64 bits, além de 8 GB de memória de RAM.

4.2. Definição dos parâmetros do AG para montar os horários de todos os cursos juntos

Foi realizado um teste fatorial com os parâmetros de porcentagem de mutação, elitismo, cruzamento e tamanho da população, com o intuito de analisar cada parâmetro e suas interações. Neste primeiro teste, o percentual de intersecção entre os professores foi definido como 0 para que os horários de todos os cursos fossem montados de uma única vez, sem processamento distribuído. Os valores testados para cada parâmetro são:

- Porcentagem de Mutação: 5, 10, 20, 30 e 50;
- Porcentagem de Elitismo: 4, 8, 20, 30 e 40;
- Porcentagem de Cruzamento: 30, 50, 60 ,70 e 90;
- Tamanho da População: 100, 200, 300, 400 e 500.

Após a execução dos testes foram realizadas as análises de variância e os testes de *Tukey* em relação a três variáveis: resultado, tempo e total de gerações.

A análise de variância é um teste estatístico que visa verificar a existência de diferenças significativas entre as médias, e se os fatores exercem influência em alguma variável em situações que seja necessária a comparação entre dois ou mais grupos de amostras.

As tabelas de análise de variância apresentam as seguintes colunas:

- Fator: Indica qual a variável (tratamento) ou interações entre variáveis que serão analisadas. A linha residual representa as variações que não são explicadas pelos demais fatores;
- Soma dos quadrados: Representa uma medida de variação ou desvio da variável em relação a média geral dos dados, obtido através da soma dos quadrados das diferenças da média;
- Média dos quadrados: É o valor que identifica a variância do tratamento. Esse valor é obtido através da divisão da soma dos quadrados do tratamento, pelo número de grau de liberdade (elementos da amostra menos um);
- F: Descreve o tamanho da diferença entre as amostras em função da variação dentro de cada amostra, através da divisão do valor da média dos quadrados do tratamento pelo residual. Esse valor indica se o fator é significativo ou não nas diferenças dos resultados;
- P: É a probabilidade de a estatística do teste ser um valor extremo em relação ao valor observado. Para os testes, foi definido como 5% (0,05), então caso o valor seja menor que esse, aceita-se como uma variável significativa e rejeita-se caso contrário.

Nas tabelas de análise de variância apresentadas na sequência, as linhas destacadas com um asterisco indicam que os diferentes valores atribuídos àquele parâmetro ou combinação

de parâmetros levam a diferenças estatisticamente significativas nos resultados. Para cada parâmetro identificado como significativo pela análise de variância, foi realizado o teste de *Tukey* já que esse é capaz de identificar quais valores dos parâmetros levam aos melhores resultados e quais são estatisticamente diferentes.

A primeira variável analisada foi o resultado, a qual indica a qualidade da solução gerada. Neste teste, 1500 indica uma solução sem a violação de nenhuma restrição. Este é o principal critério para escolher os melhores parâmetros do AG. Além dela, analisou-se o tempo de processamento e, por último, o total de gerações executadas como critérios de desempate caso existam mais de uma configuração de parâmetros que leve aos melhores resultados.

A análise de variância em relação a variável resultado do AG é apresentada na Tabela 5. Ela indica que os parâmetros população, mutação e elitismo individualmente são significativos, com destaque a mutação que apresentou um valor F bastante elevado. Em relação às interações entre parâmetros, cabe destacar que todas as que foram consideradas significativas estão relacionadas à mutação, em função do seu alto valor de F já mencionado.

Tabela 5. Análise de variância entre os parâmetros em relação ao resultado (centralizado)

Fator	Soma dos Quadrados	Média dos Quadrados	F	P
Somente População	8,37E+02	2,09E+02	78,29*	0*
Somente Mutação	2,72E+03	6,79E+02	254,12*	0*
Somente Cruzamento	7,97E+00	1,99E+00	0,75	0,560669
Somente Elitismo	7,58E+01	1,90E+01	7,09*	0,000011*
População X Mutação	1,76E+03	1,10E+02	41,07*	0*
População X Cruzamento	4,61E+01	2,88E+00	1,08	0,369788
Mutação X Cruzamento	2,61E+01	1,63E+00	0,61	0,877529
População X Elitismo	4,58E+01	2,86E+00	1,07	0,376972
Mutação X Elitismo	2,20E+02	1,38E+01	5,15*	0*
Cruzamento X Elitismo	3,90E+01	2,43E+00	0,91	0,555708
População X Mutação X Cruzamento	1,28E+02	2,00E+00	0,75	0,932729
População X Mutação X Elitismo	2,42E+02	3,78E+00*	1,42*	0,016691*
População X Cruzamento X Elitismo	2,16E+02	3,38E+00	1,26	0,07724
Mutação X Cruzamento X Elitismo	1,63E+02	2,55E+00	0,96	0,577139
População X Mutação X Cruzamento X Elitismo	7,36E+02	2,88E+00	1,08	0,197666
Residual	1,50E+05	2,67E+06	-	-

Com base nos resultados da análise de variância, foi possível identificar que ao menos um valor dentre os parâmetros População, Mutação e Elitismo se difere dos demais. Porém, através dela não é possível constatar quais valores possuem uma diferença significativa. Devido a isso, foram realizados testes de *Tukey*, que faz a comparação de todos os possíveis pares de médias através da Diferença Mínima Significativa (D.M.S). Então, a partir desses testes, é possível identificar qual o melhor valor para cada parâmetro. Para isso, são utilizadas classes (letras presentes nos gráficos) para representá-los, sendo que a letra "a" indica o melhor valor para o parâmetro, "b" o segundo melhor e assim sucessivamente. Existem casos em que devido a variação residual dos resultados, alguns valores podem assumir duas classes ao mesmo tempo.

A Figura 13 apresenta o resultado do teste de *Tukey* para cada parâmetro considerado significativo na análise de variância em relação à variável resultado, vale ressaltar que a escala da Figura vai de 1498 até 1500 para uma melhor visualização das diferenças. A partir dele, percebe-se que os valores identificados para cada parâmetro que levam aos melhores resultados são:

- Porcentagem de Mutação: 20, 30 e 50;

- Tamanho da População: 300, 400 e 500;
- Porcentagem de Elitismo: 20, 30 e 40;

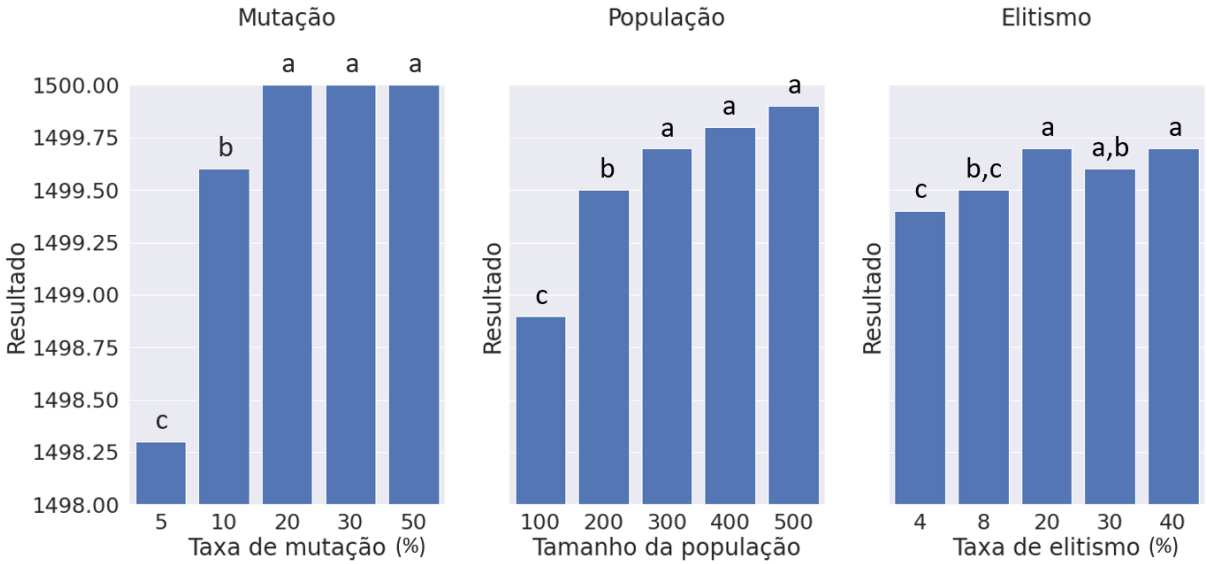


Figura 13. Teste de Tukey em relação ao resultado (centralizado)

Considerando que mais de um valor para os parâmetros testados levam aos melhores resultados, realizou-se a análise de variância em relação ao tempo de execução, como mostra a Tabela 6. Nela é possível identificar, novamente, que os parâmetros população, mutação e elitismo apresentam significância, assim como as interações relacionadas ao parâmetro mutação.

Tabela 6. Análise de variância entre os parâmetros em relação ao tempo (centralizado)

Fator	Soma dos Quadrados	Média dos Quadrados	F	P
Somente População	22915725	5728931	66,084*	0*
Somente Mutação	326158531	81539633	9,41E+02*	0*
Somente Cruzamento	3,44E+05	8,60E+04	9,92E-01	4,10E-01
Somente Elitismo	29019648	7254912	83,686*	0*
População X Mutação	18179184	1136199	13,106*	0*
População X Cruzamento	1,13E+06	7,07E+04	8,15E-01	6,69E-01
Mutação X Cruzamento	1,05E+06	6,54E+04	7,55E-01	7,39E-01
População X Elitismo	2745486	171593	1,98*	0,011202*
Mutação X Elitismo	9530094	595631	6,871*	0*
Cruzamento X Elitismo	1,27E+06	7,93E+04	9,14E-01	5,52E-01
População X Mutação X Cruzamento	2,63E+06	4,11E+04	4,74E-01	1,00E+00
População X Mutação X Elitismo	7538952	117796	1,359*	0,030651*
População X Cruzamento X Elitismo	4,50E+06	7,04E+04	8,12E-01	8,59E-01
Mutação X Cruzamento X Elitismo	4,59E+06	7,18E+04	8,28E-01	8,34E-01
População X Mutação X Cruzamento X Elitismo	1,85E+07	7,22E+04	8,33E-01	9,74E-01
Residual	4,88E+14	8,67E+10	-	-

O teste de *Tukey* em relação ao tempo de processamento é apresentado na Figura 14, variando de 100000 até 700000 para acentuar a diferença entre os parâmetros. A partir dele, conclui-se que os valores que levam aos melhores resultados em menor tempo são:

- Porcentagem de Mutação: 50;
- Tamanho da População: 300 e 400;

- Porcentagem de Elitismo: 20, 30 e 40;

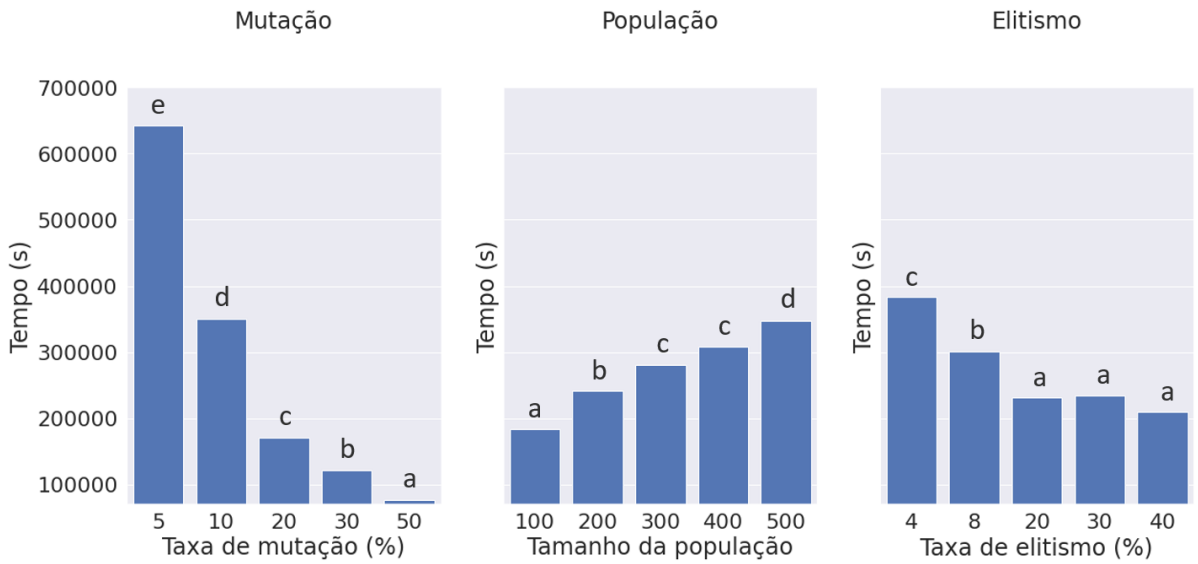


Figura 14. Teste de Tukey em relação ao tempo (centralizado)

Por fim, a análise de variância presente na Tabela 7 apresenta os resultados em relação à variável total de gerações. Nela percebe-se o mesmo comportamento que ocorreu nas duas análises anteriores.

Tabela 7. Análise de variância entre os parâmetros em relação ao total de gerações (centralizado)

Fator	Soma dos Quadrados	Média dos Quadrados	F	P
Somente População	2,97E+17	7,41E+16	170.142.648*	9,52E-132*
Somente Mutação	1,60E+18	3,99E+17	915.391.662*	0,00E+00*
Somente Cruzamento	1,12E+15	2,80E+14	642.137	6,32E+05
Somente Elitismo	9,27E+16	2,32E+16	53.197.075*	4,58E-38*
População X Mutação	1,10E+17	6,86E+15	15.750.975*	2,30E-37*
População X Cruzamento	5,94E+15	3,71E+14	852.412	6,26E+05
Mutação X Cruzamento	4,70E+15	2,94E+14	674.061	8,22E+05
População X Elitismo	3,93E+15	2,46E+14	564.053	9,12E+05
Mutação X Elitismo	1,54E+16	9,63E+14	2.211.067	3,60E+03
Cruzamento X Elitismo	7,50E+15	4,69E+14	1.075.711	3,72E+05
População X Mutação X Cruzamento	1,93E+16	3,02E+14	692.169	9,71E+05
População X Mutação X Elitismo	3,68E+16	5,75E+14	1.319.929	4,54E+04
População X Cruzamento X Elitismo	3,05E+16	4,77E+14	1.095.007	282.418
Mutação X Cruzamento X Elitismo	2,42E+16	3,78E+14	866.388	767.188
População X Mutação X Cruzamento X Elitismo)	1,10E+17	4,29E+14	984.366	558.188
Residual	2,45E+18	4,36E+14	-	-

O resultado do teste de *Tukey* para a variável total de gerações é apresentado na Figura 15, a escala vai de 20000 até 60000 para melhor visualizar as diferenças entre os parâmetros, assim como nos testes anteriores. Os valores dos parâmetros que levam aos melhores resultados, em menor tempo e com o menor número de gerações são:

- Porcentagem de Mutação: 50;
- Tamanho da População: 400;
- Porcentagem de Elitismo: 20, 30 e 40;

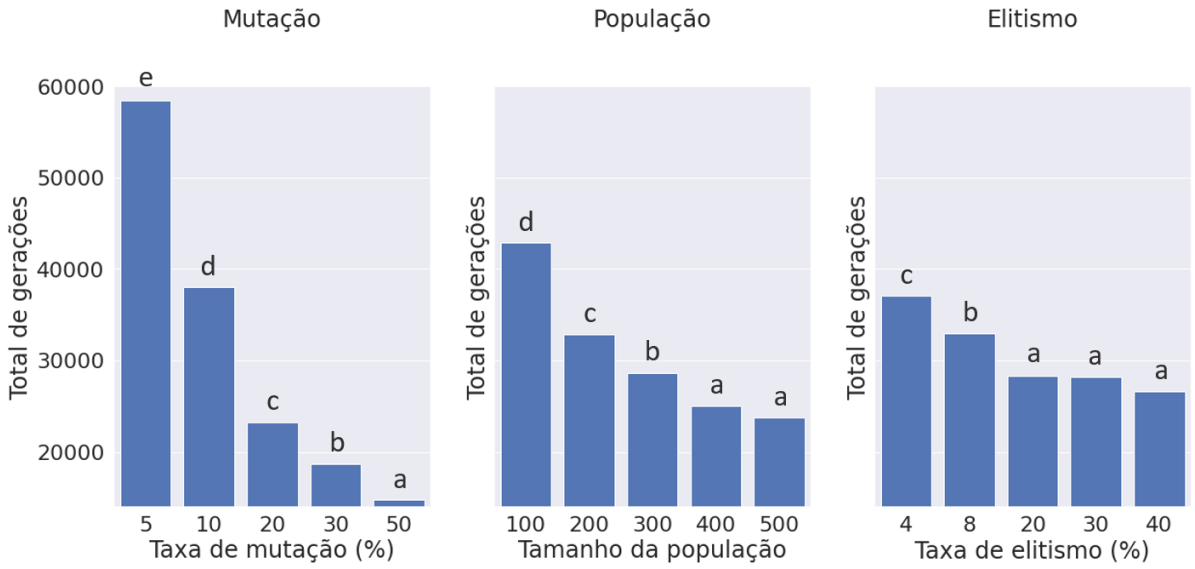


Figura 15. Teste de Tukey em relação ao total de gerações (centralizado)

Realizando análises preliminares sobre tais resultados, foi identificado que as interações significativas se deram por conta do alto valor F do parâmetro de mutação, que influenciou fortemente essas interações, assim uma análise individual de cada parâmetro foi suficiente para definir os melhores valores dos parâmetros.

A partir dos resultados da análise de variância, também identificou-se que o parâmetro de cruzamento não apresentou considerável significância, o que indica que as demais variáveis não serão influenciadas de maneira relevante, independente do valor que assumirem.

Pode-se concluir, que baseado nos testes de *Tukey* os melhores parâmetros de cada uma das variáveis, para o conjunto de dados utilizado, são:

- Porcentagem de Mutação: 50;
- Tamanho da População: 400;
- Porcentagem de Elitismo: 40;
- Porcentagem de Cruzamento: 60;
- Porcentagem de Intersecção: 0.

Vale ressaltar que o parâmetro taxa de cruzamento não afetou os resultados de maneira significante. Assim, a taxa de cruzamento foi definida como 60 por ser um valor intermediário entre os testados.

4.3. Definição dos parâmetros do AG para montar os horários dos cursos de modo distribuído

Assim como nos testes feitos na seção anterior, foi executado um teste fatorial com os parâmetros de porcentagem de mutação, elitismo, cruzamento e tamanho da população. Nesse teste o percentual de intersecção foi definido como 60%, o que gerou quatro conjuntos, assim distribuindo-os entre três computadores servidores e um computador cliente, disponíveis no momento do teste, e os valores testados para os demais parâmetros são os mesmos apresentados no teste anterior.

A partir dos resultados dos testes, também foram realizadas as análises de variância e os testes de *Tukey* em relação ao resultado, tempo e total de gerações. Seguindo o modelo dos testes anteriores, os valores destacados com um asterisco indicam os resultados significativos.

A primeira variável analisada foi o Resultado, a qual indica a qualidade da solução gerada. Neste teste, 1500 indica uma solução sem a violação de nenhuma restrição. Este é o principal critério para escolher os melhores parâmetros do AG. Além dela, analisou-se o tempo de processamento e, por último, o total de gerações executadas como no teste anterior.

A análise de variância em relação a variável Resultado do AG é apresentada na Tabela 8. Ela indica que os parâmetros população e mutação individualmente são significativos. Em relação às interações entre parâmetros, cabe destacar que as que foram consideradas significativas estão relacionadas à população e mutação.

Tabela 8. Análise de variância entre os parâmetros em relação ao resultado(distribuído)

Fator	Somas dos quadrados	Média dos quadrados	F	P
Somente População	4,39E+03	1,10E+03	3*	0,014729*
Somente Mutação	1,17E+04	2,92E+03	8*	0,000001*
Somente Cruzamento	2,05E+03	5,12E+02	1	0,21647
Somente Elitismo	3,57E+02	8,94E+01	0	0,90845
População X Mutação	1,36E+04	8,50E+02	2*	0,001382*
População X Cruzamento	9,94E+03	6,21E+02	2*	0,031609*
Mutação X Cruzamento	7,71E+03	4,82E+02	1	0,151685
População X Elitismo	5,26E+03	3,29E+02	1	0,53591
Mutação X Elitismo	4,87E+03	3,05E+02	1	0,617065
Cruzamento X Elitismo	2,38E+03	1,49E+02	0	0,978403
População X Mutação X Cruzamento	2,71E+04	4,23E+02	1	0,140409
População X Mutação X Elitismo	2,22E+04	3,46E+02	1	0,527541
População X Cruzamento X Elitismo	1,44E+04	2,26E+02	1	0,98934
Mutação X Cruzamento X Elitismo	1,76E+04	2,75E+02	1	0,903682
População X Mutação X Cruzamento X Elitismo	7,34E+04	2,87E+02	1	0,987631
Residual	1,99E+12	3,54E+08	-	-

A Figura 16 apresenta o resultado do teste de *Tukey* para os parâmetros considerados significativos no análise de variância em relação à variável resultado. Faz-se necessário comentar que escala da Figura vai de 1490 a 1500, para melhorar visualização entre os parâmetros. A partir dele, percebe-se que os valores identificados para cada parâmetro que levam aos melhores resultados são:

- Porcentagem de Mutação: 5;
- Tamanho da População: 200, 300, 400 e 500;

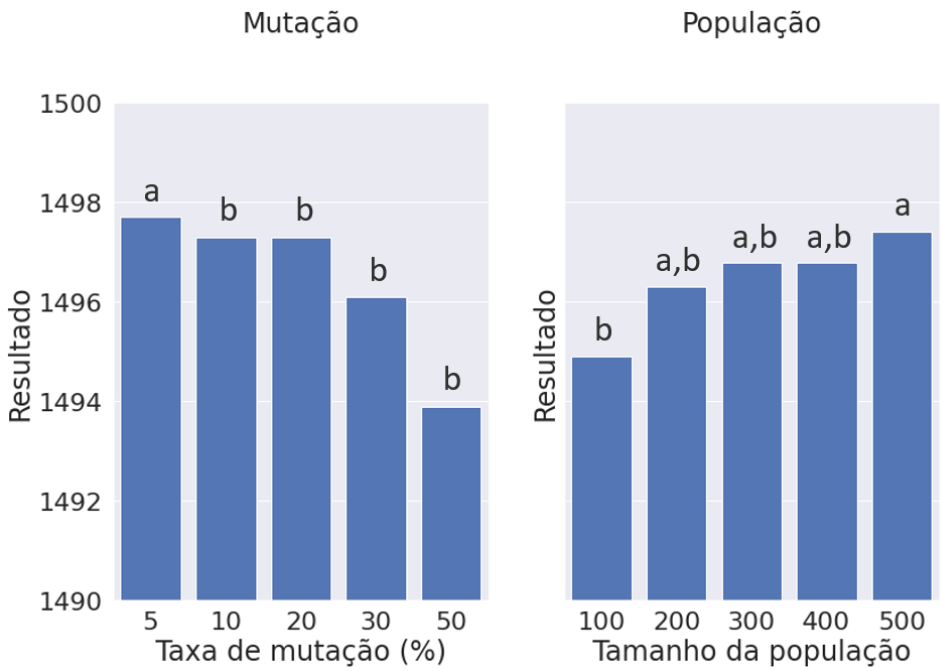


Figura 16. Teste de Tukey em relação ao resultado(distribuído)

Considerando que mais de um valor para os parâmetros testados levam aos melhores resultados, realizou-se a análise de variância em relação ao tempo de execução, como mostra a Tabela 9. Nela é possível identificar, que os parâmetros população, mutação e cruzamento apresentam significância nas interações individuais e também entre si.

Tabela 9. Análise de variância entre os parâmetros em relação ao tempo (distribuído)

Fator	Somas dos quadrados	Média dos quadrados	F	P
Somente População	2,48E+12	6,21E+11	1969,42*	0*
Somente Mutação	8,07E+10	2,02E+10	63,97*	0*
Somente Cruzamento	4,43E+10	1,11E+10	35,09*	0*
Somente Elitismo	1,25E+09	3,14E+08	0,99	0,409384
População X Mutação	5,43E+10	3,39E+09	10,76*	0*
População X Cruzamento	3,00E+10	1,88E+09	5,95*	0*
Mutação X Cruzamento	5,70E+10	3,56E+09	11,29*	0*
População X Elitismo	4,77E+09	2,98E+08	0,94	0,516313
Mutação X Elitismo	4,69E+09	2,93E+08	0,93	0,533252
Cruzamento X Elitismo	6,92E+09	4,32E+08	1,37	0,145831
População X Mutação X Cruzamento	5,02E+10	7,84E+08	2,49*	0*
População X Mutação X Elitismo	1,43E+10	2,23E+08	0,71	0,962739
População X Cruzamento X Elitismo	2,24E+10	3,51E+08	1,11	0,253455
Mutação X Cruzamento X Elitismo	1,92E+10	3,00E+08	0,95	0,587199
População X Mutação X Cruzamento X Elitismo	8,54E+10	3,34E+08	1,06	0,255738
Residual	3,54E+08	3,15E+14	-	-

O teste de *Tukey* em relação ao tempo de processamento é apresentado na Figura 17, e contém uma escala que vai de 20000 até 70000, com o intuito de facilitar a visualização das diferenças entre os parâmetros. A partir dele, conclui-se que os valores que levam aos melhores resultados em menor tempo são:

- Porcentagem de Mutação: 5;

- Tamanho da População: 200;
- Porcentagem de Cruzamento: 30;

Apesar de o resultado para a variável tamanho da população apresentar o valor 100 como o melhor no teste da variável tempo, o mesmo não foi considerado, pois no teste anterior (variável resultado) verificou-se que ele possui uma diferença estatisticamente significativa em relação aos resultados gerados, quando comparados com os resultados gerados pelos valores 200, 300, 400 e 500. Em função disto, o melhor valor para o parâmetro tamanho da população foi definido como 200.

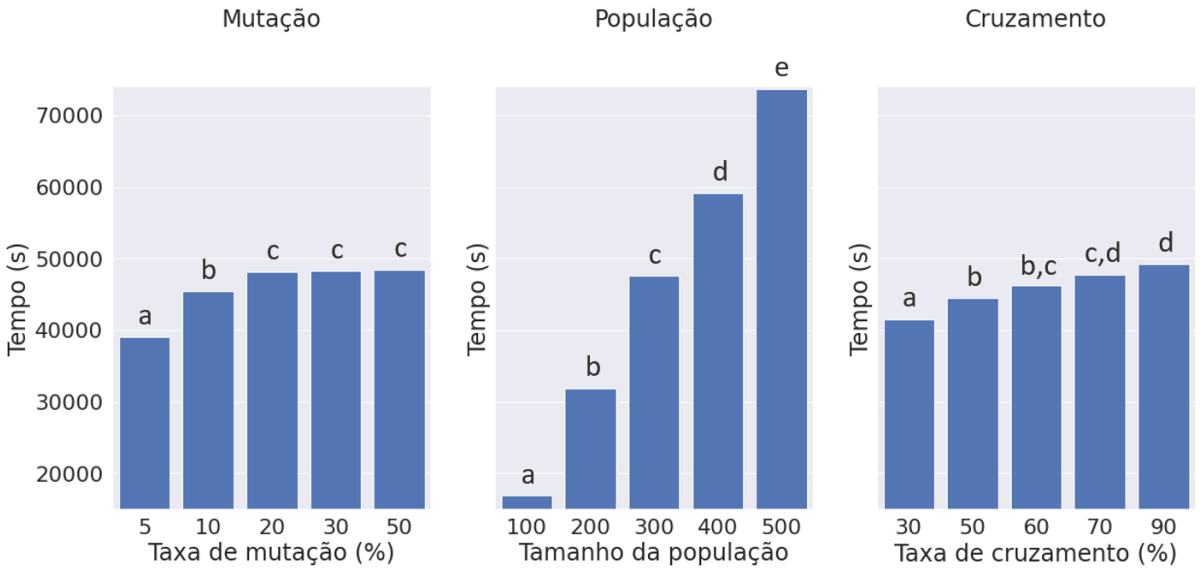


Figura 17. Teste de Tukey em relação ao tempo(distribuído)

A análise individual de cada parâmetro utilizados nos testes foi suficiente para definir os melhores valores dos parâmetros, assim como concluído na análise dos testes centralizados.

A partir dos resultados da análise de variância, também identificou-se que o parâmetro de elitismo não apresentou considerável significância, o que indica que as demais variáveis não serão influenciadas de maneira relevante, independente do valor que assumir.

Pode-se concluir, que baseado nos testes de *Tukey* os melhores parâmetros de cada uma das variáveis, a partir do conjunto de dados utilizado, são:

- Porcentagem de Mutação: 5;
- Tamanho da População: 200;
- Porcentagem de Elitismo: 20;
- Porcentagem de Cruzamento: 30;
- Porcentagem de Intersecção: 60.

Vale ressaltar que o parâmetro porcentagem de elitismo não afetou os resultados de maneira significante. Assim, a taxa de cruzamento foi definida como 20 por ser um valor intermediário entre os testados.

4.4. Comparação da solução centralizada e distribuída

Considerando os testes feitos em ambiente centralizado e distribuído, foram obtidos as melhores configurações de parâmetros de execução do AG para cada um dos casos. A partir disso,

foi feito um teste, com dez replicações, para comparar as duas formas de execução utilizando suas configurações ótimas, com o objetivo de identificar a forma de execução com melhor desempenho.

Na Figura 18 é possível visualizar a comparação dos resultados das duas execuções quando utilizadas as melhores configurações, identificadas nas duas seções anteriores. No gráfico à esquerda pode-se visualizar que ambas as execuções obtiveram resultados perfeitos em 100% dos testes realizados, sendo assim foi possível visualizar diferenças entre elas apenas quando se compara o tempo de execução, no qual pode-se observar que a execução distribuída obteve tempos melhores do que a execução centralizada.

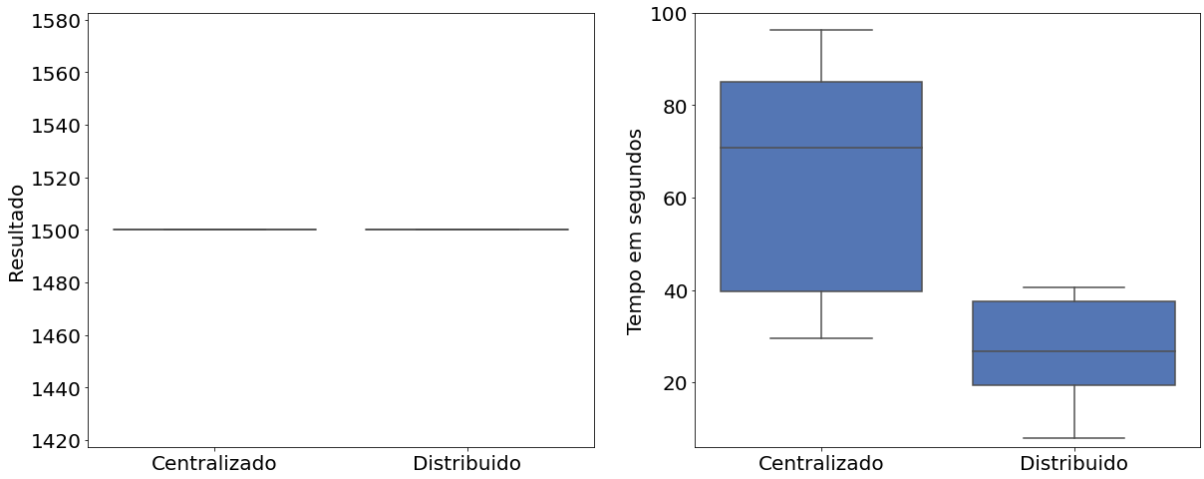


Figura 18. Diagrama de caixas comparando a solução centralizada com a distribuída

Por fim, a Tabela 10 apresenta a análise de variância dos testes executados, para comparar o tempo de execução dos diferentes tipos de processamento - centralizado e distribuído. Como pode ser verificado, o resultado é estatisticamente significativo, o que indica que há uma clara diferença no tempo de execução do AG de forma centralizada e distribuída.

Tabela 10. Análise de variância entre a execução centralizada e distribuída em relação ao tempo

Fator	Soma dos quadrados	Média dos quadrados	F	P
C(Tipo)	7260,793459	7260,793459	18,099868	0,000477
Residual	6,66E+15	3,70E+14	-	-

5. Considerações Finais

Este trabalho teve como objetivo elaborar um AG para resolução de *timetabling* para o IFSC Câmpus Lages, de acordo com seus recursos e restrições. O algoritmo pode elaborar horários de vários cursos juntos, o que leva a um grande espaço de busca mas gera uma solução final para o problema, ou de cada curso por vez, o que reduz drasticamente o espaço de busca mas gera uma solução que pode ter restrições violadas quando as soluções de cada curso são unidas para formar a solução final.

Com o objetivo de melhorar o desempenho do processamento do AG foi implementado um módulo que tem o papel de processar os dados provenientes do IFSC, dividindo os cursos em conjuntos menores para posteriormente serem processados no AG. Para ser possível realizar o processamento do AG, primeiro foi necessário fazer a modelagem do AG de forma a respeitar a estrutura curricular do IFSC, bem como resolver possíveis problemas, como matérias fora de seus cursos e problemas de carga horária.

Assim como nos trabalhos relacionados, a estrutura do cromossomo se dá por um vetor de números inteiros que representa uma matéria, lecionada por um professor, em um curso. Além disso, cada posição do cromossomo representa duas aulas e a cada dez posições tem-se uma turma. A fim de resolver os possíveis problemas de modelagem citados, a inicialização foi modificada para inserir as matérias de cada turma e depois realizar trocas aleatórias dentro de seus respectivos intervalos. Com o mesmo objetivo, foi implementado um operador de cruzamento baseado na Recombinação Ordenada, para manter a estrutura das turma com suas respectivas matérias, assim como o operador de mutação.

Quanto a implementação, o primeiro modelo de execução do AG foi desenvolvido de forma centralizada, utilizando uma única *thread*, porém os resultados não foram tão expressivos como o esperado. Sendo assim, foi implementada a utilização de *threads* no processamento do AG, na função de avaliação, o que gerou resultados promissores e concluiu a etapa de execução em ambiente centralizado. Para execução em ambiente distribuído, foi abordada a mesma ideia de utilização de *threads* para otimizar ainda mais o processamento. Com a execução distribuída, fez-se necessário a elaboração de um algoritmo de árvore de busca em profundidade limitada para resolver os conflitos entre os conjuntos resultantes do processamento distribuído.

Em relação a comparação dos resultados das diferentes formas de implementações, através da análise de variância e testes de *Tukey* foi possível identificar os melhores valores para cada um dos parâmetros tanto no ambiente centralizado, quanto no distribuído. Após a obtenção destes valores, foi realizado um teste comparativo entre ambos os ambientes, no qual foi possível identificar que ambos obtiveram 100% de eficácia em relação aos resultados e a solução distribuída proposta chegou no resultado perfeito em um tempo menor, com o AG chegando a solução em uma média de 26 segundos, enquanto a solução centralizada levou, em média, 70 segundos. Assim, pode-se concluir que a solução distribuída proposta neste artigo apresenta um desempenho melhor.

Como possíveis trabalhos futuros, tem-se a possibilidade de realizar o balanceamento de carga no ambiente distribuído levando em consideração critérios de *benchmark* e configuração dos computadores. Além disso, é necessário realizar mais comparações utilizando conjuntos de dados de outros semestres, para assim garantir que o AG proposto pode substituir o *software* atual de geração de horários do IFSC. Por fim, cabe a melhoria na modelagem dos conjuntos de dados para possibilitar a utilização dos dados do ITC, para possibilitar a comparação com os resultados apresentados em outros trabalhos que usaram os dados deste conjunto em seus testes.

Referências

Alves, S. S. A., Oliveira, S. A. F., e Rocha Neto, A. R. (2017). *A Recursive Genetic Algorithm-Based Approach for Educational Timetabling Problems*, pages 161–175. Springer International Publishing, Cham.

Bolaji, A. L., Khader, A. T., Al-Betar, M. A., e Awadallah, M. A. (2014). University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of Computational Science*, 5(5):809–818.

Borges, C. P. (2007). Aperfeiçoamento de um algoritmo genético para elaboração de quadros de horários em instituições de ensino. Trabalho de conclusão de curso em Sistemas de Informação. Universidade do Planalto Catarinense - SC.

Burke, E., Mccollum, B., Meisels, A., Petrovic, S., e Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176:177–192.

Burke, E. K. e Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74.

Burke, E. K., Qu, R., e Soghier, A. (2009). Adaptive selection of heuristics within a grasp for exam timetabling problems. *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications*, pages 10–12.

Colorni, A., Dorigo, M., e Maniezzo, V. (1999). Metaheuristics for high-school timetabling. *Computational Optimization and Applications*, 9.

Concilio, R. (2000). Contribuições à solução de problemas de escalonamento pela aplicação conjunta de computação evolutiva e otimização com restrições.

Costa, L. H. M., de Castro, M. A. H., e Ramos, H. (2010). Utilização de um algoritmo genético híbrido para operação ótima de sistemas de abastecimento de água. *Engenharia Sanitária e Ambiental*, 15(2):187 – 196.

Cruz, R. F., dos Santos Júnior, G. P., Fontes, L. B., dos Anjos Santos, M., e da Silva, B. L. C. (2019). Geração automática de horário escolar com algoritmo genético. *Eixo*, 8(2):230–241.

da Silva, E. M. (2010). Algoritmos genéticos para geração de quadros de horários. Trabalho de conclusão de curso em Sistemas de Informação. Universidade do Planalto Catarinense - SC.

Daskalaki, S. e Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160:106–120.

Di Gaspero, L. e Schaerf, A. (2001). Tabu search techniques for examination timetabling. In Burke, E. e Erben, W., editors, *Practice and Theory of Automated Timetabling III*, pages 104–117, Berlin, Heidelberg. Springer Berlin Heidelberg.

Edelkamp, S. e Schrödl, S. (2012). *Heuristic Search - Theory and Applications.* Morgan Kaufmann, 1st edition.

Linden, R. (2012). *Algoritmos Genéticos*. Ciência Moderna, Rio de Janeiro, 3 edition.

Luger, G. (2013). *Inteligência artificial*. Pearson Education do Brasil, São Paulo, 6th edition.

Montana, D. J., Brinn, M., Moore, S., e Bidwell, G. (1998). Genetic algorithms for complex, real-time scheduling. *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, 3:2213–2218 vol.3.

Nourmohammadi-Khiarak, J., Zamani-Harghalani, Y., e Feizi-Derakhshi, M.-R. (2017). Combined multi-agent method to control inter-department common events collision for university courses timetabling. *Journal of Intelligent Systems*, 29(1):110–126.

Ramos, P. d. S. R. (2002). Sistema automático de geração de horários para a ufla utilizando algoritmos genéticos. Trabalho de conclusão de curso em Ciência da Computação. Universidade Federal de Lavras - MG.

Resende, M. e Ribeiro, C. (2010). *Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications*, volume 146, pages 283–319. Springer.

Salza, P. e Ferrucci, F. (2018). Speed up genetic algorithms in the cloud using software containers. *Future Generation Computer Systems*, 92.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127.

Schiezaro, M. e Pedrini, H. (2013). Data feature selection based on artificial bee colony algorithm. *EURASIP Journal on Image and Video Processing*, 2013:47.

Socha, K., Knowles, J., e Sampels, M. (2002). A max-min ant system for the university course timetabling problem. In *International Workshop on Ant Algorithms*, pages 1–13. Springer.

Sutar, S. R. e Bichkar, R. S. (2017). High school timetabling using tabu search and partial feasibility preserving genetic algorithm. *International Journal of Advances in Engineering & Technology*, 10(3):421.

Wren, A. (1995). Scheduling, timetabling and rostering—a special relationship? In *International conference on the practice and theory of automated timetabling*, pages 46–75. Springer.