Integração de Sistemas para Elaboração dos Horários do IFSC Câmpus Lages

Cícero Marques, Felipe D. Varela, Wilson Castello Branco Neto

[1]Instituto Federal de Santa Catarina - Câmpus Lages (IFSC)
Rua Heitor Villa Lobos, 222 - São Francisco, 88506-400 - Lages - SC - Brasil

ciceromarques2000@gmail.com, felipeduartevarela@gmail.com

wilsoncastello@ifsc.edu.br

Abstract. *This article presents improvements carried out in the Genetic Algorithm developed by Bianquini e Silva (2020), to be used at the Instituto Federal de Santa Catarina Câmpus Lages, aiming at elaborating the timetable considering the restrictions imposed by the institution. For this purpose, the implementation of new features that allow Genetic Algorithm reads data contained in the database was carried out. A web interface was also developed in order to customize schedules by periods, and to allow its use by people without specific computer knowledge. Finally, tests were carried out to demonstrate that the changes made improved the results generated.*

Resumo. *Este artigo apresenta as melhorias realizadas no Algoritmo Genético desenvolvido por Bianquini e Silva (2020), para ser utilizado no Instituto Federal de Santa Catarina Câmpus Lages, com o objetivo de elaborar os horários das aulas respeitando as restrições impostas pela instituição. Para isto, foi realizada a implementação de novos recursos que permitem a comunicação e a leitura dos dados contidos no banco de dados pelo Algoritmo Genético. Também foi desenvolvida uma interface web que possibilita a customização dos horários por períodos, e sua utilização por pessoas sem conhecimentos específicos de informática. Por fim, foram realizados testes que demonostram que as mudanças realizadas melhoraram os resultados gerados.*

1. Introdução

Com os avanços das tecnologias na sociedade e a Internet chegando a mais pessoas, o uso da informática se tornou essencial dentro de qualquer instituição. Na educação, ela vem ganhando mais espaço, tanto em atividades pedagógicas quanto administrativas. Com isso, surgiram várias empresas que investem e desenvolvem tecnologias para a educação, denominadas Edtech. Segundo o Centro de Inovação para a Educação Brasileira (CIEB), essas empresas buscam diversas soluções para levar a tecnologia para dentro da sala de aula ou de processos que facilitem a aprendizagem e aprimoram os sistemas educacionais. Essas soluções podem ser laboratórios virtuais, jogos educativos, bibliotecas digitais, educação a distância (EaD), sistemas de gestão educacional, entre outras (na Educação, 2021).

Um dos problemas recorrentes às instituições de ensino que é abordado por algumas empresas do setor de EdTech é o *timetabling*. O *timetabling*, de acordo com Ross

et al. (2003), é um problema de escalonamento sujeito a restrições que possui eventos que devem acontecer em um período de tempo limitado. Esse problema se faz presente em todas as instituições de ensino uma vez que, antes de iniciar o período letivo, é necessario verificar a disponibilidade de professores, disciplinas e turmas, bem como suas restrições. Desse modo, a definição de um quadro de horários que não possua conflitos e atenda a todas as restrições é uma tarefa difícil, pois gera um elevado grau de desgaste por parte dos profissionais responsáveis por essa atividade.

Problemas complexos, como o *timetabling*, podem ser resolvidos por técnicas como algoritmos de busca em grafos, mas em geral eles levam muito tempo ou necessitam de muita memória, o que inviabiliza o seu uso em grandes instituições. Por isto, técnicas advindas da área de Inteligência Artificial (IA) são aplicadas. Segundo Barr et al. (1981), IA é a parte da ciência da computação voltada ao desenvolvimento de sistemas de computadores inteligentes, ou seja, sistemas que exibem características relacionadas à inteligência no comportamento do homem. Atualmente, a inteligência artificial já alcançou diversos campos dentro da sociedade, ela está presente na saúde, segurança, indústria, comércio, educação, entre outros.

Métodos de busca, como busca em largura, aprofundamento iterativo e a Busca A* (A estrela), são técnicas de IA que permitem chegar a uma solução ótima, mas que possuem elevada complexidade de tempo e espaço. Por isso, outras estratégias para solução desse tipo de problemas tem sido pesquisadas. Dentre elas, as meta-heurísticas de busca local têm se mostrado mais eficientes, como Subida da Encosta, Simulated Annealing (Tempera Simulada) e Algoritmos Genéticos (AG) (Russell e Norvig, 2010). Algoritmos Genéticos são algoritmos de otimização baseados nas teorias darwinianas. De acordo com Goldberg et al. (1989), AG são algoritmos de busca inspirados na teoria de seleção natural e genética, idealizada por Charles Darwin. Por possuírem robustez, serem generalistas e de fácil adaptação, é uma estratégia largamente estudada e aplicada em diversas áreas (Lucas, 2002).

O problema do *timetabling* também pode ser observado no Instituto Federal de Ciência e Tecnologia de Santa Catarina (IFSC). Atualmente, no IFSC Câmpus Lages não existe um sistema completo que realize a montagem dos horários. Utiliza-se um sistema interno criado pela Coordenadoria de Tecnologia da Informação e Comunicação (CTIC) do próprio câmpus, no qual os coordenadores preenchem as disciplinas a serem ministradas, informam os professores e a carga horária. A partir desse sistema é gerado um arquivo XML que é importado por um programa terceirizado responsável por gerar o quadro de horários. Após isso, caso necessário, os horários são encaminhados novamente aos coordenadores de curso para realizarem os ajustes finais, devido a incapacidade do sistema terceirizado resolver 100% dos conflitos em todos os cenários.

Como solução, foi desenvolvido por Bianquini e Silva (2020) um AG para resolução do problema de *timetabling*. Este trabalho teve como principais objetivos a modelagem e a implementação de um AG para solucionar o *timetabling* dentro do IFSC Câmpus Lages, utilizando duas abordagens. A primeira delas é a centralizada, na qual o processamento do AG é realizado utilizando apenas um computador com vários fluxos de execução (*threads*). Já a segunda, caracteriza-se como distribuída, dividindo o processamento do AG em diversas máquinas, também com vários fluxos de execução (*threads*) em cada uma delas, com o intuito de avaliar o desempenho computacional das diferentes

implementações.

Os resultados obtidos por Bianquini e Silva (2020) foram promissores. De acordo com eles, o AG obteve 100% de eficácia em ambos os ambientes, centralizado e distribuído, sendo esse último, o mais rápido, com o tempo médio de processamento do AG em 26 segundos, enquanto o centralizado levou 70 segundos.

As contribuições realizadas por Bianquini e Silva (2020) estão relacionadas à pesquisa pela melhor técnica meta-heurística para solucionar o problema de *timetabling* com eficiência, e sua implementação tanto em ambiente centralizado quanto em ambiente distribuído. Por outro lado, ainda existe a necessidade de alterar manualmente o arquivo XML gerado pelo sistema interno do Câmpus, de forma que seja possível sua execução no AG. Também percebe-se a falta de opções quanto ao agrupamento das aulas no modelo proposto, pois ele não permite indicar que se deseja ministrar as quatro aulas de uma disciplina em um mesmo dia, e também existem restrições quanto às disciplinas com cargas horárias ímpares.

Dessa forma, o objetivo deste trabalho é desenvolver novas funcionalidades, respeitando as restrições impostas pela instituição, de modo que não seja necessária nenhuma intervenção humana durante e após a execução do software, integrando todo o processo da criação dos horários em uma só ferramenta.

Para que esse objetivo seja alcançado foram definidos quatro objetivos específicos:

- Recuperar os dados diretamente do banco utilizado pelo sistema interno do IFSC Câmpus Lages, excluindo a necessidade de exporta-lo como XML.
- Permitir que o usuário escolha os cursos que serão incluídos no horário, possibilitando a montagem de horários separadas para cada período.
- Comparar o desempenho do novo sistema com o desempenho alcançado no trabalho de Bianquini e Silva (2020) e realizar novos testes com dados de outros semestres.
- Integrar todas as funcionalidades em uma só ferramenta, para que esta substitua o sistema terceirizado sendo atualmente utilizado no IFSC Câmpus Lages.

Este trabalho é uma pesquisa aplicada, pois visa o desenvolvimento de novas funcionalidades do AG implementado por Bianquini e Silva (2020), tendo como sua finalidade a implantação no IFSC Câmpus Lages. A abordagem do problema é quantitativa, uma vez que serão usadas técnicas estatísticas para comparação dos resultados gerados (qualidade da solução e tempo de processamento). Da perspectiva dos objetivos segue o caminho exploratório, visto que aprofunda-se no problema e na solução existente, para agregar novas funcionalidades e possibilidades de melhorias.

Inicialmente foi realizado um estudo geral sobre os temas *timetabling*, Algoritmos Genéticos e o seu uso na resolução desse problema, por meio de livros, artigos científicos e outros materiais disponibilizados na internet. Paralelamente, também foi realizada uma análise do sistema interno presente no Câmpus, buscando identificar quais tecnologias são utilizadas para obtenção, persistência e visualização dos dados, bem como o seu funcionamento, para possibilitar a integração com o AG desenvolvido por Bianquini e Silva (2020).

Na sequência foi realizado um estudo detalhado referente ao trabalho realizado por Bianquini e Silva (2020) para identificar novas necessidades que o AG deve abranger,

assim como novas funções que precisam ser implementadas. Esses aperfeiçoamentos são implementados no AG utilizando a linguagem de programação *Java* com o *Framework Spring*. Os dados são persistidos em uma base de dados *MySQL* utilizando o *Framework Hibernate*.

Por fim, foram efetuados os testes no novo AG, para avaliar os resultados no que diz respeito à eficiência de tempo e se estão respeitando as restrições da instituição. Esses testes foram realizados com dados de semestres passados do IFSC Câmpus Lages.

Este trabalho está dividido em cinco seções. A primeira seção conta com a introdução ao tema juntamente com os objetivos do trabalho. Na segunda seção apresenta-se o referencial teórico. A terceira seção descreve a modelagem e implementação das novas funcionalidades. A seção 4 expõe os resultados obtidos através dos testes realizados. Na quinta e última seção são descritas as considerações finais deste trabalho.

2. Referencial Teórico

Nesta seção são introduzidos os temas pertinentes ao desenvolvimento deste trabalho. A primeira subseção apresenta os temas *timetabling* e AG, além de trabalhos relacionados ao tema. A segunda subseção descreve a solução desenvolvida por Bianquini e Silva (2020), a modelagem do AG e suas etapas para a elaboração dos horários. A terceira subseção demonstra os testes realizados e os resultados obtidos. A quarta e última subseção trata das limitações encontradas no AG implementado por Bianquini e Silva (2020), bem como as melhorias necessárias.

2.1. Timetabling e Algoritmos Genéticos

O *timetabling* é caracterizado como um problema de escalonamento. Para Schaerf (1999), o problema consiste em organizar uma sequência de eventos em um determinado período de tempo que satisfaça um conjunto de restrições. Essas restrições estão divididas em *hard*, as quais devem ser satisfeitas com a finalidade de encontrar uma solução viável, e *soft*, as quais devem ser minimizadas, mas a sua existência não inviabiliza a solução. A solução que, além de ser viável (atende a todas as restrições do tipo *hard*), resolve todas as do tipo *soft*, é considerada uma solução ótima (McMullan, 2007). Um exemplo de restrição do tipo *hard* é a impossibilidade de alocar um professor para duas turmas no mesmo horário, inviabilizando a solução. Já a restrição classificada como *soft*, se dá, por exemplo, pela preferência do docente em lecionar no horário em que lhe convém, a qual, caso não seja atendida, não resulta em uma solução impraticável.

O problema do *timetabling* voltado a escolas e universidades, segundo Schaerf (1999), pode ser classificado em três diferentes variações, *school timetabling*, *course timetabling* e *examination timetabling*. O *school timetabling* refere-se à programação semanal dos horários de todas as turmas da escola, para que seja evitado que os professores e os alunos tenham duas aulas no mesmo horário. O *course timetabling* é utilizado em instituições de ensino superior, sendo uma programação semanal de horários de disciplinas dos cursos, evitando a sobreposição de disciplinas dos cursos que têm estudantes em comum. Por fim, o *examination timetabling* evita com que avaliações sejam sobrepostas para disciplinas que têm estudantes em comum, além de distanciar ao máximo possível as datas de avaliações de outras disciplinas, evitando assim, a sobrecarga.

O *timetabling* também pode ser visto em outras áreas como, por exemplo, na escala de horários dos enfermeiros, médicos e técnicos de um hospital (Cheang et al., 2003), bem como no planejamento de transporte público (Portugal et al., 2009).

As principais técnicas para a resolução do *timetabling* são baseadas em meta-heurísticas, as quais são conhecidas por encontrar soluções genéricas para problemas envolvendo otimização combinatória. Elas podem ser empregadas quando se desconhece um algoritmo determinístico capaz de solucionar com eficiência um dado problema. Dentre outras, pode-se destacar a Busca Tabu, *Simulated Annealing*, *Greedy Randomized Adaptive Search Procedure* (GRASP) e Algoritmos Genéticos.

Todas estas técnicas mantém informações sobre o estado atual da busca e avaliam os estados possíveis em seu entorno para definir o próximo estado a ser analisado. A diferença entre elas é como os novos estados são gerados e quantos estados são armazenados. Segundo Russell e Norvig (2010), Algoritmos Genéticos são como "uma busca de subida de encosta estocástica em que é mantida uma grande população de estados". Novos estados são gerados por mutação e por cruzamento, que combinam pares de estados da população". Ou seja, um AG pode ser definido como uma busca que se inicia por soluções aleatórias e realiza modificações, a fim de melhorar os resultados ou resolver a situação.

Um estudo realizado por Gonçalves et al. (2020) buscou comparar duas diferentes meta-heurísticas. *Simulated Annealing* e AG foram escolhidos para elaborar a alocação de horários no Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), Câmpus Crato. Os testes foram aferidos em condições análogas considerando as restrições impostas pela instituição. As comparações foram feitas em relação à: rapidez (A), qualidade (B), assegurabilidade (C) e quantidade das soluções geradas (D). Os resultados obtidos podem ser observados na Figura 1.

A assegurabilidade (C) indica o percentual de testes em que cada técnica encontrou pelo menos uma solução, no tempo pré-definido de quatro segundos. Desse modo, o *Simulated Annealing* encontrou a solução em 100% das situações, enquanto o AG apenas em 92%. O quesito qualidade (B) mostra o percentual de restrições (soft constraints) satisfeitas pela solução encontrada (100% representa uma solução ótima). Nesse quesito, o *Simulated Annealing* encontrou a solução perfeita, atingindo o percentual de 100%, já o AG alcançou a marca de 90%. No quesito tempo de processamento (A) o AG obteve uma larga vantagem em relação ao *Simulated Annealing*, sendo 6,6 vezes mais rápido para encontrar a solução. No quesito quantidade de soluções encontradas (D), o AG também obteve melhores resultados, gerando sete soluções, enquanto que o *Simulated Annealing* conseguiu gerar apenas uma. Esse quesito expressa a quantidade de soluções diferentes, geradas dentro do tempo limite de quatro segundos.

A partir dos resultados, os autores concluíram que a utilização do AG é mais interessante para contextos maiores, que, por serem mais complexos, são resolvidos mais facilmente pelas diferentes soluções que o AG é capaz de encontrar. O fato do tempo estipulado ser de apenas quatro segundos pode ter impactado negativamente o AG nos quesitos de assegurabilidade e qualidade, o qual pode obter resultados melhores se o limite de tempo for maior.

Apesar do ganho nos quesitos de rapidez e quantidade de soluções geradas pelo

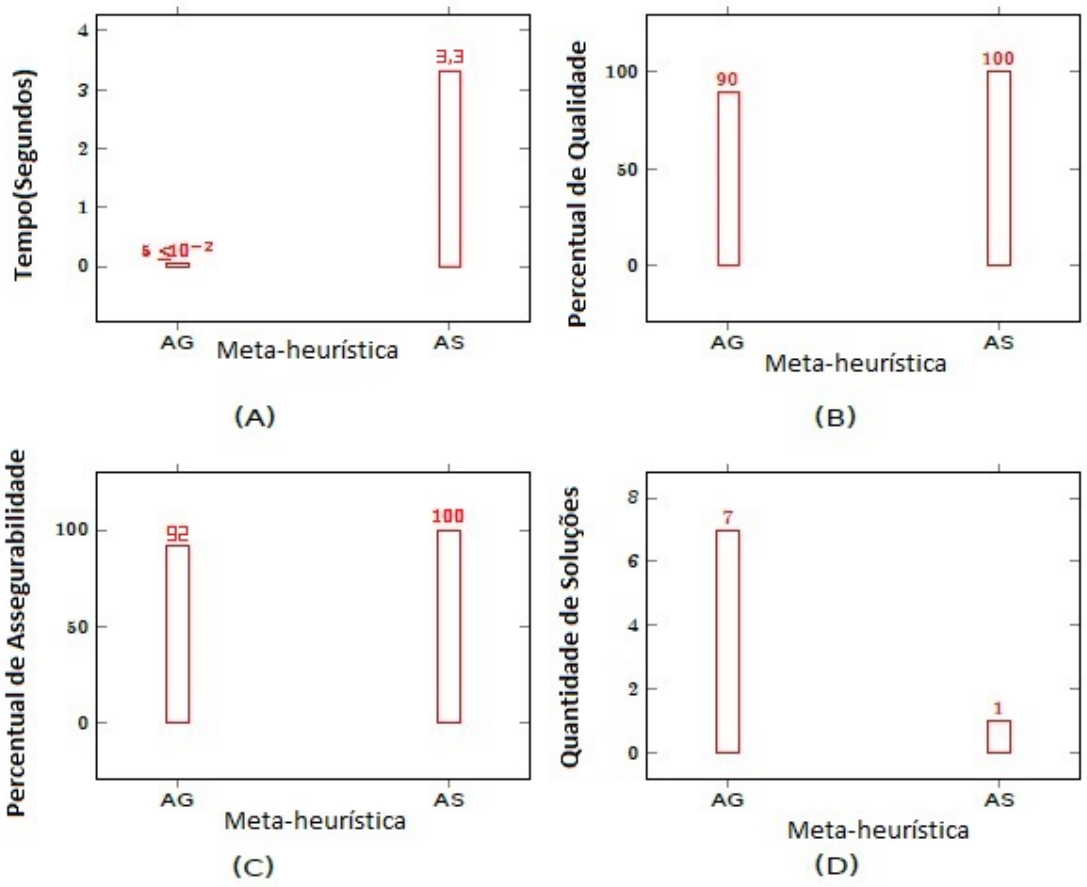

Figura 1. Resultados dos testes quanto a rapidez (A), qualidade (B), assegurabilidade (C) e quantidade de soluções geradas (D).
Fonte: Gonçalves et al. (2020)

AG ser mais expressivo ao ganho do *Simulated Annealing* em assegurabilidade e qualidade, os autores optaram por fazer uso do segundo, ressaltando o contexto em que se encontra o IFCE Câmpus Crato. Conforme exposto por Gonçalves et al. (2020), o AG é mais indicado para ambientes com maiores níveis de complexidade e escala, bem como para instituições que precisem lidar com uma demanda mais alta de flexibilidade, a fim de alcançar a melhor solução para todos os cenários.

Visando resolver problemas de *timetabling* em outras áreas, de Faria Passos et al. (2017) desenvolveram um AG para elaborar o quadro de horários do Simpósio Científico de Pesquisa Operacional, organizado pela Sociedade Brasileira de Pesquisa Operacional (SOBRAPO). Diferentemente do *timetabling* educacional, este possui algumas características específicas, como por exemplo: o vínculo entre o palestrante e o tema a ser abordado, a preferência pela concentração de temas similares e a preocupação de se ter palestras com temas correlacionados acontecendo no mesmo intervalo de tempo, podendo despertar interesse no mesmo público.

Para o desenvolvimento do AG, hipóteses e simplificações foram adotadas perante o problema em questão. Os testes foram realizados utilizando quatro taxas de mutação, 1%, 3%, 5% e 10%. Os resultados obtidos em todas os experimentos com diferentes ta-

xas de mutação obtiveram êxito, conseguindo descobrir uma solução logo nas primeiras iterações, sendo passada adiante até a finalização do AG ou até uma melhor ser encontrada, o que demonstra a capacidade dos AG para resolver este tipo de problema (de Faria Passos et al., 2017).

2.2. A Solução de Bianquini e Silva (2020)

A Figura 2 apresenta o fluxo de informações para a geração dos horários do IFSC Lages atualmente. O sistema (1) desenvolvido pelo IFSC Câmpus Lages possibilita o cadastro dos cursos, disciplinas, docentes e restrições associadas. Esse sistema gera como saída um arquivo no formato XML (2) com os dados armazenados no banco de dados (5). Este arquivo é importado por um sistema terceirizado (3) que elabora os horários e salva em um arquivo no formato PDF (4).

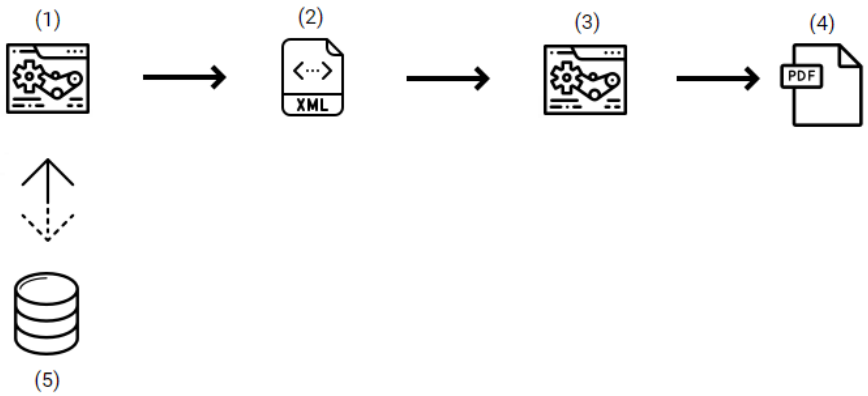

Figura 2. Representação do sistema utilizado pelo IFSC.

A solução implementada por Bianquini e Silva (2020) substitui o sistema terceirizado, mas segue o mesmo fluxo de informações apresentado. A principal diferença, é que após a leitura do arquivo XML e seu processamento pelo AG, é gerado um arquivo JSON com o quadro de horários.

Para que os dados contidos no arquivo XML pudessem ser interpretados pelo AG desenvolvido por Bianquini e Silva (2020), foi necessário convertê-los para uma nova representação desses dados, criado com base no modelo proposto pela Competição Internacional de *Timetabling* (ITC), pois da maneira como estavam organizados, eles impactariam negativamente no processamento do algoritmo. A Figura 3 apresenta o Diagrama de Classes usado pelo AG desenvolvido por Bianquini e Silva (2020).

Com o modelo de dados pronto, criou-se o modelo do AG com o objetivo de gerar os horários do IFSC Câmpus Lages respeitando as seguintes restrições (Bianquini e Silva, 2020):

1. Turmas com duas aulas no mesmo período;
2. Turmas com aulas de outras turmas;
3. Disciplinas com aulas excedentes ou faltantes;
4. Conflito de horários entre professores;
5. Indisponibilidade dos professores.

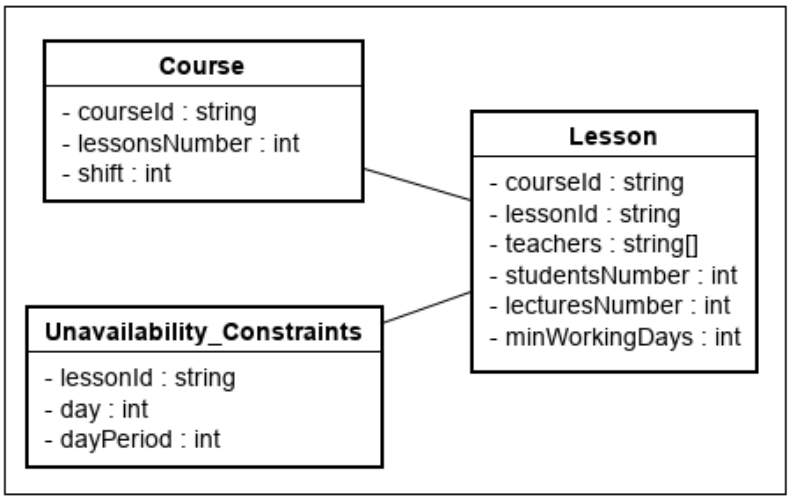

Figura 3. Relação do modelo adaptado por Bianquini e Silva (2020).
Fonte: Bianquini e Silva (2020).

Com o propósito de atender a essas restrições, o cromossomo, que é a estrutura de dados capaz de representar uma solução final do problema, foi elaborado como mostra a Figura 4. Cada turma é representada por dez posições do cromossomo, e cada posição simboliza um período (dia da semana e horário). Os números alocados dentro de cada posição do vetor correspondem ao identificador de cada uma das disciplinas, as quais estão organizadas em duas aulas, como comumente ocorre no IFSC Câmpus Lages. A forma como a estrutura do cromossomo está organizada garante que a primeira restrição não ocorrerá, dispensando sua verificação na função de avaliação, tornando o AG mais eficiente (Bianquini e Silva, 2020).

Nos casos em que disciplinas com horários ímpares são lecionadas, elas são agrupadas com a finalidade de formar uma única aula par, e sendo organizadas da seguinte maneira: enquanto uma disciplina inicia-se junto com o semestre e estende-se até a metade dele, a outra começa na metade e só termina junto com o semestre (Bianquini e Silva, 2020).

A execução do AG implementado por Bianquini e Silva (2020) foi subdividida em 5 etapas, são elas: inicialização, avaliação, seleção, cruzamento e mutação.

Primeiramente, na etapa de inicialização, é preenchido, de forma aleatória, as posições do cromossomo com base nas disciplinas de cada turma. Para cada turma, obtêm-se todas as disciplinas vinculadas a ela, e seus identificadores são inseridos no intervalo de posições respectivo à turma. Logo após, são realizadas trocas entre os identificadores em posições aleatórias desse intervalo. Esse procedimento visa garantir que as restrições dois e três sejam satisfeitas, evitando turmas com aulas de outras turmas e turmas com aulas excedentes ou faltantes.

Na avaliação, verifica-se a qualidade da solução de cada cromossomo e se o resultado foi alcançado. No AG em questão, a função de avaliação verifica duas restrições, uma *hard* e uma *soft*, pois a restrição (1) é solucionada na modelagem do cromossomo e as restrições (2) e (3) na etapa de inicialização, cruzamento e mutação. A restrição *hard* verificada refere-se aos conflitos de horários entre professores (4), no caso de um mesmo

	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 4. Representação da estrutura do cromossomo.
Fonte: Bianquini e Silva (2020).

professor lecionar duas disciplinas no mesmo horário. A restrição *soft* trata da indisponibilidade de professores (5), quando os mesmos indicam os dias ou horários em que não gostariam de lecionar. Cada cromossomo é inicializado com uma pontuação pré definida, e toda vez que alguma restrição é violada esse valor é decrementado, em outros termos, quanto maior o valor, melhor a avaliação do cromossomo.

Na etapa de seleção, realiza-se a escolha dos cromossomos que são enviados à próxima geração. Essa seleção é realizada por meio de dois sub-processos: o elitismo e a roleta. No elitismo os cromossomos mais bem avaliados são enviados diretamente para a etapa de mutação e para a nova geração. Na roleta, pares de cromossomos pais são escolhidos aleatoriamente com base no resultado da sua função de avaliação, e são enviados à etapa de cruzamento.

Na etapa de cruzamento, os pares de cromossomos selecionados pelo método da roleta são usados para criar os cromossomos da nova geração. Para implementar essa etapa foi utilizado o método de Recombinação Ordenada, no qual a nova geração é criada a partir de uma sequência de valores de um dos genitores, mantendo a ordem de valores do outro genitor. Desta forma, garante-se que os filhos gerados não violarão a restrição 3.

O cruzamento funciona da seguinte maneira: Dois valores são gerados aleatoriamente para determinar os pontos de corte (3 e 7 no exemplo da Figura 5). Em seguida, os números presentes no intervalo desses pontos são copiados dos Genitores (P1 e P2) e inseridos na mesma posição nos Descendentes (C1 e C2). Logo após, para gerar o primeiro conjunto (R1), são utilizados os valores contidos depois do segundo ponto de corte do genitor P2. Ao atingir o final do cromossomo, é preciso voltar ao início do mesmo. Após esse processo, são removidos os valores contidos no intervalo dos pontos de corte do P1, obtendo-se os elementos dos conjuntos, como mostra a Figura 6.

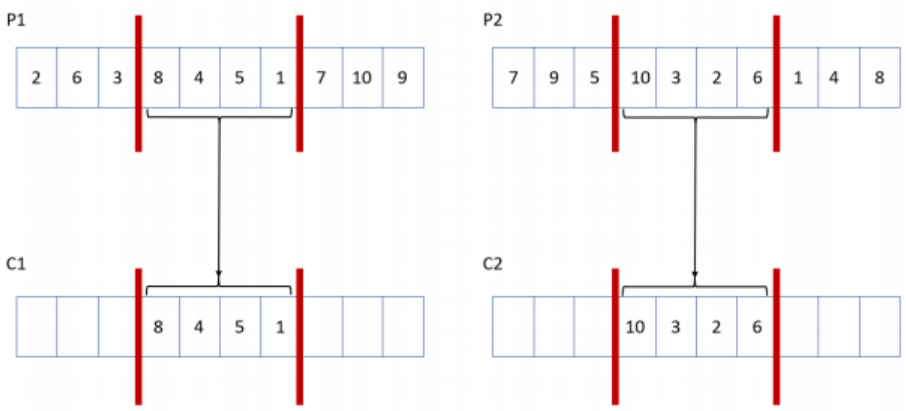

Figura 5. Exemplo de cruzamento parte 1
Fonte: Bianquini e Silva (2020).

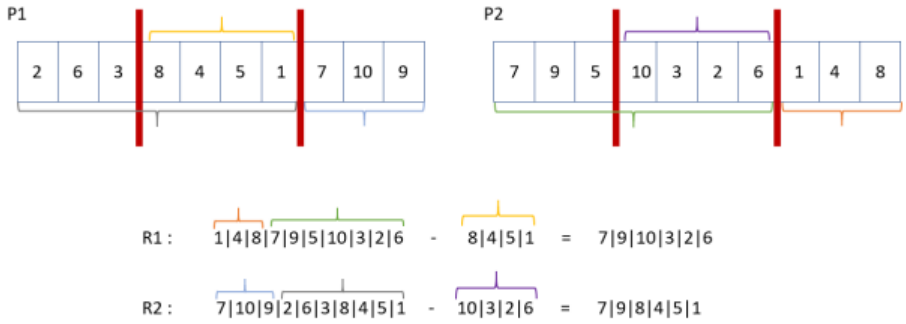

Figura 6. Exemplo de Cruzamento parte 2
Fonte: Bianquini e Silva (2020).

Para o segundo conjunto (R2) são realizadas as mesmas etapas. Ao final do processo, os valores dos conjuntos R1 e R2 são inseridos nos descendentes C1 e C2 respectivamente, mantendo os valores inseridos anteriormente, como mostra a Figura 7.

Figura 7. Exemplo de Cruzamento parte 3
Fonte: Bianquini e Silva (2020).

Na etapa de mutação são alterados aleatoriamente dois genes, a partir de um parâmetro que descreve a porcentagem dessa etapa ocorrer. De modo semelhante ao que acontece na inicialização, os valores são trocados de posições aleatoriamente dentro de uma mesma turma, mas essa troca é realizada apenas uma vez. Ao final desse processo, obtém-se uma nova geração a partir dos cromosossomos que foram selecionados, cruzados e modificados. O processo é repetido até que se encontre uma solução desejável ou atinja o valor limite de iterações.

O AG desenvolvido por Bianquini e Silva (2020) pode ser utilizado para elaborar os horários de um curso ou de vários cursos simultaneamente. A segunda abordagem demanda mais tempo de processamento, devido ao espaço de busca crescer exponencialmente conforme a quantidade de cursos aumenta. Apesar disso, quando um resultado é retornado, ele apresenta a solução final do problema.

A execução do AG para formar os horários dos cursos individualmente é mais interessante, por ser mais simples, rápida e viabilizar seu processamento em ambiente distribuído. Entretanto, essa solução pode causar uma série de conflitos nos horários de aulas dos professores que lecionam em dois ou mais cursos. Para evitar esses conflitos e permitir que o AG se beneficie da paralelização de processamento, os autores elaboraram uma solução intermediária, agrupando em conjuntos os cursos com muitos professores em comum, como mostra a Figura 8. Assim, o espaço de busca é menor que elaborar todos os cursos de uma única vez, enquanto minimiza os conflitos entre professores que lecionam em mais de um curso.

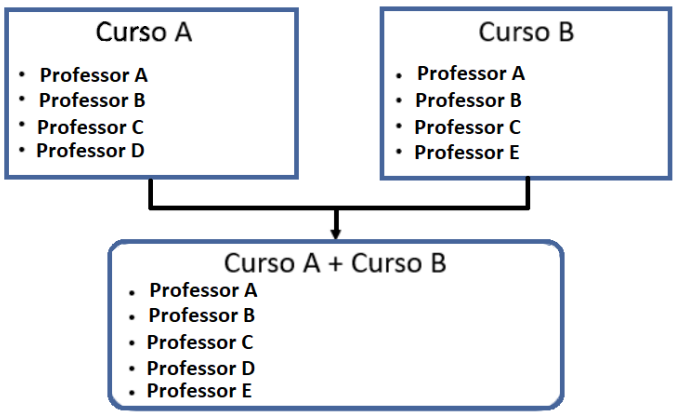

Figura 8. Agrupamento de cursos referente ao pré-processamento
Fonte: Bianquini e Silva (2020).

O algoritmo 1 descreve os passos necessários para gerar os horários em ambos os cenários de execução do AG (centralizado e distribuído). Os dados essenciais para o funcionamento do AG são:

- Arquivo de configuração com os valores dos parâmetros do AG;
- Arquivo XML com dados do IFSC, tais como professores, turmas, aulas, etc.;
- Arquivo com os endereços IPs dos computadores utilizados para processar o AG.

Os dados contidos nos arquivos são carregados pelo AG para dar início a sua execução. Após isso, ocorre a conversão dos dados presentes no XML para o modelo apresentado na Figura 3, permitindo a sua utilização pelo AG. O pré-processamento agrupa os cursos que possuem uma porcentagem pré-determinada de professores em comum e cada grupo é dividido entre os computadores que participam do processamento em ambiente distribuído. No final do algoritmo é gerada uma lista contendo os melhores cromossomos obtidos pela etapa de processamento em cada uma das máquinas quando a execução é distribuída, ou apenas um cromossomo quando é centralizada.

Algoritmo 1: Algoritmo 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

Restrições do tipo *hard* podem ser violadas após a execução do AG de forma distribuída, pois os resultados gerados para cada conjunto podem apresentar restrições envolvendo professores que lecionam em cursos alocados em diferentes conjuntos. Utilizando a técnica de busca em profundidade foi desenvolvido um algoritmo que une as soluções geradas para cada conjunto e é capaz de gerar uma grade de horários sem restrições do tipo *hard*. O funcionamento desse algoritmo ocorre da seguinte maneira: o cromossomo que possui a união de todas as soluções encontradas pelos AG de forma distribuída é inserido como a raiz da árvore e, a cada nível dessa árvore, uma restrição é solucionada, ou seja, o último nível apresenta nenhuma ou a menor quantidade de restrições violadas.

Os algoritmos que realizam o pré-processamento, a execução do Algoritmo Genético e o pós-processamento, bem como maiores detalhes sobre eles, podem ser encontrados no trabalho de Bianquini e Silva (2020).

2.2.1. Resultados Alcançados

Inicialmente, os testes realizados por Bianquini e Silva (2020) tiveram como objetivo estabelecer a combinação de parâmetros que conseguem gerar os melhores resultados para ambos os ambientes, centralizado e distribuído. Para tal, foram utilizados seis computadores do IFSC Câmpus Lages. Em ambos os ambientes o AG fez o uso de *threads* para diminuir seu tempo de processamento. Todos os computadores utilizados dispõe das

mesmas especificações: processador Intel Core i3 6100T 3.2 Ghz de 64 bits e 8 GB de memória RAM.

O primeiro teste buscou definir os parâmetros do AG para elaborar os horários de todos os cursos juntos. Para isso, foi efetuado um teste fatorial com os parâmetros: mutação, elitismo, cruzamento e tamanho da população, avaliados tanto individualmente como as suas interações. Para esse teste o valor 0 foi determinado como percentual de intersecção entre os professores, para que os horários dos cursos fossem montados em uma única máquina.

Cada parâmetro foi testado com os seguintes valores:

- 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.

Para o ambiente distribuído, o teste também abordou os mesmos parâmetros e valores. Nesse teste, a quantidade mínima de professores que os cursos precisam ter em comum para serem alocados em um mesmo conjunto foi definida em 60%, gerando quatro conjuntos. Após a execução dos testes, os valores das variáveis Resultado, Tempo de Processamento e Total de Gerações foram analisados por meio da Análise de Variância e do Teste de Tukey. Os resultados estão expressos na Tabela 1, que apresenta os melhores valores para cada um dos parâmetros em ambos os ambientes, levando em consideração as três variáveis.

Tabela 1. Melhores valores obtidos para cada parâmetro em ambiente centralizado e distribuído.

Parâmetros	Centralizado	Distribuído
Porcentagem de Mutação	50	5
Porcentagem de Elitismo	40	20
Porcentagem de Cruzamento	60	30
Porcentagem de Intersecção	0	60
Tamanho da População	400	200

Na execução em apenas uma máquina (centralizado), os parâmetros que apresentaram significância foram: *Tamanho da População*, *Mutação* e *Elitismo*, já o parâmetro *Porcentagem de Cruzamento* não apresentou uma significância considerável, desse modo, seu valor não exerce influência no resultado das variáveis finais e foi definido com o valor 60 por ser o valor intermediário entre os testados. No processamento realizado em mais de uma máquina, os parâmetros que apresentaram significância foram: *Tamanho da População*, *Mutação* e *Cruzamento*. Em ambiente distribuído o parâmetro que não exerce influência no resultado final das variáveis foi o *Elitismo*, o qual teve seu valor definido em 20 por ser o intermediário entre os testados.

Após identificar os melhores parâmetros para as duas abordagens, foram realizados testes de comparação entre elas, nas suas configurações ótimas, a fim de encontrar a melhor abordagem.

A Figura 9 mostra a comparação desses resultados, após a realização de dez replicações, utilizando as configurações identificadas anteriormente. Como demonstrado

no gráfico à esquerda, as duas execuções encontraram a solução perfeita (1500) em todos os testes realizados. Mas ao comparar o tempo de processamento, visualizado no gráfico à direita, é possível perceber a diferença. A execução distribuída se mostrou mais veloz em comparação com a centralizada.

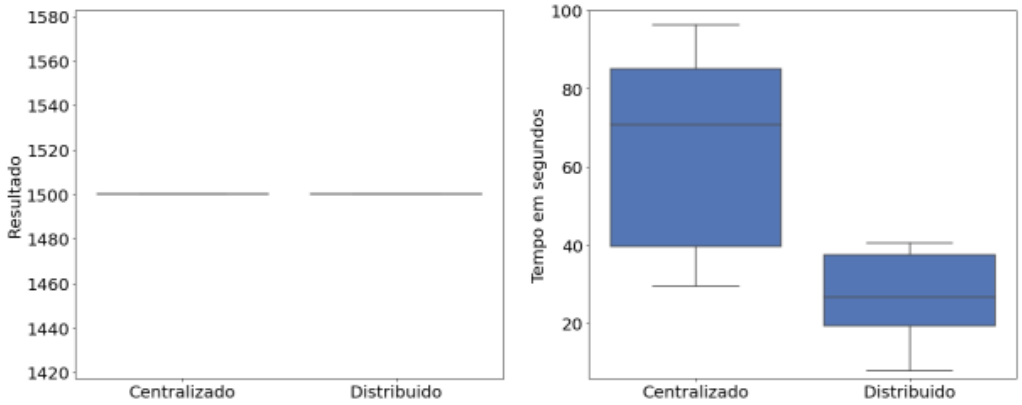

Figura 9. Diagrama de caixas comparando a solucão centralizada com a distribuída

Fonte: Bianquini e Silva (2020).

2.2.2. Limitações e Melhorias Necessárias

Após analisar o trabalho desenvolvido por Bianquini e Silva (2020), pôde-se identificar as principais limitações do sistema desenvolvido e as melhorias que poderiam ser feitas, conforme descrito a seguir.

Os dados referentes aos professores, turmas e disciplinas estão armazenados em um Banco de Dados (BD) *MySQL* conectado ao sistema do IFSC Câmpus Lages. Para que o AG implementado por Bianquini e Silva (2020) receba como entrada esses dados, eles precisam, antes, ser exportados para o formato XML, ocasionando mais etapas no processo de escalonamento dos horários. Além disto, alguns ajustes devem ser feitos manualmente neste arquivo para que o AG possa ser executado, o que dificulta a utilização do sistema por usuários que não conheçam a estrutura do AG. Para diminuir a quantidade de etapas, bem como o tempo da geração dos horários e evitar a necessidade de intervenção do usuário, a solução é realizar a conexão do AG diretamente com o BD, eliminando a etapa de exportação do XML.

Os cursos que possuem quantidade de aulas ímpares por período não podem ter seus horários gerados pelo AG implementado por Bianquini e Silva (2020), pois o cromossomo foi modelado levando em consideração o contexto geral da instituição, que apresenta em sua maioria cursos com aulas pares. Para permitir a partipação dos cursos com aulas ímpares, são necessárias alterações na modelagem do cromossomo.

O sistema terceirizado usado atualmente pelo IFSC Câmpus Lages, bem como o AG implementado, geram como saída um arquivo do tipo PDF e JSON, respectivamente, dificultando a integração com outros sistemas, como o sistema que faz o agendamento das salas de aula e laboratório do câmpus. Dessa forma, a solução é salvar os dados gerados

pelo cromossomo no BD, possibilitando a integração de novas ferramentas que venham a surgir posteriormente, assim como a alocação de laboratórios.

A solução desenvolvida por Bianquini e Silva (2020) foi exposta a testes a fim de verificar sua eficácia na elaboração dos horários. Esses testes foram realizados com os dados de um único semestre, para encontrar a solução ótima. Porém, é necessário a execução de mais testes com dados de diferentes semestres para validar a eficiência do AG.

3. Modelagem e Implementação do Sistema

Nesta seção são apresentadas as melhorias implementadas para que o sistema proposto nesse trabalho substitua o sistema terceirizado utilizado no IFSC Câmpus Lages. A primeira subseção descreve a conexão do AG ao BD eliminando a necessidade do XML. Na segunda subseção é apresentada a interface criada para permitir que usuários sem conhecimentos na área de programação de computadores possam usar o sistema.

3.1. Conexão com o Banco de Dados

O IFSC Câmpus Lages utiliza uma base de dados relacional criada no *MySQL* (versão 8.0.26), que contém os dados dos professores, disciplinas, cursos e ambientes. Estes dados são exportados pelo sistema do IFSC e salvos em um arquivo XML, que é importado pelo sistema desenvolvido porBianquini e Silva (2020). O primeiro objetivo deste trabalho é possibilitar a coleta de dados diretamente do banco de dados dispensando a geração e leitura do arquivo XML como ocorria com a versão anterior do sistema. Mas para que isto seja possível, alguns ajustes são necessários em função das diferenças entre os modelos.

A Figura 10 apresenta a estrutura do BD através do modelo relacional.

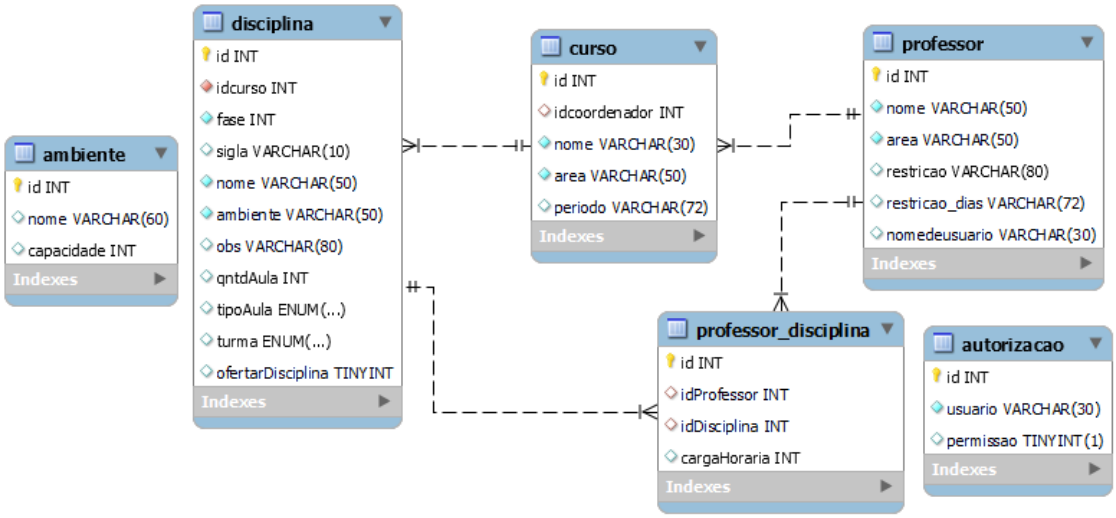

Figura 10. Modelo relacional

A tabela *Ambiente* representa as salas e laboratórios que podem ser usados. Ela não possui relação com nenhuma outra tabela e não é utilizada nesse trabalho. A tabela *Autorização* contém os dados que indicam quais funções cada usuário pode realizar no

sistema do IFSC e também não são considerados no trabalho atual. Na tabela *Professor*, os dados fazem referência aos professores que ministram alguma aula no semestre atual. Além dos dados básicos de um professor, ela possui a coluna *Restrição*, que registra o porquê o professor não pode dar aula em determinado dia, e a coluna *restricao-dias*, que apresenta os horários que os professores estão disponíveis para ministrar aulas. Esta informação é representada por uma sequência de números binários alocados em um Varchar. O número 1 indica o horário que o docente está disponível e o número 0 quando ele não está. Para cada dia existem doze números, representando quatro horários de manhã, quatro horários à tarde e quatro horários à noite, de segunda a sabádo. A tabela *curso* contém os dados referentes aos cursos ofertados, e está relacionada ao professor que é o coordenador do curso. A tabela *Disciplina* contém as disciplinas de todos os cursos cadastrados. A coluna *TipoAula* armazena um dos quatro valores: *simples*, *dupla*, *tripla* ou *quádrupla* que se referem ao número de aulas consecutivas, sendo simples uma aula, dupla duas aulas, tripla três e quádrupla quatro aulas, e a coluna *ofertar-disciplina* representa se essa disciplina está ativa. Por fim, a tabela *professor-disciplina* é a tabela que faz a ligação entre a disciplina ministrada e o professor que à ministra.

A Figura 11 apresenta o diagrama de classes, que representa como os dados são armazenados na estrutura do arquivo XML gerado pelo sistema do IFSC, o qual é importado pela versão desenvolvida por Bianquini e Silva (2020). Na versão desenvolvida no trabalho atual, os dados lidos do BD também são importados para esse mesmo modelo, a fim de reaproveitar o método que faz a conversão desse modelo para o modelo adaptado (Figura 3) desenvolvido por Bianquini e Silva (2020). Dessa forma, menos alterações foram realizadas nos demais módulos do projeto original realizado pela dupla.

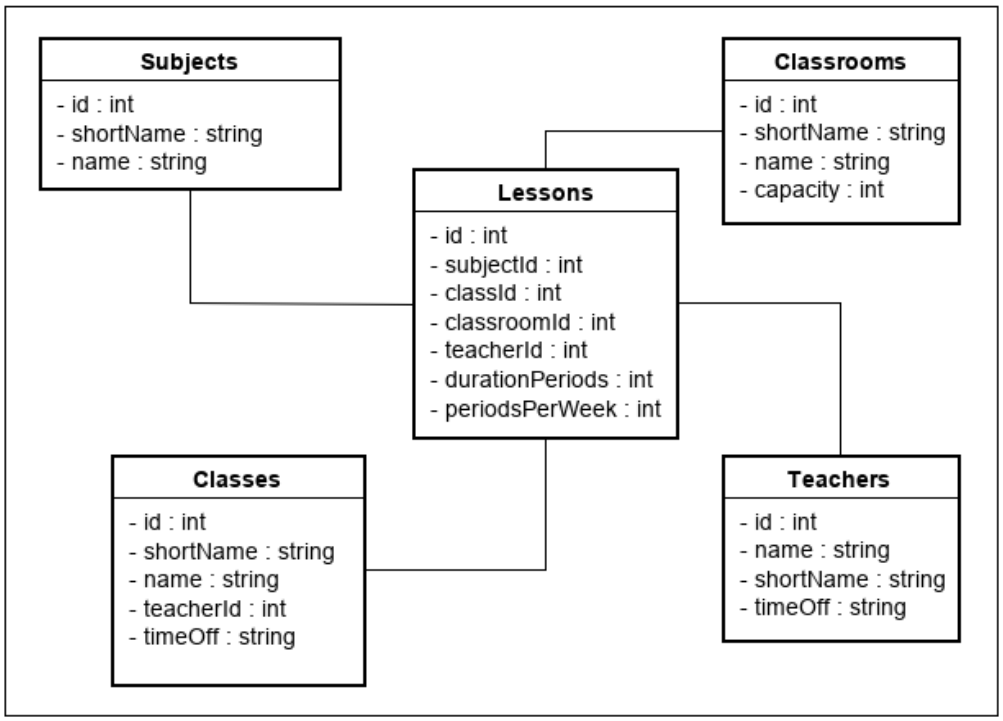

Figura 11. Modelo do IFSC
Fonte: Bianquini e Silva (2020).

O Quadro 1 relaciona os modelos do arquivo XML gerado pelo sistema do IFSC (Figura 11) e o modelo relacional do BD em que estes dados estão armazenados (Figura 10). O quadro mostra como cada atributo de determinada classe do modelo do IFSC é relacionado em uma coluna de determinada tabela do modelo relacional no banco de dados. Por exemplo: o atributo *ShortName* da classe *Subjects* é referente a coluna *Sigla* da tabela *Disciplina*.

Quadro 1. Comparação do Modelo do arquivo XML do IFSC com o Modelo Relacional

Modelo IFSC		**Modelo Relacional**	
Classe	**Atributo**	**Tabela**	**Coluna**
Teacher	Id	Professor	Id
	Name		Nome
	TimeOff		RestricaoDias
Classrooms	Id	Ambiente	Id
	Name		Nome
Subjects	Id	Disciplina	Id
	ShortName		Sigla
	Name		Nome
Classes	Id	Curso	Id
	Name		Nome
	TeacherId		IdCoordenador
	TimeOff		Período
Lessons	Id	-	- (um identificador próprio)
	SubjectId	Disciplina	Id
	ClassId		IdCurso
	TeacherId	ProfessorDisciplina	IdProfessor
	DurationPeriods	Disciplina	QntdAula
	PeriodsPerWeek		TipoAula

Como já foi citado, a decisão de converter os dados do BD para o modelo usado por Bianquini e Silva (2020), o qual é baseado na estrutura do arquivo XML usado por eles, possibilita a reutilização de todos os métodos relacionados à etapa de pré-processamento e do algoritmo genético. Entretanto, ela trouxe algumas dificuldades durante a importação dos dados, tendo em vista que não foi possível alterar a estrutura do BD pois ele é usado por outros sistemas internos do IFSC Câmpus Lages.

A conexão com o Banco de Dados do IFSC Câmpus Lages é realizada através do *Java Persistence API*, com implementação do *Hibernate 5.4*. O *Hibernate* faz o mapeamento objeto-relacional para Java (versão 8), ou seja, ajuda a representar dados da estrutura lógica de um banco de dados em classes dentro do projeto. Entre uma de suas funções, ele automatiza tarefas realizadas com o banco de dados a fim de simplificar o código de aplicação.

Para realizar a conexão e, posteriormente, o mapeamento e a troca de dados com o BD do IFSC Câmpus Lages, foi necessária a criação de um arquivo denominado *persistence.xml*, que contém os dados de configuração da unidade de persistência. Para isso,

neste arquivo, foi criada uma tag que contém o nome que identifica a aplicação, como por exemplo *persistence-unit name="TimeTabling-PU"*. Como propriedades é passado o caminho onde está armazenado o banco para a conexão e também é informado o usuário e a senha, para que seja possível acessá-lo. Por último é definido o dialeto utilizado para realizar a comunicação, nesse caso, o *MYSQL8Dialect*.

Após feita a conexão através do arquivo *persistence.xml*, foram modificadas as classes para possibiliar o mapeamento objeto-relacional dentro do projeto, como por exemplo, nas classes: *Teacher*, *Subject* e *Lesson*. Essas classes devem possuir uma tabela correspondente dentro do banco de dados, o mapeamento da classe é realizado através de anotações do JPA, como mostrado no algoritmo 2.

Algoritmo 2: Algoritmo que representa a classe *Teacher*

```
início
1   @Entity
2   @Table(name = ”professor”)
3   public class Teacher
4
5   @Id
6   @GeneratedValue(strategy = GenerationType.IDENTITY)
7   private int id ;
8
9   @Column(name=”nome”)
10  private String name;
11
12  @Column(name=”restricao-dias”)
13  private String timeoff;
fim
```

O algoritmo 2 representa a classe *Teacher* dentro do sistema, a qual faz ligação com a tabela *professor* no BD. Na linha 1, a tag *@Entity* representa uma nova entidade do Banco de Dados. Na linha seguinte, a tag *@Table* representa a qual tabela a classe está sendo referenciada, normalmente, a classe e a tabela possuem o mesmo nome, porém quando são diferentes, como nesse caso, é passado como parâmetro o nome da tabela que está no banco. Na linha 5 a tag *@Id* é usada para referenciar o identificador da classe. Por ser um valor que é auto incrementado conforme novos registros são realizados, na linha 6 isso é definido através da tag *@GenerateValue*, passando como parâmetro o identificador. Nas linhas 9 e 12 a tag *@Column* é usada para referenciar uma coluna do banco de dados com os atributos da classe e, por possuir nome diferente do que está cadastrado no banco, é passado como parâmetro o nome da coluna exatamente como está no BD. Após realizado os mapeamentos de todos os atributos são gerados os contrutores, os métodos *Getters* e *Setters* e o método *ToString*.

O algoritmo 3 apresenta a classe que realiza as pesquisas, inserções, exclusões e alterações no banco, nesse exemplo a classe *ProfessorDAO* faz a busca dos dados dos professores que estão cadastrados no banco de dados. Como nessa classe é necessário apenas ler os registros armazenados, o método *readAllProfessors()* realiza uma busca, e retorna para uma lista os professores cadastrados e os atributos requeridos. Na linha 7, a

string *findAllProfessorsJPQL* representa uma pesquisa dentro de uma query que retorna os dados requeridos.

Algoritmo 3: Algoritmo que representa uma pesquisa dentro do banco de dados (tabela professor)

início

```
1  public class ProfessorDAO
2  public ProfessorDAO()
34  public List Teacher readAllProfessores()
5  EntityManagerFactory entityManagerFactory =
     Persistence.createEntityManagerFactory( persistenceUnitName:
     "TimeTabling-PU");
6  EntityManager entityManager =
     entityManagerFactory.createEntityManager();
7  String findAllProfessorsJPQL = "SELECT id, nome, restricao-dias
     FROM professor";
8  List Teacher professors = new ArrayList();
9  Query query = entityManager.createNativeQuery(findAllProfessorsJPQL,
     Teacher.class);
10 professors = query.getResultList();
11 return professors;
```

fim

Saída: É retornado uma lista de professores com id, nome e restrição de dias

Assim como a classe *Teacher* no algoritmo 2, o mapeamento de todas as outras classes é realizado de forma semelhante, respeitando as relações e as dependências das colunas e atributos do BD e do modelo do IFSC, respectivamente, como apresentados na Tabela 2.

3.2. Seleção de Cursos

O objetivo principal do trabalho de Bianquini e Silva (2020) foi implementar um AG capaz de elaborar os horários dos semestres do IFSC e realizar testes para definir os valores dos parâmetros que levam aos melhores resultados, tanto na abordagem centralizada quanto na abordagem distribuída. Não fazia parte dos objetivos da dupla desenvolver interfaces gráficas que fossem interativas para que usuários comuns pudessem elaborar os horários, visto que era necessário conhecimento técnico pela necessidade de alterações dos dados no arquivo XML. Para que o AG desenvolvido por Bianquini e Silva (2020) possa ser usado por qualquer pessoa, é necessário que ele possua uma interface que possibilite a seleção dos dados.

A Figura 12 representa a interface inicial do sistema web, desenvolvido em HTML, CSS e JavaScript, que se comunica com o AG desenvolvido em Java. Nela são carregados, do BD, os cursos vigentes no semestre atual, tanto os cursos de graduação quanto os técnicos. O botão *Selecionar*, seleciona o curso destacado e busca as fases que estão sendo ofertadas no semestre, como mostra a Figura 13.

As fases carregadas do BD são mostradas abaixo da seleção de cursos e tem como objetivo permitir ao usuário que sejam selecionadas as fases que irão compor os horários

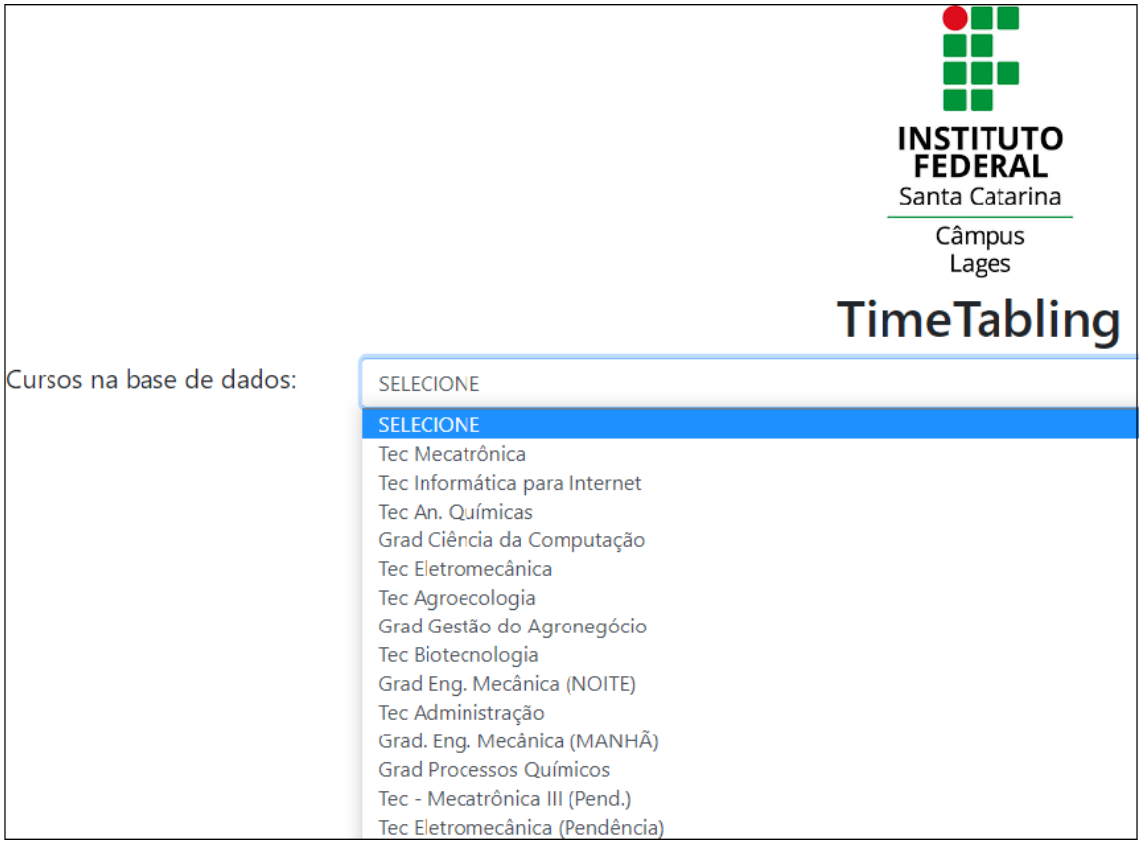

Figura 12. Interface do Programa (Cursos)

das aulas do semestre atual. Quando selecionadas, a partir do segundo botão *Selecionar*, são mostradas logo abaixo o curso e as fases escolhidas.

A Figura 14 apresenta a seleção de alguns cursos, com suas respectivas fases. Após escolher todos os cursos e turmas, o usuário pode clicar no botão *Gerar Hórario*, que é responsável pelo envio das turmas selecionadas ao AG para a elaboração dos horários. A seleção e a importação desses dados foi um processo custoso por não ser possível alterar a estrutura do BD, levando à necessidade de criar classes auxiliares para adaptar os dados contidos na base de dados ao sistema, possibilitando seu processamento pelo AG. Essas classes representam as tabelas do banco e foram realizadas alterações via código que leem os dados da base na mesma estrutura que o AG lia o arquivo XML, visto que esses dois possuem estruturas diferentes.

Esta interface cumpre o objetivo específico de integrar todas as funcionalidades em uma só ferramenta, sendo essa a leitura e a entrada dos dados, e também elimina a etapa de exportação do XML junto com os ajustes que eram realizados nesse arquivo. Na versão anterior, os horários não podiam ser gerados por usuários comuns, pois necessitava de conhecimentos prévios da estrutura geral da grade e pela necessidade de edições no arquivo XML. Nessa interface, usuários que não possuem conhecimentos técnicos podem gerar os novos horários, escolhendo os cursos, turmas e fases que o irão compor.

A nova interface também permite que os horários dos cursos sejam elaborados por turnos, diferentemente da versão anterior, onde não possuía a opção de escolher o curso

Figura 13. Interface do Programa (Fases)

Figura 14. Interface do Programa

e suas fases. Desse modo, o usuário possui maior flexibilidade e opções na escolha da

melhor maneira para gerar o horário.

Ao final de todo o processo de escolha dos cursos e das fases eles são apresentados na interface web, como mostra a Figura 15.

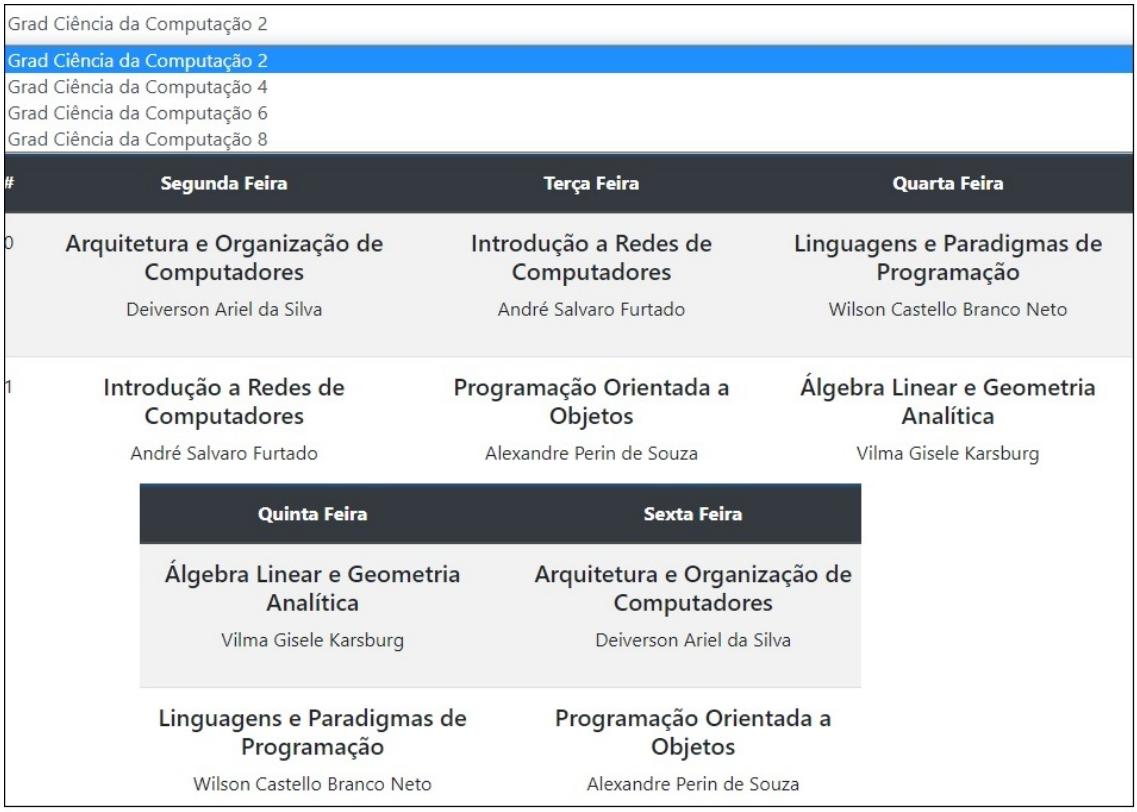

Figura 15. Apresentação da interface web do horário de uma turma

4. Resultados

Nessa seção são apresentados os resultados referentes ao tempo de processamento do AG utilizando a abordagem centralizada de Bianquini e Silva (2020). Os testes foram realizados de dois modos, o primeiro consiste em verificar o desempenho do AG ao processar os dados de todos os cursos juntos, enquanto que no segundo teste é verificado o tempo de processamento dividido por períodos (matutino, vespertino e noturno). O computador utilizado para realizar os testes possui as seguintes configurações: processador Intel Core i7 6700 3.4 GHz de 64 bits, 8 GB de memória de RAM 2133MHz, SSD de 1TB e placa de vídeo Nvidia Geforce GTX 1060. Os resultados são apresentados na Tabela 2.

Para o AG gerar os horários, além da leitura dos dados do banco do IFSC é necessário também a leitura de um arquivo TXT, contendo os valores referentes aos parâmetros de configuração do AG. Os valores utilizados para os parâmetros foram obtidos dos testes realizados por Bianquini e Silva (2020). Segundo eles, os melhores valores para cada parâmetro em ambiente centralizado são: porcentagem de mutação: 50, tamanho da população: 400, porcentagem de elitismo: 40, porcentagem de cruzamento: 60 e porcentagem de intersecção: 0.

Todos os testes alcançaram a solução perfeita, descartando choques de horários. O valor do resultado quando é processado todos os períodos juntos (1500), é devido a

Tabela 2. Comparação do tempo de processamento do AG, separado por períodos e com todos os períodos juntos.

Três períodos juntos			Três Períodos Separados						Total	
			Matutino		Vespertino		Noturno			
Replicação	Result	Tempo	Result	Tempo	Result	Tempo	Result	Tempo	Result	Tempo
1	1500	6.86	500	0.143	1000	0.88	1000	0.567	2500	1.59
2	1500	5.5	500	0.108	1000	0.105	1000	0.415	2500	0.628
3	1500	6.66	500	0.81	1000	0.124	1000	0.368	2500	1.302
4	1500	6.36	500	0.56	1000	0.81	1000	0.332	2500	1.702
5	1500	6.53	500	0.37	1000	0.116	1000	0.312	2500	0.798
6	1500	5.96	500	0.79	1000	0.217	1000	0.379	2500	1.386
7	1500	6.44	500	0.71	1000	0.158	1000	0.517	2500	1.385
8	1500	5.13	500	0.103	1000	0.131	1000	0.306	2500	0.54
9	1500	5.71	500	0.111	1000	0.116	1000	0.378	2500	0.605
10	1500	5.4	500	0.155	1000	0.114	1000	0.36	2500	0.629

quantidade de cursos presentes para gerar o horário, que nesse caso pode variar de 7 até 10 cursos. Se a quantidade de cursos selecionados for menor ou igual a 3, o valor de avaliação para a solução perfeita é 500. O valor de avaliação é 1000 quando a quantidade de cursos é maior que 3 e menor ou igual a 6, e o valor de 2500 representa a soma dos resultados dos três períodos (matutino, vespertino e noturno). Todas as replicações realizadas resultaram em horários sem restrições violadas.

Na Figura 16 é apresentado o resultado da comparação do processamento do AG em ambiente centralizado, com todos os cursos e com os cursos separados por períodos. A diferença entre esses dois modos de processamento fica evidente ao comparar o tempo de execução. O tempo médio de processamento dos três períodos juntos foi de 6.055 segundos, enquanto que ao gerar os horários por períodos, resultou no tempo médio de 1.056 segundos. Isso se deve ao fato de que, ao gerar os horários dos cursos separados por períodos o espaço de busca para encontrar a solução é menor em comparação com a montagem dos horários de todos os cursos, resultando em um menor tempo de processamento.

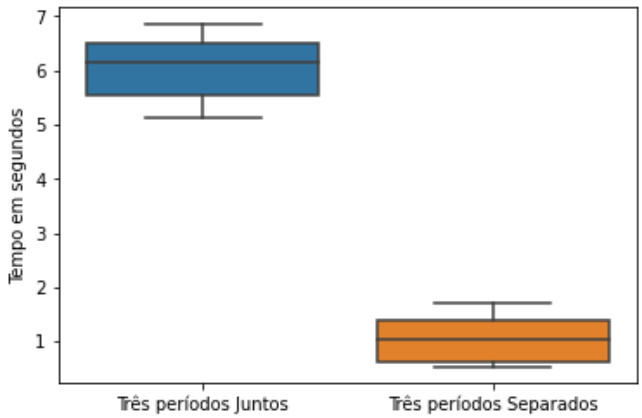

Figura 16. Diagrama de caixas comparando o processamento de todos os períodos juntos com todos os períodos separados em ambiente centralizado.

Finalmente, a Tabela 3 apresenta a análise de variância dos testes realizados, mostrando a comparação do tempo de execução do AG ao gerar os horários com todos os

cursos juntos e separados por períodos. Essa análise estatística tem por objetivo verificar se as médias de 2 ou mais grupos são significativamente diferentes ou se a diferença dos resultados deve-se a fatores aleatórios. Para isso, é calculado o valor de F, o qual é comparado com um valor tabelado. Como o valor calculado (442.34) é maior que o valor tabelado (4.41), conclui-se que há diferença no tempo médio de processamento entre os grupos que representam a elaboração dos horários de todos os cursos juntos ou separados por turno. Já o valor de P representa a probabilidade desta conclusão estar errada. Desse modo, o resultado obtido demonstra que existe uma diferença estatisticamente significativa em montar os horários utilizando a divisão de cursos por períodos.

Tabela 3. Análise de variância entre o processamento do AG com todos os cursos juntos e com os cursos separados por períodos, em relação ao tempo.

Fator	Soma dos quadrados	Média dos quadrados	F	P
Tratamentos	124.9250113	124.9250113	442.3420002	4.03444E-14
Residual	5.0835105	0.28241725		
Total	130.0085218			

5. Conclusão

O objetivo principal do trabalho de Bianquini e Silva (2020) foi implementar um AG capaz de elaborar os horários do IFSC e realizar testes para definir os valores dos parâmetros que levam aos melhores resultados, tanto na abordagem centralizada quanto na abordagem distribuída. Em função do tempo para realização do projeto, os autores não focaram no desenvolvimento de interfaces gráficas que possibilitassem o uso do sistema por parte dos usuários comuns, pois qualquer alteração que se desejasse, como por exemplo, elaborar os horários de um único turno, deveria ser feita diretamente nos dados do arquivo XML, o que requer conhecimentos específicos. Para que o AG desenvolvido por Bianquini e Silva (2020) possa ser usado por qualquer pessoa, é preciso que uma interface que possibilite a seleção dos dados seja implementada.

Este trabalho teve como objetivo desenvolver novas funcionalidades ao AG implementado por Bianquini e Silva (2020), para que fosse possível a sua utilização pelo IFSC Câmpus Lages, de modo que não seja necessária nenhuma intervenção humana durante e após a execução do software, integrando todo o processo da elaboração dos horários em uma só ferramenta, respeitando as restrições da instituição.

Para alcançar o primeiro objetivo específico deste trabalho, foi realizada a modificação do AG para ler os dados diretamente do banco de dados do IFSC, descartando a exportação do arquivo XML pelo sistema terceirizado. Para possibilitar essa alteração foram criados recursos extras para poder adequar os dados vindos do BD ao AG, uma vez que o BD da instituição é utilizado em conjunto com outros sistemas, impedindo qualquer modificação em sua estrutura.

Após concluir as modificações na leitura de dados no AG, foi desenvolvido uma interface WEB que possibilitou a customização da seleção dos cursos presentes no banco de dados. Uma vez selecionados, os dados dos cursos são enviados ao AG para serem processados em ambiente centralizado. Após o término é exibido os horários de cada turma separadamente.

A comparação do desempenho do AG desenvolvido por Bianquini e Silva (2020) com as modificações feitas neste trabalho foi o único objetivo específico atingido parcialmente, não sendo possível executar os testes em ambiente distribuído, pois as modificações necessárias para o AG conseguir ler os dados do banco foram custosas, consumindo uma parte considerável de tempo. Assim sendo, foi realizado os testes com os dados de outros semestres para averiguar a eficácia do AG, o qual elaborou a montagem dos horários respeitando todas as restrições da instituição.

Foram realizados testes para descobrir a melhor maneira para montar os horários dos cursos, levando em consideração todos os períodos juntos e os cursos separados por períodos. Os resultados podem ser observados na seção 4, e mostram que a diferença é considerada estatisticamente significante ao processar os cursos por períodos separados em relação a todos os períodos juntos, comprovada pelo teste de variância em relação ao tempo de processamento do AG.

Foi desenvolvida uma interface que centraliza e integra todas as funcionalidades, sendo elas: a leitura dos dados contidos na base, o envio para o AG, o processamento pelo mesmo e o resultado com o horário gerado na interface. Também permite ao usuário um maior controle da elaboração dos horários dos cursos de um semestre, dessa forma, descontinua o uso do sistema terceirizado que antigamente era responsável pela função de gerar os horários usando os dados que recebia através do arquivo XML.

Ainda existem algumas pendências que devem ser realizadas, como remodelar o AG desenvolvido por Bianquini e Silva (2020) para que os cursos de carga horária ímpar existentes no IFSC possam ser tratados e terem seu horário gerado com os demais cursos, a realização de testes deste projeto em ambiente distribuído e futuramente disponibilizar esse algoritmo para que ele possa ser implementado em instituições de ensino.

Referências

Barr, A., Feigenbaum, E. A., e Cohen, P. R. (1981). *The handbook of artificial intelligence*, volume 1. William Kaufmann.

Bianquini, I. e Silva, O. (2020). Algoritmo Genético distribuído para problema de Timetabling. Orientado por Wilson Castello Branco Neto. Lages, 2020. Trabalho de conclusão de curso (Ciência da Computação), IFSC (Instituto Federal de Ciência e Tecnologia de Santa Catarina). Lages, 2020.

Cheang, B., Li, H., Lim, A., e Rodrigues, B. (2003). Nurse rostering problems—a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460.

de Faria Passos, A. R., Diaz, E. Y. V., Tavera, M. J. M., Jequenesse, P. M., e Machado, M. N. (2017). Uma aplicação de algorítmos genéticos ao problema de timetabling. *Engevista*, 19(4):862–880.

Goldberg, D., David Edward, G., Goldberg, D., e Goldberg, V. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley Publishing Company.

Gonçalves, M. B., Freire, F. J. T., Lima, D. R., e Viana, T. B. (2020). Aplicação e comparação das meta-heurísticas arrefecimento simulado e algoritmos genéticos para alocação de horários universitários. *Revista Acta Kariri-Pesquisa e Desenvolvimento*, 2(1).

Lucas, D. C. (2002). Algoritmos genéticos: uma introdução. *Apostila referente a dis-*

ciplina de Inteligencia Computacional, Universidade Federal do Rio Grande do Sul, Brasil.

McMullan, P. (2007). An extended implementation of the great deluge algorithm for course timetabling. In Shi, Y., van Albada, G. D., Dongarra, J., e Sloot, P. M. A., editors, *Computational Science – ICCS 2007*, pages 538–545, Berlin, Heidelberg. Springer Berlin Heidelberg.

na Educação, D. (2021). Edtech: o que é, como funciona e qual a situação no brasil, 2021. Disponível em: `https://desafiosdaeducacao.grupoa.com.br/edtech-o-que-e-como-funciona-e-qual-a-situacao-no-brasil/`. Acesso em: 12 de janeiro de 2022.

Portugal, R., Lourenço, H., e Paixão, J. (2009). Driver scheduling problem modelling. *Public Transport*, 1:103–120.

Ross, P., Hart, E., e Corne, D. (2003). *Genetic Algorithms and Timetabling*, pages 755–771. Springer Berlin Heidelberg, Berlin, Heidelberg.

Russell, S. J. e Norvig, P. (2010). *Artificial intelligence : a modern approach*. Prentice Hall, Upper Saddle River, New Jersey.

Schaerf, A. (1999). A survey of automated timetabling. *ARTIFICIAL INTELLIGENCE REVIEW*, 13(2):87–127.