



UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS – CMPF
CURSO BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

ROBSON THIAGO BATISTA REGO

**INVESTIGAÇÃO DO *SOFTWARE* FET E IMPLEMENTAÇÃO DE NOVAS
RESTRIÇÕES DE TEMPO DE PROFESSORES**

PAU DOS FERROS

2020

ROBSON THIAGO BATISTA REGO

**INVESTIGAÇÃO DO *SOFTWARE* FET E IMPLEMENTAÇÃO DE NOVAS
RESTRICÇÕES DE TEMPO DE PROFESSORES**

Trabalho de Conclusão de Curso apresentado a Universidade Federal Rural do Semi-Árido – UFERSA, *Campus* Pau dos Ferros, para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Marco Diego Aurélio Mesquita

PAU DOS FERROS

2020

© Todos os direitos estão reservados a Universidade Federal Rural do Semi-Árido. O conteúdo desta obra é de inteira responsabilidade do (a) autor (a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei nº 9.279/1996 e Direitos Autorais: Lei nº 9.610/1998. O conteúdo desta obra tomar-se-á de domínio público após a data de defesa e homologação da sua respectiva ata. A mesma poderá servir de base literária para novas pesquisas, desde que a obra e seu (a) respectivo (a) autor (a) sejam devidamente citados e mencionados os seus créditos bibliográficos.

R343i Rego, Robson Thiago Batista.
Investigação do software FET e Implementação de
novas Restrições de Tempo de Professores / Robson
Thiago Batista Rego. - 2020.
47 f. : il.

Orientador: Marco Diego Aurélio Mesquita.
Monografia (graduação) - Universidade Federal
Rural do Semi-árido, Curso de Engenharia de
Computação, 2020.

1. Free Timetabling. 2. Restrição. 3.
Calendário escolar. I. Mesquita, Marco Diego
Aurélio, orient. II. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC's) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

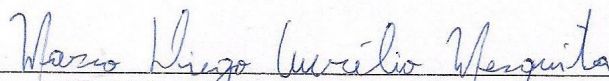
ROBSON THIAGO BATISTA REGO

**INVESTIGAÇÃO DO *SOFTWARE* FET E IMPLEMENTAÇÃO DE NOVAS
RESTRICÇÕES DE TEMPO DE PROFESSORES**

Trabalho de Conclusão de Curso apresentado a Universidade Federal Rural do Semi-Árido – UFERSA, *Campus* Pau dos Ferros, para obtenção do título de Bacharel em Engenharia de Computação.

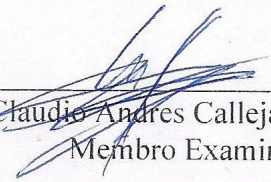
Defendido em: 05 / 02 / 2020.

BANCA EXAMINADORA

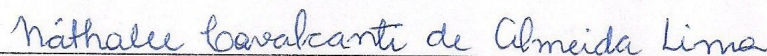


Prof. Me. Marco Diego Aurélio Mesquita (UFERSA)

Presidente


Prof. Dr. Claudio Andres Callejas Olguin (UFERSA)

Membro Examinador



Prof^a. Dra. Náthalee Cavalcanti de Almeida Lima (UFERSA)

Membro Examinadora

Aos meus pais, Rosano Rego Gonçalves e Betânia Maria Moreira Batista Gonçalves, por estarem ao meu lado e lutarem por educar e apoiarem seus filhos nas horas mais precisas.

AGRADECIMENTOS

A todos da minha família, por estarem ao meu lado nos momentos difíceis e por fazerem companhia.

A todos os meus amigos, pois a amizade é uma das coisas que me proporciona alegria.

A meus colegas que se tornaram amigos durante todo o ciclo acadêmico, e que demonstraram companheirismo.

Ao meu orientador Me. Marco Diego Aurélio Mesquita, por me ajudar na elaboração deste trabalho.

A todos os docentes, no qual tive a oportunidade de receber um pouco dos conhecimentos deles.

À UFERSA, por me ceder a oportunidade de conquistar mais uma graduação.

“Se eu vi mais longe, foi por estar sobre ombros
de gigantes”.

Isaac Newton

RESUMO

Nos últimos anos, problemas e soluções de preenchimento de horário de instituições de ensino vem sendo tema de muitas pesquisas. Ferramentas computacionais podem otimizar essas soluções, encontrando-as em poucos segundos, como os algoritmos de busca, otimização e heurístico, que vêm sendo utilizados para solucionar para tal problema. Uma solução satisfatória e eficaz torna-se difícil diante a complexidade e quantidade de restrições imposta para a solução desse problema, pois cada instituição contém suas restrições e particularidades. O *software Free Timetabling*, no qual tem a funcionalidade de processar o agendamento de horários de aulas automático de escolas de ensino primário e médio e universidades, vem sendo utilizado mundialmente em várias instituições de ensino para automatizar o processo de elaboração de horários do calendário escolar, já que de forma manual requer muito tempo. Com isso, foi proposto duas novas implementações de restrições que se aplicam na instituição de ensino Universidade Federal Rural do Semi Árido, *campus* Pau dos Ferros, e que não são contempladas no *software*.

Palavras-chave: Free Timetabling. Restrição. Calendário escolar.

ABSTRACT

In recent years, problems and solutions for filling the hours of educational institutions have been the objective of much research. Computational tools can optimize these solutions, being found in a few seconds, as the search, optimization and heuristic algorithms, which have been used to solve this problem. A satisfactory and effective solution becomes difficult in view of the complexity and amount of constraints imposed on a solution to this problem, since each institution contains these constraints and particularities. The Free Timetabling software, have the capacity to process or class schedules for elementary and high schools and universities, that is being used worldwide in several educational institutions to automate the school timetabling programming process, since it manual model requires a lot of time. With that, two new implementations of constraints have been proposed that apply the author's educational institution and which is not included in software.

Keywords: Free Timetabling. Constraint. School timetabling.

LISTA DE FIGURAS

Figura 1	–	Ilustração do Algoritmo de Busca.....	15
Figura 2	–	Exemplo de horário gerado.....	19
Figura 3	–	Tela principal dos Dados.....	21
Figura 4	–	Definição das horas.....	22
Figura 5	–	Tela das Atividades.....	23
Figura 6	–	Teste de QI de Einstein.....	26
Figura 7	–	Resultado das implementações.....	27
Figura 8	–	Aviso do FET ao ocorrer quebra de restrição <i>soft</i>	29
Figura 9	–	Conflitos de restrições <i>soft</i>	30
Figura 10	–	Calendário de aulas do professor Vinícius com restrição insatisfeita.....	31
Figura 11	–	Calendário de aulas do professor Cecílio com restrição insatisfeita.....	32
Figura 12	–	Aviso do FET ao ocorrer nenhuma quebra de restrição <i>soft</i>	33
Figura 13	–	Calendário de aulas do professor Vinícius com restrição satisfeita.....	34
Figura 14	–	Calendário de aulas do professor Cecílio com restrição satisfeita.....	35
Figura 15	–	Ilustração do método <i>fitness</i> da restrição Extensão máxima de dias por semana.....	37
Figura 16	–	Lista das restrições de tempo para professor.....	38
Figura 17	–	Detalhes da restrição Extensão máxima de dias por semana.....	39
Figura 18	–	Tela para adicionar a restrição Extensão máxima de dias por semana.....	39
Figura 19	–	Ilustração do método <i>fitness</i> da restrição Mínimo de horas diárias de descanso contínuas.....	40
Figura 20	–	Detalhes da restrição Mínimo de horas diárias de descanso contínuas.....	41
Figura 21	–	Tela para adicionar a restrição Mínimo de horas diárias de descanso contínuas.....	42
Figura 22	–	Tela de conflitos de restrições <i>soft</i>	42

LISTA DE TABELAS

Tabela 1	–	Comparação entre o processo manual e informatizado.....	17
----------	---	---	----

LISTA DE ABREVIATURAS E SIGLAS

- FET – Free Timetabling
- NP – Non-deterministic polynomial time (Tempo polinomial não determinístico)
- STP – School timetabling problem (problema de calendário escolar)

SUMÁRIO

1 INTRODUÇÃO	12
1.1 OBJETIVOS	14
1.1.1 Objetivo Geral	14
1.1.2 Objetivos Específicos	14
1.2 ESTRUTURAÇÃO DOS CAPÍTULOS	14
2 REFERENCIAL TEÓRICO	15
2.1 ALGORITMO DE BUSCA	15
2.2 ALGORITMO DE OTIMIZAÇÃO	16
2.3 TRABALHOS RELACIONADOS	16
2.4 SOFTWARE FET	18
2.5 ALGORITMO UTILIZADO	18
2.6 APRESENTAÇÃO DO FET	20
3 ESTUDO DE CASO	26
3.1 ESTUDO DO FET	28
4 DESENVOLVIMENTO E TESTES	36
4.1 RESTRIÇÃO EXTENSÃO MÁXIMA DE DIAS POR SEMANA	36
4.2 RESTRIÇÃO MÍNIMO DE HORAS DIÁRIAS DE DESCANSO CONTÍNUAS	40
5 CONSIDERAÇÕES FINAIS	43
6 REFERÊNCIAS BIBLIOGRÁFICAS	44
ANEXO I	46
ANEXO II	47

1 INTRODUÇÃO

Problemas e soluções de preenchimento de horário de instituições de ensino vem sendo tema de muitas pesquisas nos últimos anos. Tal problema conhecido por *School Timetabling Problem* (STP) é um desafio para funcionários de instituições que podem levar dias para alocar horários de professores, nos quais existem variantes restrições dependendo de cada instituição de ensino e nível escolar da mesma. Ferramentas computacionais podem otimizar essa solução, encontrando-a em poucos segundos. Algoritmos de busca e otimização vêm sendo utilizados para tal problema, porém existem diferentes restrições para cada tipo de STP fazendo com que tenha uma diversidade maior de utilização dos algoritmos (POULSEN, 2012).

Problemas de organização de horários são classificados por Schaerf (1999) em três tipos, “Horário Escolar”: um horário semanal que evita que um mesmo professor tenha duas aulas ao mesmo tempo. “Horário universitário”: o agendamento de disciplinas que minimiza que alunos tenham disciplinas diferentes no mesmo horário. “Cronograma de exames”: otimiza a distribuição de exames durante a semana para que alunos não tenham exames diferentes no mesmo horário.

As instituições de ensino no Brasil, especificamente as de ensino superior, tem um grande problema na elaboração de horários de aulas, pois geralmente em todo início de período letivo é necessário recriar uma grade de horas aulas. Um dos fatores para a recriação é a indisponibilidade de horário dos professores, aposentadorias e novas contratações (BARATA, *et. al*, 2010). A mesma será utilizada sem modificação em todo o período letivo, assim afetando diretamente o corpo docente e discente caso seja distribuída de forma incorreta e não respeitando restrições não essenciais. Este problema já vem sendo resolvido em algumas instituições de ensino utilizando algoritmos genéticos, como na Universidade Federal de Uberlândia, porém cada instituição contém restrições específicas (HAMAWAKI, 2005).

Em linhas gerais, o problema de alocação de salas, é conhecido como problema de tabela-horário, sendo estudado e discutido a mais de 50 anos (SCHAERF, 1999). Vale salientar que este tipo de problema pode ser aplicado a diversas situações como: Escalas de Trabalhadores, Escalas de Condutores de Veículos de Transportes entre outras demandas, (ROCHA, 2013). O problema de alocação de salas é caracterizado como um problema de otimização, e na concepção de Schaerf (1999) é classificado como um problema NP-completo de otimização combinatória, pois é um problema de tempo polinomial não determinístico. Nesse contexto, a utilização desses tipos de algoritmos, como o algoritmo genético, vem se

mostrando uma alternativa promissora para a solução do agendamento de horários escolares. Tendo em vista sua imensa dificuldade imposta pela grande quantidade de restrições, bem como sua necessidade de resolvê-las semestralmente, e que em muitas instituições a solução de problemas semelhantes ainda é feita de maneira manual, levando horas e por muitas vezes até dias, fazendo com que funcionários desperdicem seu tempo em uma atividade que poderia ser automatizada (HOLANDA, 2018).

Diante desse cenário, podem-se enumerar diversos fatores que dificultam uma solução ótima e trivial para tal problema, sendo estes: A quantidade de docentes disponíveis na instituição, a quantidade de salas, turmas e cursos que compartilham as mesmas disciplinas, disciplinas de uma mesma grade curricular semestral não podem coincidir no mesmo horário, dentre outras restrições. As restrições na literatura são classificadas em dois tipos: *hard* e *soft*. As restrições *hard* são as que não podem ser violadas em qualquer hipótese, pois causariam a inviabilidade da grade de horário. Já as restrições *soft*, são restrições desejáveis, que devem ser preferencialmente seguidas, sua violação, no entanto, não inviabiliza a solução (VIEIRA, F; MACEDO, H, 2011). Neste sentido, uma solução satisfatória e eficaz torna-se difícil diante da complexidade e quantidade de restrições imposta para a solução desse problema, pertencendo assim à classe dos problemas NP-difícil, (BARATA, *et al*, 2010; HAMAWAKI, 2005; MENEZES JÚNIOR, 2017; ROCHA, 2013).

Como este problema tem sido muito discutido na última década, trabalhos vêm sendo realizados com ajuda da inteligência artificial para automatizar o processo de construção do agendamentos de horários escolares. Algoritmos de otimização combinatória e heurísticos são bastantes populares para esse problema. Na década passada foi criado um *software open source* conhecido por *Free Evolutionary Timetabling* (FET) para solucionar problemas de grade horária escolar. Utilizando-se de um algoritmo genético, o FET era capaz de resolver problemas simples. Em 2007, houve uma melhoria e se aplicou um algoritmo heurístico para resolver problemas complexos com muitas restrições. Hoje é um *software* utilizado mundialmente (LALESCU, L; DIRR, V, 2018).

O mesmo problema e semelhantes ao preenchimento de horários do calendário escolar ocorrem na UFERSA *campus* Pau dos Ferros (HOLANDA, 2018). Com isso, este trabalho propõe investigar o *software open source* FET e as restrições de tempo necessárias na instituição de ensino Universidade Federal Rural do Semi Árido, *campus* Pau dos Ferros, e elaborar uma solução utilizando o FET para a distribuição de horários.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

- Investigar o *software open source Free Timetabling* (FET) e as restrições de tempo dos professores presente na Universidade Federal Rural do Semi Árido, *campus* Pau dos Ferros.
- Investigar uso de algoritmos diferentes para problemas de preencher tabelas respeitando restrições.

1.1.2 Objetivos Específicos

- Identificar todas as restrições de tempo possíveis dos professores da instituição.
- Implementar restrições ausentes do *software* FET que seja necessário aplicar na instituição e suas interfaces.
- Implementar e analisar desempenho de algoritmos para soluções de problemas de preenchimento de tabelas respeitando restrições.

1.2 ESTRUTURAÇÃO DOS CAPÍTULOS

No capítulo 2, serão apresentadas informações necessárias para entendimento do trabalho, como algoritmos associados, tal qual o de busca e de otimização. Também será discutido sobre trabalhos relacionados, e por fim será apresentado o *software* FET e o algoritmo utilizado no mesmo. No terceiro capítulo está o estudo de caso do trabalho, no qual serão apresentados os passos iniciais para entender como o FET funciona. No quarto capítulo está presente todos os resultados desenvolvidos e obtidos. E por último, o capítulo 5 em que são apresentadas as considerações finais do trabalho.

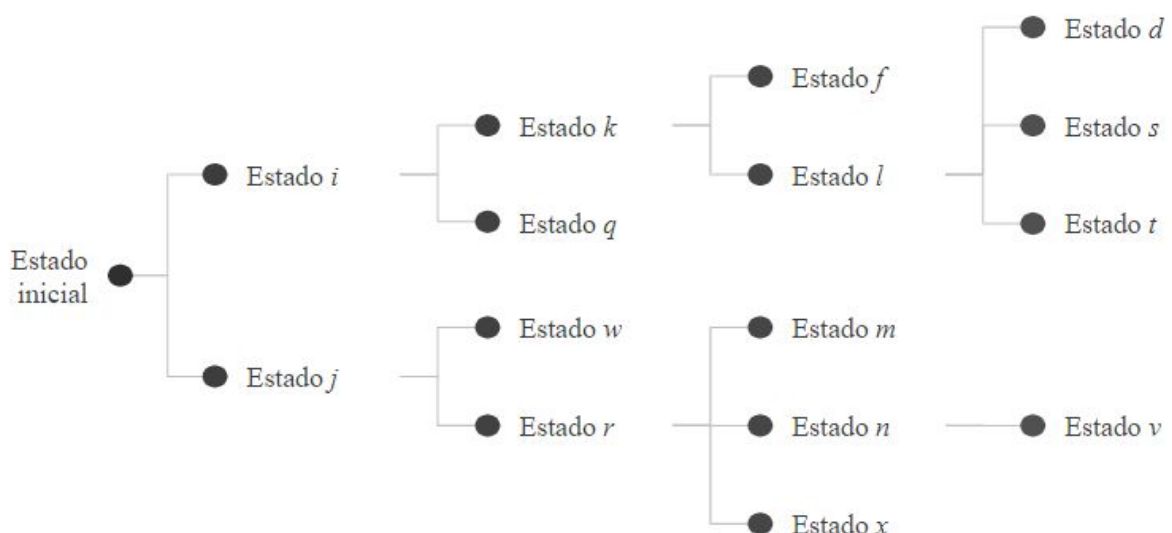
2 REFERENCIAL TEÓRICO

A compreensão deste trabalho exige o conhecimento de alguns conceitos e trabalhos relacionados. Entre os conceitos necessários para o entendimento do trabalho estão: algoritmos de busca, otimização e algoritmos genéticos. Alguns trabalhos relacionados são: Holanda (2018) que mostrou que a UFERSA *Campus* Pau dos Ferros tem problemas de alocação de salas e horários. Também Vieira e Macedo (2011) utilizou algoritmo genético para resolver o problema em questão.

2.1 ALGORITMO DE BUSCA

Algoritmo de busca é um método que usa o poder computacional para varrer um espaço de busca, iniciando de um estado inicial em busca de um determinado objetivo. Como ilustra a Figura 1, em cada estágio é feita uma verificação do estado, se um determinado conjunto de restrições for satisfeito, o espaço de busca é aumentado gerando novos estágios para serem verificados. Esse algoritmo é encerrado quando o espaço de busca chega ao seu limite ou quando um estado objetivo é encontrado. O problema de agendamento de horários escolares tem a característica de ter o espaço de busca muito grande, com isso este tipo de algoritmo não é indicado para este problema. Os algoritmos mais populares são de Busca em Profundidade e Busca em Largura (COPPIN, 2017).

Figura 1: Ilustração do Algoritmo de Busca



Fonte: Autor (2020)

2.2 ALGORITMO DE OTIMIZAÇÃO

Algoritmos de Otimização são usados em problemas que utilizam um grupo de variáveis, e tem a função de encontrar o melhor conjunto de valores para este grupo de variáveis. O problema abordado neste trabalho pode ser adequado a este tipo de algoritmo, os horários dos professores seriam as variáveis, e a organização dos mesmos seriam o melhor conjunto do grupo. Alguns algoritmos famosos deste tipo são a Busca Tabu, Têmpera Simulada e Algoritmo Genético. Algoritmo Genético utiliza otimização combinatória para solucionar problemas complexos utilizando um método baseado no processo de evolução observado na natureza. Indivíduos mais adaptados sobrevivem, no algoritmo, os estados mais próximos do objetivos são selecionados para gerarem sucessores que geralmente estarão mais próximos do objetivos, assim fazendo combinações com os melhores estados até chegar ao estado objetivo (COPPIN, 2017).

Para avaliar um estado no Algoritmo Genético, é utilizado uma função *fitness* que irá atribuir um valor para o estado, no qual em problemas de otimização, o estado com a maior avaliação *fitness* é o estado mais próximo do objetivo. São geradas populações (um conjunto de estados) para avaliar quais os estados são os mais próximos do objetivo. Em seguida, estes estados são combinados, ou seja, realizam um cruzamento. No Algoritmo Genético, o cruzamento é uma forma de gerar sucessores, pois unem características entre dois estados com um valor *fitness* bom e gera um novo estado com a avaliação *fitness* melhor. A mutação é outra forma de gerar um sucessor, no qual é escolhido um estado e o mesmo será modificado a fim de gerar um estado melhor. Os novos sucessores irão ser melhores do que os ancestrais, assim a nova população irá está mais próximo do objetivo (COPPIN, 2017).

2.3 TRABALHOS RELACIONADOS

A utilização de meios computacionais para otimizar trabalhos humanos já são comuns nos últimos anos, fazendo com que trabalhos sejam realizados para agilizar atividades humanas em ambientes universitários. Métodos computacionais foram estudados e indicados para otimizar trabalhos humanos que demoram até 3 dias para serem realizados, usando inteligência artificial para alocação de salas de aulas e laboratórios de forma rápida respeitando algumas restrições (HOLANDA, 2018).

Alocação de horários também é um problema popular nas instituições de todos os níveis de ensino. Tal problema é tão debatido que já existem *softwares* que geram soluções

rápidas que poderiam levar dias de forma manual, como mostra a comparação na Tabela 1, porém dependendo de cada instituição, este problema de alocação pode ser resolvido de formas diferentes, e o *software* pode não acobertar todas restrições possíveis (BARATA, *et. al.*, 2010), com isto, esse trabalho tem como objetivo adicionar novas restrições que são particular da instituição de ensino do autor, e que não contempladas no *software* FET. Algoritmo Genético foi utilizado para testes e foram obtidos resultados consideráveis para a alocação de horários em universidade, considerando algumas restrições do departamento de Computação da Universidade Federal de Sergipe (VIEIRA, F; MACEDO, H, 2011).

Tabela 1: Comparação entre o processo manual e informatizado

Quesito	Processo Manual	Processo Informatizado (FET)
Tempo gasto	Alguns dias	Poucos segundos
Qualidade da solução	Satisfatória	Boa
Facilidade de alteração	Difícil ou Impossível	Fácil
Dependência do usuário	Total	Somente para cadastro
Praticidade	Pouca ou nenhuma	Muita

Fonte: BARATA, *et. al.* (2010)

Este problema de alocação de horários pode ser resolvido de diversas formas, a melhor forma irá depender das restrições a serem consideradas. Algoritmos de otimização são populares nesse tipo de problema, Rocha (2013) utilizou o algoritmo de otimização *Greedy Randomized Adaptive Search Procedures* para resolver o problema de uma instituição de nível superior. A Busca Tabu também foi utilizado para resolver o problema de alocação de salas de aulas e horários. Em Subramanian, *et. al* (2006) utilizou-se a Busca Tabu para otimizar o problema de alocação de salas de aulas, já que o processo manual levaria muito tempo e podendo haver inviabilidade no resultado final. Com este algoritmo de otimização, foi capaz resolver adequadamente o problema e mostrou grande variabilidade na solução final, pois sempre gerou soluções satisfatórias e com baixo custo computacional. A Busca Tabu junto com a Coloração de Grafos foi capaz de solucionar o problema de alocação de horário do calendário escolar com o tempo computacional bastante curto e considerando restrições rígidas e flexíveis adicionais (BELLO, G. S, 2007).

2.4 SOFTWARE FET

O *software open source* inicialmente chamado de *Free Evolutionary Timetabling* (FET), o qual posteriormente ficou conhecido por apenas *Free Timetabling*, porém a sigla continuou a mesma, é um *software* disponível em vários idiomas, o qual tem a funcionalidade de processar o agendamento de horários de aulas automático de escolas de ensino primário e médio e universidades (LALESCU, L; DIRR, V, 2018).

O autor Liviu Lalescu iniciou este projeto em outubro de 2002 como trabalho final para obter seu diploma da universidade, utilizando algoritmo genético para solucionar problemas de agendamento, porém essa técnica foi ineficiente, pois resolvia somente horários fáceis. Depois de anos trabalhando neste projeto, Lalescu nunca encontrou uma forma eficiente do algoritmo genético para resolver problemas complexos. Em 2006, Volker Dirr contribuiu para o desenvolvimento do projeto, no qual no ano seguinte houve um grande avanço. Os desenvolvedores implementaram um novo algoritmo heurístico, baseado na troca recursiva de atividades para solucionar problemas de horários complexos, chegando a resolver em minutos. Hoje o FET se encontra na versão 5.42.3 e licenciado pela GNU Affero General Public License v3. Geralmente, o FET é capaz de resolver um horário complexo, que contém muitas restrições, em 5 a 20 minutos, segundo o autor. Para horários mais simples, pode levar um tempo menor do que 5 minutos, e se for muito simples, leva alguns segundos. Para horários extremamente difíceis, pode demorar mais tempo, resolvendo em algumas horas. O *software* é capaz de resolver problemas de alocação de horários e alocação de salas de aulas (LALESCU, L; DIRR, V, 2018; LALESCU, L; DIRR, V, 2020a).

2.5 ALGORITMO UTILIZADO

O algoritmo utilizado no FET para gerar agendamentos de horários automáticos é heurístico e simula um pouco o procedimento manual de agendamento. Tem como entrada um conjunto de restrições e de atividades (A_1, A_2, \dots, A_n) , que serão especificadas pelo o usuário, no qual uma atividade é constituída por turma, disciplina e professor. A saída do algoritmo é o conjunto de horários alocados respeitando as restrições e a quantidade de horários pré-estabelecidos (T_1, T_2, \dots, T_m) onde se dá pelo o número de dias vezes a quantidade de horários por dia (LALESCU, L, 2019a).

Figura 2: Exemplo de horário gerado

	All Activities				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	7 periodo Eletronica Analogica Leaut Ernano LEAUT	7 periodo Eletronica Analogica Leaut Ernano LEAUT	---	7 periodo Eletronica Analogica Leaut Ernano LEAUT	8 periodo Sistemas de T de D Segundo
08:50 - 09:45	7 periodo Eletronica Analogica Leaut Ernano LEAUT	7 periodo Eletronica Analogica Leaut Ernano LEAUT	---	7 periodo Eletronica Analogica Leaut Ernano LEAUT	8 periodo Sistemas de T de D Segundo
09:45 - 10:40	8 periodo Sistemas de T de D Segundo	9 periodo Sistemas Inteligentes Marco	10 periodo Logica Matematica Claudio	10 periodo Logica Matematica Claudio	---
10:40 - 11:35	8 periodo Sistemas de T de D Segundo	9 periodo Sistemas Inteligentes Marco	10 periodo Logica Matematica Claudio	10 periodo Logica Matematica Claudio	---
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	8 periodo Sistemas de Controle I Adller	8 periodo Sistemas de Controle I Adller	---	---	9 periodo Sistemas Inteligentes Marco
14:50 - 15:45	8 periodo Sistemas de Controle I Adller	8 periodo Sistemas de Controle I Adller	---	---	9 periodo Sistemas Inteligentes Marco
15:45 - 16:40	---	---	---	---	---
16:40 - 17:35	---	---	---	---	---

Fonte: Autor (2020)

A Figura 2 mostra um exemplo de horário gerado pelo *software*. No qual cada célula representa um horário (T_i) e é atribuída uma atividade. Uma atividade são aulas distribuídas na semana, no qual é um conjunto de mesma turma, disciplina e professor. O *split* é a quantidade de fragmentos de aulas divididas pela semana. Na figura citada, apresenta como exemplo várias atividades. Considere uma aula da turma “9 período”, da disciplina Sistemas Inteligentes, ministrada pelo professor Marco. Estas aulas estão distribuídas em 2 *split*: terça-feira nos horários (9:45-10:40) e (10:40-11:35), sexta-feira (13:55-14:50) e (14:50-15:45). A atividade (A_j) é considerado um *split*, ou seja, uma atividade dessa aula tem duração de duas horas por dia, e quatro horas por semana.

O algoritmo heurístico utilizado pelo FET é chamado pelo autor Liviu Lalescu de “*recursive swapping*” ou troca recursiva, e segundo ele se assemelha com o algoritmo cadeia de ejeção (*ejection chain*). Lalescu diz que este algoritmo se assemelha ao processo manual de agendamento de horários, pois em um estado vazio, o primeiro passo é organizar as atividades mais críticas, ou as mais difíceis em ser alocadas, como por exemplo, restrições *hard*, assim

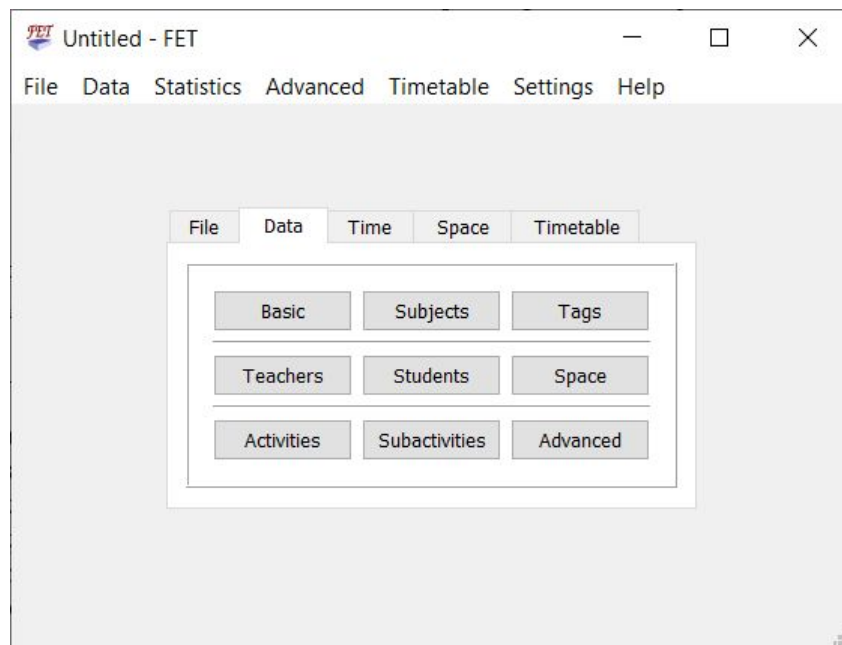
dando prioridades a elas. O algoritmo vai adicionando as atividades mais críticas em espaços de horários vazios que satisfaçam as restrições. Caso tenha mais de um espaço apropriado, é escolhido um aleatoriamente, e caso não tenha nenhum, o algoritmo procura um espaço de horário apropriado com a menor quantidade de atividades conflitantes, assim irá substituir uma das atividades conflitantes com a atual, e fará o processo recursivamente até encontrar um resultado com êxito. O limite de recursão é 2 vezes o número de atividades, esse número foi escolhido pelo autor do *software* pois foram obtidos resultados satisfatórios na prática. E para não executar processos de substituições repetidos, é utilizado a Busca Tabu para verificação. Quando o algoritmo não encontra um resultado que satisfaça todas as restrições, emitirá uma mensagem de conflito e sinalizará quais restrições não foram satisfeitas (LALESCU, L, 2019b).

Não foi investigada uma implementação própria do algoritmo utilizado no FET, porém foram implementados o algoritmo de busca em profundidade e o algoritmo genético para se resolver um problema de preencher tabela respeitando restrições, que está presente no capítulo 3. O objetivo dessas implementações foi ganhar experiência para avaliar a viabilidade do uso dessas estratégias utilizando um problema análogo ao que se deseja resolver.

2.6 APRESENTAÇÃO DO FET

O *software* FET está disponível para *download* no site oficial (LALESCU, L; DIRR, V, 2020b) e é da forma executável. Após criar um novo arquivo, deve-se adicionar dados para gerar o horário escolar. Como mostra a Figura 3, o usuário pode adicionar informações básicas, que são informações da instituição como nome e comentários, e a quantidade de dias por ciclo, que usualmente este ciclo é uma semana, então pode-se definir a quantidade de dias na semana e a quantidade de horários por dia (Figura 4).

Figura 3: Tela principal dos Dados



Fonte: Autor (2020)

Na aba de Dados pode-se definir as disciplinas, professores e marcações (que são opcionais). As *tags* (marcações) são uma forma de “marcar” disciplinas que tenham algo em comum, por exemplo, marcar as disciplinas que precisam de um laboratório de química em comum ou quadra esportiva, isto ajudará a terem horários exclusivos para uso daquele recurso naquela ocasião. O usuário pode adicionar informações dos estudantes, que será a turma/ano, grupos e subgrupos, também é possível adicionar dados do espaço físico como construções (blocos) e salas, porém quando é definido turma/ano dos estudantes não tem necessidade de adicionar salas de aulas, pois cada turma tem sua própria sala e não terá mais de uma aula no mesmo horário (LALESCU, L; DIRR, V, 2018).

Figura 4: Definição das horas

The screenshot shows a window titled "The hours of the day" with a close button and a help icon. At the top, there is a label "Number of periods (start hours) per day" and a text box containing the number "14". Below this is a grid of 8 rows and 8 columns. Each cell in the grid is labeled "Hour X" and contains a time range. The first two columns are filled with time ranges, while the remaining six columns are empty. The time ranges are as follows:

Hour 1	Hour 9	Hour 17	Hour 25	Hour 33	Hour 41	Hour 49	Hour 57
07:55 - 08:50	16:40 - 17:35						
Hour 2	Hour 10	Hour 18	Hour 26	Hour 34	Hour 42	Hour 50	Hour 58
08:50 - 09:45	17:35 - 18:40						
Hour 3	Hour 11	Hour 19	Hour 27	Hour 35	Hour 43	Hour 51	Hour 59
09:45 - 10:40	18:40 - 19:35						
Hour 4	Hour 12	Hour 20	Hour 28	Hour 36	Hour 44	Hour 52	Hour 60
10:40 - 11:35	19:35 - 20:30						
Hour 5	Hour 13	Hour 21	Hour 29	Hour 37	Hour 45	Hour 53	
11:35 - 13:55	20:30 - 21:25						
Hour 6	Hour 14	Hour 22	Hour 30	Hour 38	Hour 46	Hour 54	
13:55 - 14:50	21:25 - 22:20						
Hour 7	Hour 15	Hour 23	Hour 31	Hour 39	Hour 47	Hour 55	
14:50 - 15:45							
Hour 8	Hour 16	Hour 24	Hour 32	Hour 40	Hour 48	Hour 56	
15:45 - 16:40							

At the bottom of the window, there are two buttons: "Ok" and "Cancel".

Fonte: Autor (2020)

Após adicionar todos esses dados citados, é necessário definir as atividades, isso será os horários, no qual é um conjunto de professor, disciplina, turma e *tag* caso seja necessário. Como mostra a Figura 5, é necessário selecionar um professor que ministrará a aula, uma disciplina e a turma referente, e caso necessite, pode-se selecionar uma *tag*. A forma de como a atividade poderá ser dividida será definido pela quantidade de *slipt* (conjuntos de aulas divididas na semana), e a quantidade de aulas de cada *slipt* será definido em duração. Em estudantes pode-se colocar a quantidade de estudantes que a atividade limita, e é atribuído -1 para ilimitado (LALESCU, L; DIRR, V, 2018).

Figura 5: Tela das Atividades

Fonte: Autor (2020)

Na aba de Tempo (Figura 3), é onde se define as restrições de tempo das atividades, professores e estudantes. Também é possível definir pausas em horários. Algumas restrições básicas que são obrigatórias, no qual não dá direito ao usuário escolher, pois caso sejam insatisfeitas fugirá da realidade, como por exemplo, um par de atividades iguais no mesmo horário ou um professor ministrando duas atividades diferentes no mesmo horário. A seguir estão listadas as restrições de tempo do professor, turma e atividade presente no *software* (LALESCU, L; DIRR, V, 2018).

Restrições de tempo do professor:

- *Um professor não está disponível em um determinado horário.*
- *Máximo de dias por semana*
- *Mínimo de dias por semana*
- *Intervalos/Lacunas máximas por semana*
- *Intervalos/Lacunas máximas por dia*
- *Máximo de horas diárias*

- *Extensão máxima de horas por dia*
- *Máximo de horas diárias com uma tag de atividade*
- *Mínimo de horas diárias com uma tag de atividade*
- *Mínimo de horas diárias*
- *Máximo horas continuamente*
- *Máximo de horas continuamente com uma tag de atividade*
- *Mínimo de lacunas/intervalos entre um par ordenado de tags de atividades*
- *Trabalhar em um intervalo por hora, no máximo dias por semana*
- *Mínimo de horas de descanso*

Restrições de tempo da turma:

- *Uma turma não está disponível em um determinado horário*
- *Máximo de dias por semana*
- *Intervalos/Lacunas máximas por dia*
- *Intervalos/Lacunas máximas por semana*
- *Máximo de início na segunda hora*
- *Máximo de horas diárias*
- *Extensão máxima de horas por dia*
- *Máximo de horas diárias com uma tag de atividade*
- *Mínimo de horas diárias com uma tag de atividade*
- *Mínimo horas diárias*
- *Máximo horas continuamente*
- *Máximo de horas continuamente com uma tag de atividade*
- *Mínimo de lacunas/intervalos entre um par ordenado de tags de atividades*
- *Trabalhar em um intervalo por hora, no máximo dias por semana*
- *Mínimo horas de descanso*

Restrições de tempo da atividade:

- *Uma atividade tem um horário de início preferido*
- *Uma atividade tem um conjunto de horários de início preferidos*
- *Uma atividade tem um conjunto de horários preferenciais*
- *Um conjunto de atividades possui um conjunto de horários de início preferidos*
- *Um conjunto de atividades possui um conjunto de horários preferenciais*
- *Um conjunto de subatividades possui um conjunto de horários de início preferidos*

- *Um conjunto de subatividades possui um conjunto de intervalos de tempo preferenciais*
- *Dias mínimos entre um conjunto de atividades*
- *Dias máximos entre um conjunto de atividades*
- *Uma atividade termina no dia da turma*
- *Um conjunto de atividades termina no dia da turma*
- *Um conjunto de atividades tem a mesma hora de início (dia + hora)*
- *Um conjunto de atividades tem o mesmo dia inicial (a qualquer hora)*
- *Um conjunto de atividades tem a mesma hora inicial (todos os dias)*
- *Um conjunto de atividades ocupa o máximo de intervalos de tempo selecionados*
- *Um conjunto de atividades ocupa o mínimo de intervalos de tempo selecionados*
- *Duas atividades são ordenadas*
- *Duas atividades são consecutivas*
- *Duas atividades são agrupadas*
- *Três atividades são agrupadas*
- *Um conjunto de atividades não se sobrepõe*
- *Um conjunto de tags de atividades não se sobrepõe*
- *Máximo de atividades simultâneas de um conjunto em intervalos de tempo selecionados*
- *Mínimo de atividades simultâneas de um conjunto em intervalos de tempo selecionados*
- *Diferenças mínimas (horas) entre um conjunto de atividades*

Já na aba de restrições por espaço físico (Figura 3), resumidamente existem restrições para disciplinas e tags de atividades que permitem haver uma preferência por uma sala de aula. Os professores e as turmas têm restrições que podem limitar o máximo ou/e mínimo de aulas por dia ou semana em uma determinada sala ou bloco (LALESCU, L; DIRR, V., 2018).

Caso alguma restrição não seja atendida, o *software* alertará o conflito da restrição ocorrido. O resultado do horário escolar é gerado em HTML, podendo ser visualizado de várias maneiras. De forma geral, individualmente por turmas, professores entre outros.

3 ESTUDO DE CASO

A fim de atender os objetivos deste trabalho, inicialmente foi feito um estudo sobre um problema de lógica conhecido por “Teste de QI de Einstein” (Figura 6), que se assemelha a um calendário de horas escolares, no qual existem cinco homens com nacionalidades diferentes, morando em cinco casas com cores distintas e vizinhas, onde cada morador tem uma bebida, cigarro e animal de estimação preferidos, e nenhum homem tem a mesma preferência do outro (RACHA CUCA, 2019).

Figura 6: Teste de QI de Einstein

	1ª Casa	2ª Casa	3ª Casa	4ª Casa	5ª Casa
Cor	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Nacionalidade	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Bebida	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Cigarro	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Animal	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Fonte: RACHA CUCA (2019)

Este problema apresenta 15 restrições:

- *O Norueguês vive na primeira casa.*
- *O Inglês vive na casa Vermelha.*
- *O Sueco tem Cachorros como animais de estimação.*
- *O Dinamarquês bebe Chá.*
- *A casa Verde fica do lado esquerdo da casa Branca.*
- *O homem que vive na casa Verde bebe Café.*
- *O homem que fuma Pall Mall cria Pássaros.*
- *O homem que vive na casa Amarela fuma Dunhill.*
- *O homem que vive na casa do meio bebe Leite.*
- *O homem que fuma Blends vive ao lado do que tem Gatos.*
- *O homem que cria Cavalos vive ao lado do que fuma Dunhill.*
- *O homem que fuma BlueMaster bebe Cerveja.*
- *O Alemão fuma Prince.*
- *O Norueguês vive ao lado da casa Azul.*
- *O homem que fuma Blends é vizinho do que bebe Água.*

Como um horário escolar/universitário apresenta restrições para professores, este problema se assemelha com o mesmo, já que ambos tratam de preencher uma tabela com várias restrições a serem respeitadas. Sendo um problema tratável mesmo sem uso computacional, optou-se por adotá-lo a fim de se verificar a viabilidade de resolvê-lo usando alguns algoritmos desenvolvidos neste trabalho. Para início do estudo, foi implementado um algoritmo de busca em profundidade e um algoritmo genético para resolver tal problema em linguagem C.

No algoritmo de busca em profundidade, o estado inicial foi uma matriz 5x5 vazia, no qual a varredura acontecia de célula em célula preenchendo-as com todas as possibilidades e gerando sucessores, a partir de um função de avaliação, para preencher a próxima célula e verificando todas as 15 restrições. Busca em profundidade aplicado a problemas deste tipo tem a complexidade de $O(r * p^h)$, no qual r é o número de restrições, h é a quantidade de células e p é a quantidade de possibilidades para serem alocadas na célula. No problema do horário escolar, a quantidade de células é referente a quantidade de horários escolar do calendário semanal da instituição e as possibilidades são a quantidade de professores. O resultado do problema de lógica Teste de QI de Einstein foi solucionado em média de 0,16 segundos na implementação do algoritmo de busca em profundidade do autor (Figura 7).

Na implementação do algoritmo genético, o problema foi resolvido em média em 0,3 segundos, com média de 600 gerações. A população inicial teve 100 indivíduos aleatórios, em seguida foi selecionado o melhor indivíduo da geração e cruzado com sua própria mutação para gerar mais 100 indivíduos da próxima geração. A função *fitness*, uma função avaliativa, é dada por um valor inteiro pela quantidade de restrições satisfeitas. A mutação é a permutação de no máximo três características de casas aleatórias, e o cruzamento é feito entre dois indivíduos, cruzando as linhas de ambos.

Figura 7: Resultado das implementações

	1ª Casa	2ª Casa	3ª Casa	4ª Casa	5ª Casa
Cor	Amarela	Azul	Vermelha	Verde	Branca
Nacionalidade	Norueguês	Dinamarquês	Inglês	Alemão	Sueco
Bebida	Água	Chá	Leite	Café	Cerveja
Cigarro	Dunhill	Blends	Pall Mall	Prince	Bluemaster
Animal	Gatos	Cavalos	Pássaros	Peixes	Cachorros

Fonte: RACHA CUCA (2019)

3.1 ESTUDO DO FET

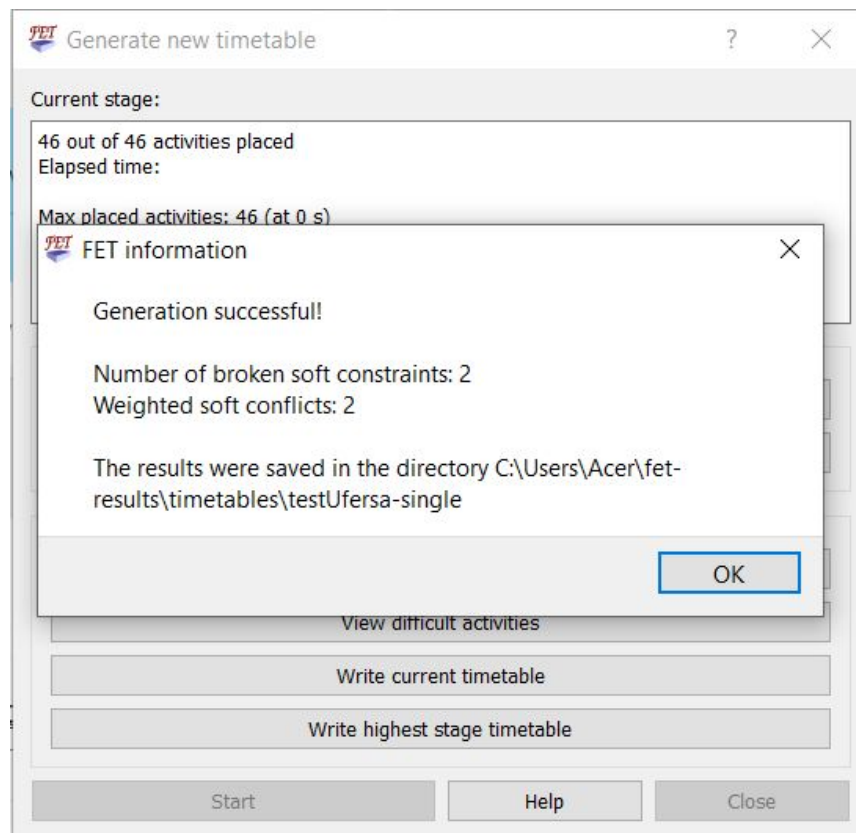
Em seguida, iniciou-se o estudo sobre o *software* FET, analisando todas as restrições de professores que o mesmo pode atribuir. Logo após, houve reuniões com coordenadores e ex-coordenadores de cursos da instituição do autor, para obter restrições de tempo de professores que são utilizadas no *campus* na elaboração de calendário escolar e não contempladas no FET. Foram encontradas duas, “Extensão máxima de dias por semana”, que é utilizado para compactar dias de trabalho de um servidor estudante, e a outra restrição foi “Mínimo de horas diárias de descanso contínuas”, esta restrição é para professores não trabalharem os três turnos, e ter pelo menos um turno de descanso.

Posteriormente, começou-se o estudo do código fonte do FET 5.41.0 para entender de como funciona a parte lógica do mesmo, o qual é escrito em C++, utilizando a biblioteca Qt, e para desenvolvimento utilizou-se o IDE Qt Creator 4.10.2.

Houve contato com o autor do FET, o Liviu Lalescu, para sanar algumas dúvidas. Assim, foi localizado o código para avaliações das restrições de tempo, o mesmo referente às restrições de tempos dos professores como as outras, que contém um método *fitness* para avaliar a restrição específica do estado do calendário gerado. Os primeiros testes foram feitos nos métodos *fitness* de restrições existentes semelhantes, no caso da nova proposta de restrição “Extensão máxima de dias por semana”, se assemelha à restrição existente “Extensão máxima de horas por dia”, pois esta restrição avalia o total de horas contínuas de trabalho durante o dia, já a nova proposta se trata do total de dias contínuos de trabalho durante a semana.

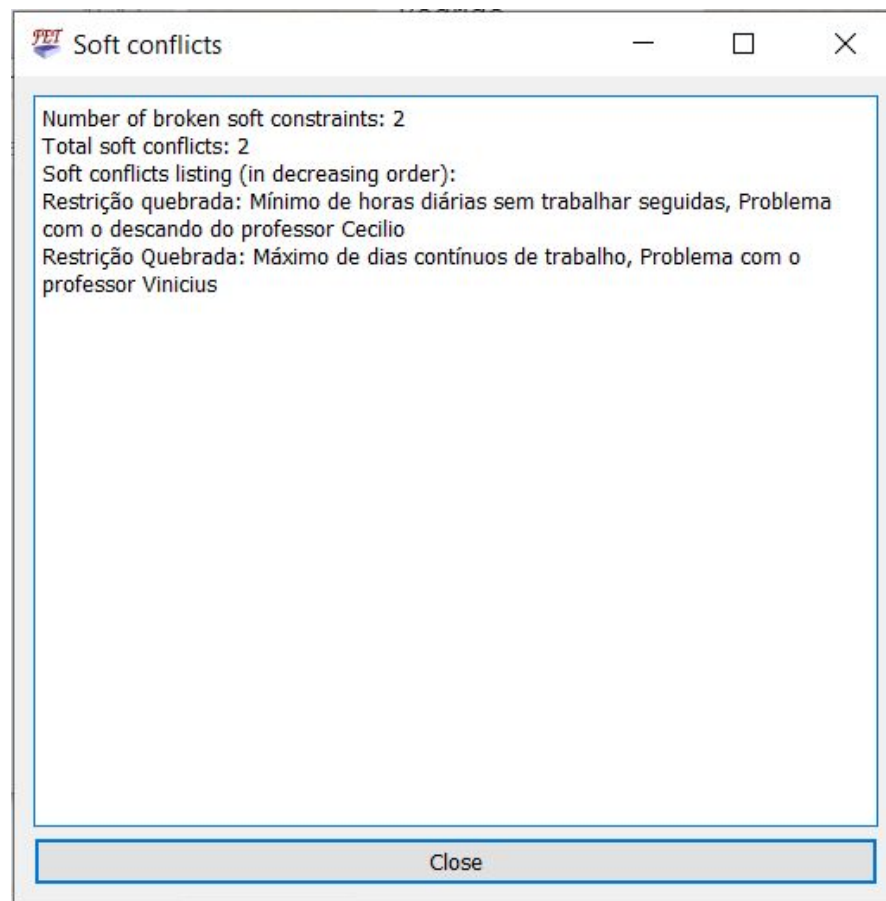
Foi feita uma alteração da restrição existente para satisfazer a nova proposta, porém esta alteração considera-se uma restrição *soft*, foi decidido implementar as restrições de forma *soft* por serem mais simples de se implementar. A implementação das restrições na forma *hard* exigirá um estudo mais aprofundado do código, mas o autor acredita que é uma tarefa viável para um trabalho futuro. Com isso, caso a restrição não for satisfeita, o FET irá avisar ao usuário (Figura 8), mas ainda permitirá a geração do horário mesmo com a restrição não-satisfeita.

Figura 8: Aviso do FET ao ocorrer quebra de restrição *soft*



Fonte: Autor (2020)

A proposta “Mínimo de horas diárias de descanso contínuas” também se assemelha com uma restrição existente, “Mínimo de horas de descanso”, esta irá avaliar o total de horas sem trabalho no final do dia e no início do próximo dia, ou seja, horas de descanso entre dois dias. A nova proposta tem finalidade de avaliar horas contínuas em um mesmo dia, fazendo com que professores não trabalhem os três turnos, tendo pelo menos um turno de descanso. Neste caso também foram feitas alterações para verificar as novas propostas. Após as alterações foram feitos alguns testes, inseriu-se dados para gerar um calendário de horário escolar, e atribui-se algumas restrições como também as restrições citadas. Na Figura 8 apresenta um alerta do FET ao ser quebradas duas restrições *soft*. É possível verificar quais restrições não foram satisfeitas como mostra a Figura 9:

Figura 9: Conflitos de restrições *soft*

Fonte: Autor (2020)

Esta figura e as seguintes, foram geradas a partir das alterações citadas feitas no FET. Com o objetivo de testarmos a implementação das restrições Extensão máxima de dias por semana e Mínimo de horas diárias de descanso contínuas. Foi aplicada a primeira restrição ao professor Vinicius e a segunda ao professor Cecílio. Com estas alterações das restrições, foi inserido mensagens para serem mostradas caso a restrição não fosse satisfeita. Logo em seguida, com o calendário gerado, é possível visualizar o calendário por professor, na Figura 10 apresenta-se o calendário de aulas do professor Vinicius, e como a restrição de Máximo de dias contínuos não foi satisfeito, o FET alertou. A restrição foi de três dias consecutivos, como é mostrado, o professor contém aulas em três dias, porém não são consecutivos, pois as mesmas estão distribuídas na segunda-feira, terça-feira e sexta-feira, tendo uma extensão de cinco dias no total.

Figura 10: Calendário de aulas do professor Vinícius com restrição insatisfeita

	Vinicius				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	---	---	---	---	---
08:50 - 09:45	---	---	---	---	---
09:45 - 10:40	8 periodo Programacao C e D	7 periodo Redes de Computadores	---	---	---
10:40 - 11:35			---	---	---
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	---	---	---	---	8 periodo Programacao C e D
14:50 - 15:45	---	---	---	---	
15:45 - 16:40	7 periodo Redes de Computadores	---	---	---	---
16:40 - 17:35		---	---	---	---

Fonte: Autor (2020)

Também a restrição de Mínimo de horas diárias de descanso contínuas não foi satisfeita para o professor Cecílio. Foi inserido a restrição para o professor ter pelo menos quatro horas de descanso por dia, e como é apresentado na Figura 11, em todos os dias que o professor trabalha, segunda-feira, quarta-feira e sexta-feira, não apresenta nenhuma sequência de quatro horas seguidas de descanso, o turno noturno foi desconsiderado.

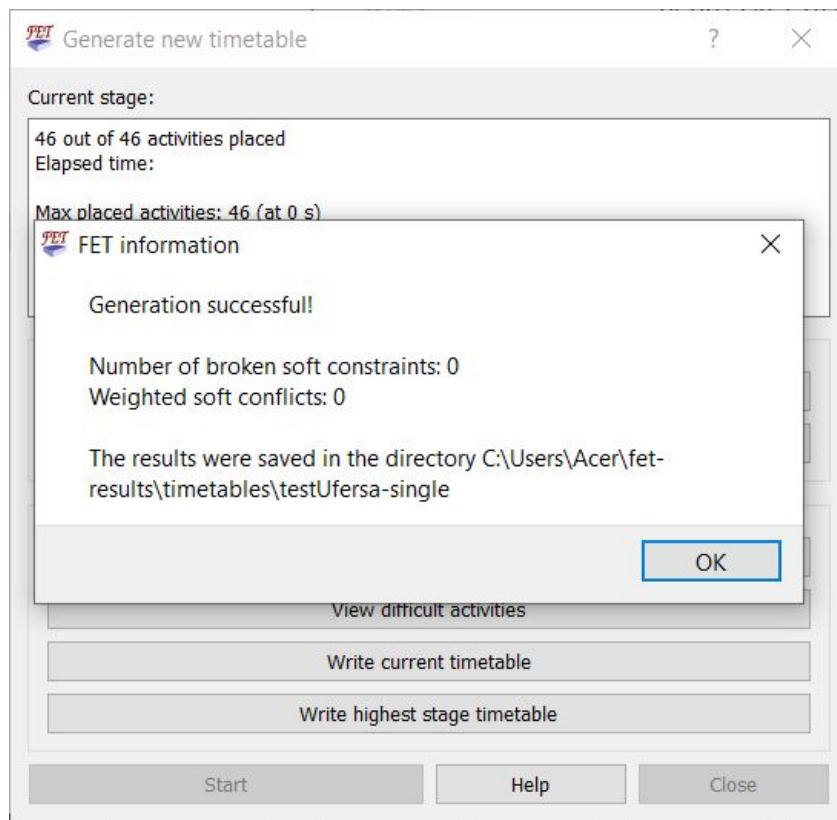
Figura 11: Calendário de aulas do professor Cecílio com restrição insatisfeita

	Cecilio				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	---	---	---	---	---
08:50 - 09:45	---	---	---	---	---
09:45 - 10:40	10 periodo Acionamentos para C e A Leaut LEAUT	---	10 periodo Acionamentos para C e A Leaut LEAUT	---	10 periodo Automacao Industrial Leaut LEAUT
10:40 - 11:35		---		---	
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	9 periodo Sistemas de Controle II	---	10 periodo Automacao Industrial Leaut LEAUT	---	9 periodo Introducao a Robotica
14:50 - 15:45		---		---	
15:45 - 16:40	9 periodo Introducao a Robotica	---	9 periodo Sistemas de Controle II	---	---
16:40 - 17:35		---		---	---

Fonte: Autor (2020)

Em seguida, foi gerado um calendário em que não houvesse nenhuma quebra de restrição, assim, diferente da Figura 8, a Figura 12 apresenta nenhuma restrição insatisfeita e conseqüentemente nenhum aviso de conflitos ao contrário da Figura 9.

Figura 12: Aviso do FET ao ocorrer nenhuma quebra de restrição soft



Fonte: Autor (2020)

Como nenhuma restrição foi insatisfeita, então foi gerado um calendário em que o professor Vinícius estava com aulas cadastradas em três dias consecutivos, consequentemente um extensão de três dias, terça-feira, quarta-feira e quinta-feira, como mostra a Figura 13.

Figura 13: Calendário de aulas do professor Vinícius com restrição satisfeita

	Vinicius				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	---	8 periodo Programacao C e D	8 periodo Programacao C e D	---	---
08:50 - 09:45	---			---	---
09:45 - 10:40	---	---	---	---	---
10:40 - 11:35	---	---	---	---	---
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	---	7 periodo Redes de Computadores	---	---	---
14:50 - 15:45	---		---	---	---
15:45 - 16:40	---	---	---	7 periodo Redes de Computadores	---
16:40 - 17:35	---	---	---		---

Fonte: Autor (2020)

O professor Cecílio apresentou quatro horas contínuas sem aula em todos os dias da semana, pois sua restrição também foi satisfeita, como é apresentado a Figura 14.

Figura 14: Calendário de aulas do professor Cecílio com restrição satisfeita

	Cecilio				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	9 periodo Introducao a Robotica	---	---	9 periodo Sistemas de Controle II	---
08:50 - 09:45		---	---		---
09:45 - 10:40	10 periodo Automacao Industrial Leaut LEAUT	---	---	10 periodo Acionamentos para C e A Leaut LEAUT	---
10:40 - 11:35		---	---		---
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	---	9 periodo Sistemas de Controle II	---	---	9 periodo Introducao a Robotica
14:50 - 15:45	---		---	---	
15:45 - 16:40	---	10 periodo Acionamentos para C e A Leaut LEAUT	---	---	10 periodo Automacao Industrial Leaut LEAUT
16:40 - 17:35	---		---	---	

Fonte: Autor (2020)

4 DESENVOLVIMENTO E TESTES

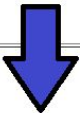
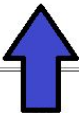
Como se sabe, a formulação e geração da grade de horários, enfrentado pelas instituições de ensino é de fato um problema antigo. Este problema se intensifica nas instituições de ensino superior, tendo em visto as diversas particularidades composta tanto pelo corpo docente como também pela oferta de espaço físico da instituição (VIEIRA, F; MACEDO, H, 2011; MENEZES JÚNIOR, 2017).

Em instituições de ensino superior, a grade horária tem que ser elaborada semestralmente, e geralmente após início das aulas precisa ser modificada. Diante do que foi visto, foram implementadas restrições particulares da Universidade Federal Rural do Semi Árido, *campus* Pau dos Ferros, no *software* FET.

4.1 RESTRIÇÃO EXTENSÃO MÁXIMA DE DIAS POR SEMANA

Após os estudos sobre o código fonte do FET 5.41.0, e comparações com restrições semelhantes ao citado no capítulo 3, o FET contém uma matriz binária para cada professor, nesta matriz se dá a dimensão da semana, as linhas serão as horas e as colunas os dias. É atribuído “0” para as células referentes às horas que o professor está sem compromisso, e atribuído “1” as células referentes às horas que o professor tem aula. Com esta informação, foi implementada a classe desta restrição respeitando todo o contexto do código já existente, e para avaliar a satisfação desta restrição, é necessário um método *fitness*, que a principal parte do código é apresentado no Anexo I. Na Figura 15 é mostrado como se dá a avaliação do método *fitness*.

Figura 15: Ilustração do método *fitness* da restrição Extensão máxima de dias por semana

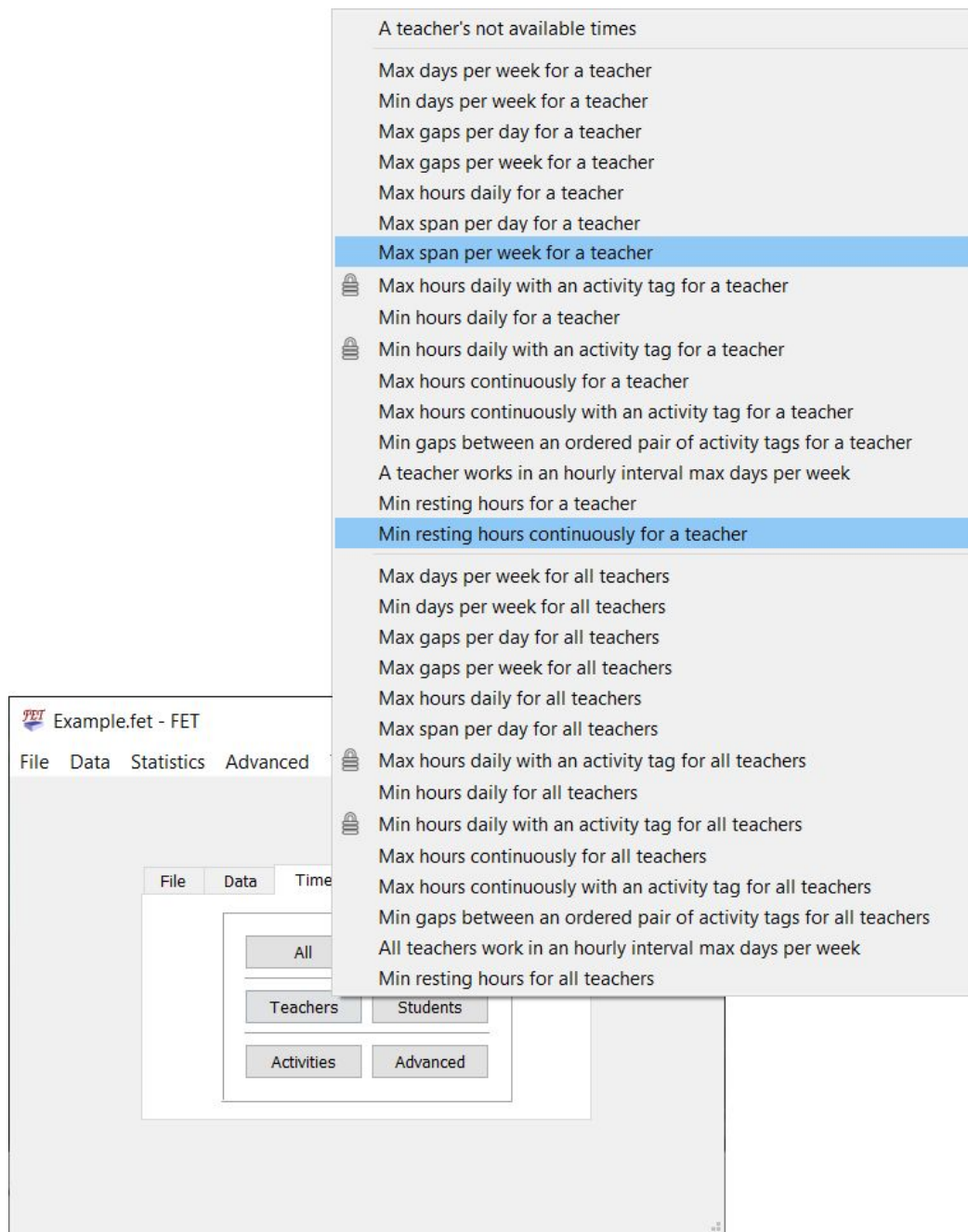
	Vinicius				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50		---	---	---	---
08:50 - 09:45		---	---	---	---
09:45 - 10:40	8 periodo Programacao C e D	7 periodo Redes de Computadores	---	---	---
10:40 - 11:35			---	---	---
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	---	---	---	---	8 periodo Programacao C e D
14:50 - 15:45	---	---	---	---	
15:45 - 16:40	7 periodo Redes de Computadores	---	---	---	
16:40 - 17:35		---	---	---	

Fonte: Autor (2020)

É feito um laço percorrendo os dias da semana da matriz do professor com esta restrição, iniciando pelo primeiro dia até o último, e a cada iteração contém outro laço que percorrerá todas as horas de um dia, iniciando pelo a primeira. Quando é encontrado uma célula que informa que o professor tem aula, o dia é atribuído a uma variável e os laços são encerrados. Em seguida é feito o processo inverso, um laço iniciando do último dia da semana até o primeiro, e outro laço da última hora do dia até a primeira, e ao encontrar um horário de trabalho do professor, o dia é memorizado em uma variável e os laços são encerrados. Com isso, é feita a diferença entre as duas variáveis alteradas, e assim encontrando a extensão de dias de trabalho. No exemplo da Figura 15 a extensão é de cinco dias, e caso seja maior do esperado, é alertado uma mensagem.

Com a função de avaliação da restrição *soft* implementada, foi para o próximo passo, a construção da interface. A Figura 16 apresenta a lista de todas as restrições de tempo para professor, e em destaque, mostra as duas restrições implementadas.

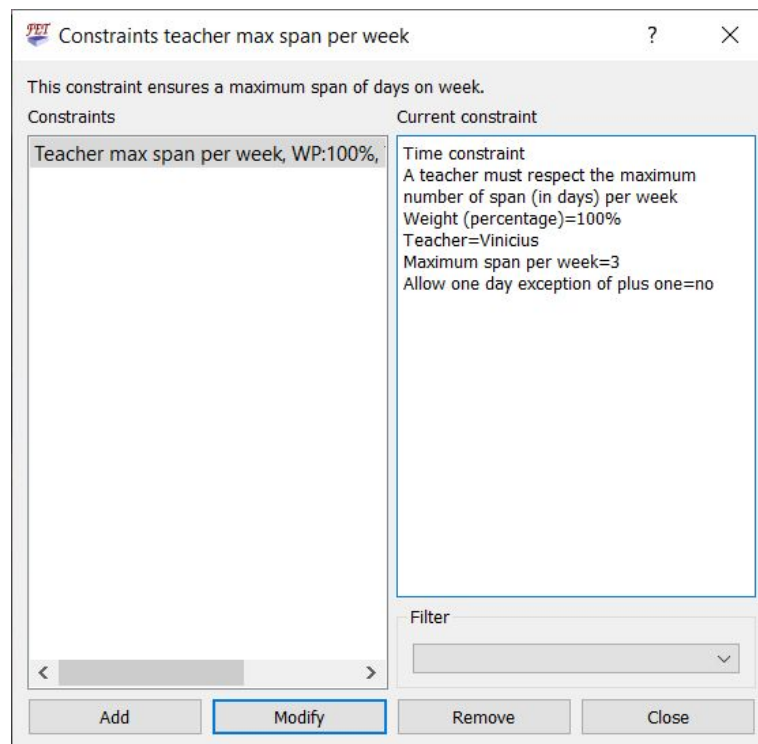
Figura 16: Lista das restrições de tempo para professor



Fonte: Autor (2020)

Ao selecionar a restrição em questão, abrirá uma tela para gerenciar a restrição (Figura 17), podendo adicionar uma restrição (Figura 18), modificar, remover e ver detalhes. Todas as restrições contém essas funcionalidades, porém estas destacadas na Figura 16 foram implementadas pelo o autor.

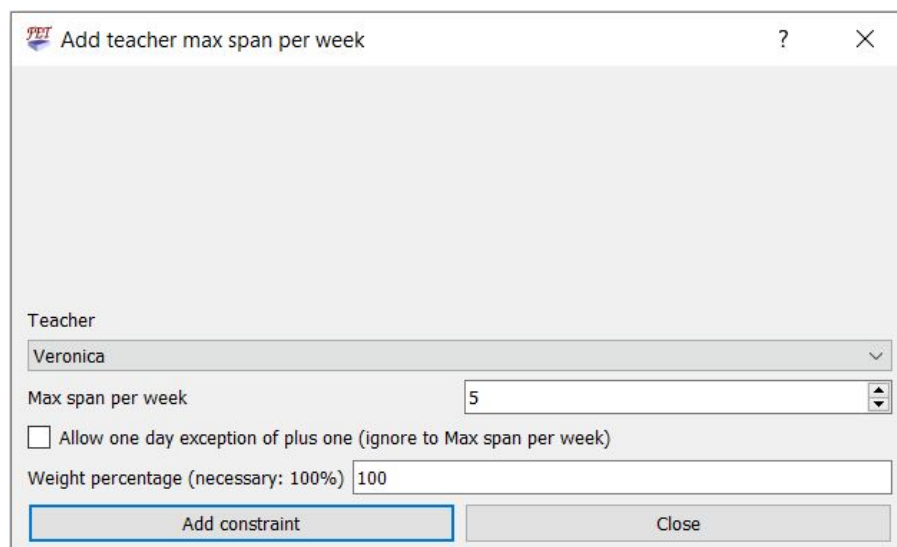
Figura 17: Detalhes da restrição Extensão máxima de dias por semana



Fonte: Autor (2020)

A tela de adição e modificação de uma determinada restrição são semelhantes. Então, para exibição dos resultados, será apresentada apenas a tela de adição da nova restrição. É possível selecionar o determinado professor para atribuir a restrição, e o máximo de dias de extensão.

Figura 18: Tela para adicionar a restrição Extensão máxima de dias por semana



Fonte: Autor (2020)

4.2 RESTRIÇÃO MÍNIMO DE HORAS DIÁRIAS DE DESCANSO CONTÍNUAS

Para esta restrição, também foi utilizada a matriz binária de cada professor e a matriz de pausas. Como esta restrição é relacionada ao turno e o FET trabalha apenas com horas sem destacar turnos, foi necessário criar horários que tenham pausas entre turnos. Como pode ser visto na Figura 19, no horário “11:35-13:55” é considerado como pausa para todos os dias. Isso será usado para identificar a separação entre turnos.

Figura 19: Ilustração do método *fitness* da restrição Mínimo de horas diárias de descanso contínuas

	Cecilio				
	Segunda	Terça	Quarta	Quinta	Sexta
07:55 - 08:50	---	---	---	---	---
08:50 - 09:45	---	---	---	---	---
09:45 - 10:40	10 periodo Acionamentos para C e A Leaut	---	10 periodo Acionamentos para C e A Leaut	---	10 periodo Automacao Industrial Leaut
10:40 - 11:35	LEAUT	---	LEAUT	---	LEAUT
11:35 - 13:55	-X-	-X-	-X-	-X-	-X-
13:55 - 14:50	9 periodo Sistemas de Controle II	---	10 periodo Automacao Industrial Leaut	---	9 periodo Introducao a Robotica
14:50 - 15:45		---	LEAUT	---	
15:45 - 16:40	9 periodo Introducao a Robotica	---	9 periodo Sistemas de Controle II	---	---
16:40 - 17:35		---		---	---

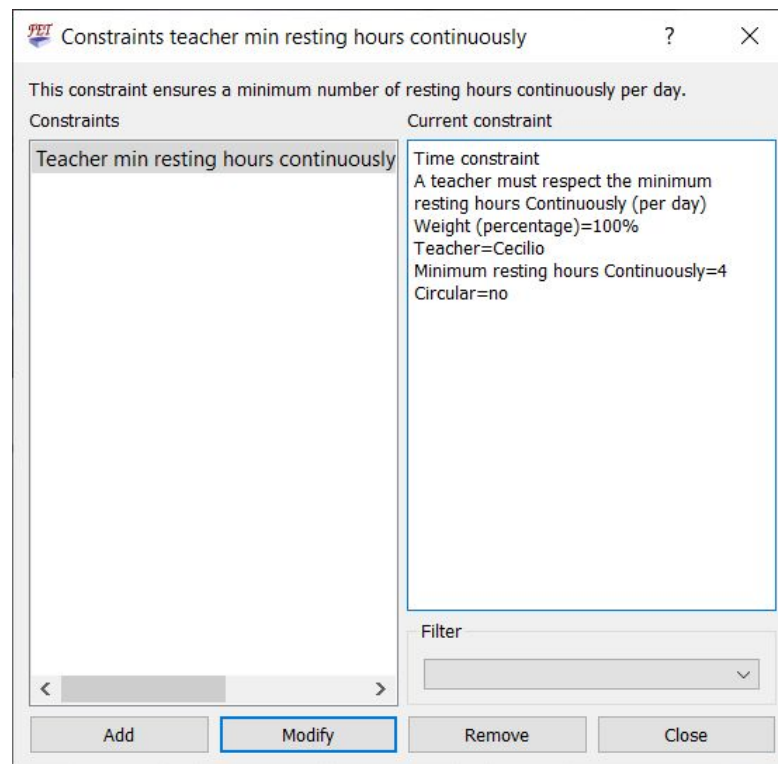
Fonte: Autor (2020)

Com estas matrizes, foi possível avaliar o estado do horário do professor com a restrição em questão. Como mostra a principal parte referente ao método *fitness* da restrição no Anexo II, existem dois laços de repetições aninhados verificando todas as horas do calendário do professor, e acumulando o total de horas de descanso consecutivas, e também acumulando o total de dias que contém pelo menos a quantidade de horas de descanso indicado. Em seguida é comparado o total de dias letivos com o total de dias que satisfazem a restrição, caso a quantidade de dias que satisfazem a restrição não seja igual ao total de dias letivos, o FET alertará o resultado. Como mostra a Figura 19, nos dias terça-feira e

quinta-feira, o professor não trabalhou, com isso houve o dia todo de descanso, já nos outros dias não houve nenhuma sequência de quatro horas de descanso.

Com a classe da restrição definida com os padrões do *software*, em seguida foi implementado o código responsável pela interface. Analogamente à restrição anterior.

Figura 20: Detalhes da restrição Mínimo de horas diárias de descanso contínuas



Fonte: Autor (2020)

As telas de gerenciamento e de adição da restrição Mínimo de horas diárias de descanso contínuas são semelhantes às anteriores, como apresentadas nas Figuras 20 e 21.

Figura 21: Tela para adicionar a restrição Mínimo de horas diárias de descanso contínuas

Fonte: Autor (2020)

Na Figura 22 apresenta-se a tela de conflitos de restrições *soft*, no qual estão presentes duas mensagens de alerta, uma de cada restrição implementada pelo autor, como já foi comentada no capítulo 3.

Figura 22: Tela de conflitos de restrições *soft*

Fonte: Autor (2020)

5 CONSIDERAÇÕES FINAIS

As investigações mostraram que o problema de preencher uma tabela respeitando restrições pode ser resolvido com busca em profundidade ou usando algoritmos genéticos. Porém as soluções implementadas podem ter uma complexidade demasiadamente alta para um número muito grande de variáveis. Foi investigado o uso do *software* FET e feitas entrevistas com coordenadores e ex-coordenadores dos cursos Engenharia de Computação, Tecnologia da Informação e Ciência e Tecnologia, para o levantamento de restrições usadas na Universidade Federal Rural do Semi Árido no *campus* Pau dos Ferros que não são atendidas pelo *software*.

O código fonte do *software* foi estudado e foram implementadas as restrições Extensão máxima de dias por semana e Mínimo de horas diárias de descanso contínuas na forma *soft* e a interface das mesmas. Também foram realizados testes bem sucedidos com o objetivo de validar as restrições implementadas. Observa-se que existe o potencial para melhoria de *software* FET de forma que ele possa ser utilizado para geração de horários na UFERSA no *campus* Pau dos Ferros. Além disso, aponta-se como trabalhos futuros a implementação das restrições de forma *hard*, que não podem ser violadas, e possíveis novas restrições.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BARATA, B, M, P; *et. al.* Problema de Alocação de Horários: um Estudo de Caso Utilizando o Software Livre FET. **Revista Eletrônica TECCEN**, Vassouras, v. 3, n.2, Edição Especial, p. 13-22, abr./jun., 2010.

BELLO, G. S. **Abordagem do problema de programação de grade horária sujeito a restrições utilizando coloração de grafos**. Dissertação de Mestrado. Universidade Federal do Espírito Santo - UFES, Vitória - ES, 2007.

COPPIN, Ben. **Inteligência Artificial**. Rio de Janeiro: LTC, 2017.

HAMAWAKI, Cristiane Divina Lemes. **Geração Automática de Grade Horária usando Algoritmos Genéticos: O caso da Faculdade de Engenharia Elétrica da UFU**. Dissertação de Mestrado. Universidade Federal de Uberlândia - UFU, Uberlândia, 2005.

HOLANDA, Iorrane Nobre de. **Um estudo sobre o Problema de Alocação de Salas aplicado à UFERSA - Pau dos Ferros**. Trabalho de Conclusão de Curso. Universidade Federal Rural do Semi Árido - UFERSA, Pau dos Ferros – RN, 2018.

LALESCU, L. **The description of the FET timetable generation algorithm**. 2019. Disponível em: <<https://lalescu.ro/liviu/fet/doc/en/generation-algorithm-description.html>>. Acesso em: 20 dez. 2019.

_____. **Some other details about the algorithm**. 2019. Disponível em: <<https://lalescu.ro/liviu/fet/doc/en/generation-algorithm-details.html>>. Acesso em: 20 dez. 2019.

LALESCU, L; DIRR, V. **FET Manual**. 2018. Disponível em: <<https://www.timetabling.de/manual/FET-manual.en.html>>. Acesso em: 24 dez. 2019.

_____. **FET Description**. 2020. Disponível em: <<https://lalescu.ro/liviu/fet/>>. Acesso em: 10 jan. 2020.

_____. **FET Download**. 2020. Disponível em: <<https://lalescu.ro/liviu/fet/download.html>>. Acesso em: 10 jan. 2020.

MENEZES JÚNIOR, Cláudio José. **Artorias: Solução para o problema de Alocação de horários universitários**. Trabalho de Conclusão de Curso. Instituto Federal Minas Gerais - IFMG, Formiga – MG, 2017.

POULSEN, Camilo José Bornia. **Desenvolvimento de um Modelo para School Timetabling Problem Baseado na Meta-Heurística Simulated Annealing**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul - UFRGS, Porto Alegre - RS, 2012.

RACHA CUCA. **Teste de QI de Einstein**. 2019. Disponível em: <<https://rachacuca.com.br/teste-de-einstein/>>. Acesso em: 25 nov. 2019.

ROCHA, Wallace de Souza. **Algoritmo GRASP para o Problema de Tabela-horário de Universidades**. Dissertação de Mestrado. Universidade Federal do Espírito Santo - UFES, Vitória - ES, 2013.

SCHAERF, A. **A Survey of Automated Timetabling**. n. 13. Artificial Intelligence Review, p. 87–127, 1999.

SUBRAMANIAN, A; *et. al.* Aplicação da metaheurística Busca Tabu na resolução do Problema de Alocação de Salas do Centro de Tecnologia da UFPB. In: Encontro Nacional de Engenharia de Produção, 2006, Fortaleza. **Anais do XXVI ENEGEP**. Fortaleza, 2006.

VIEIRA, F; MACEDO, H. Sistema de Alocação de Horários de Cursos Universitários: Um Estudo de Caso no Departamento de Computação da Universidade Federal de Sergipe. **Revista Scientia Plena**, São Cristóvão, v. 7, n.3, 2011.

ANEXO I

```

int nbroken=0;

int begin=-1;
for(int d=0; d<r.nDaysPerWeek && begin == -1; d++){
    for(int h=0; h < r.nHoursPerDay; h++){
        if(teachersMatrix[this->teacher_ID][d][h]>0){
            begin=d;
            break;
        }
    }
}

int end=begin;
for(int d=r.nDaysPerWeek-1; d>begin && end == begin; d--){
    for(int h=r.nHoursPerDay-1; h>=0; h--){
        if(teachersMatrix[this->teacher_ID][d][h]>0){
            end=d;
            break;
        }
    }
}

int span = end - begin + 1;
if(span > this->maxSpanPerWeek){
    QString s=tr("Broken constraint: Max span of days per week");
    dl.append(s + ", Problem with the teacher " + this->teacherName +
        ". Span of days: " + CustomFETString::number(span) +
        ". Span of days expected: " +
        CustomFETString::number(this->maxSpanPerWeek) + ".");
    cl.append(1*weightPercentage/100);
    nbroken++;
}

return nbroken;

```


ANEXO II

```

int nbroken=0;

int dias =0;
for(int d=0; d<r.nDaysPerWeek; d++){
    int cont = 0;
    for(int h=0; h <r.nHoursPerDay; h++){
        if(teachersMatrix[this->teacher_ID][d][h]>0 || breakDayHour[d][h]){
            cont=0;
        }else{
            cont++;
        }

        if(cont==this->minRestingHoursContinuously){
            dias++;
            break;
        }
    }
}

if(dias<r.nDaysPerWeek){
    QString s=tr("Broken constraint: Min resting hours per day continuously");
    dl.append(s + ", Problem with the teacher " + this->teacherName +
        ". Resting hours continuously in " + CustomFETString::number(dias) +
        " days. Resting hours continuously expected in " +
        CustomFETString::number(r.nDaysPerWeek) + " days.");
    cl.append(1*weightPercentage/100);
    nbroken++;
}

return nbroken;

```