Algoritmo ALNS Aplicado ao Problema de Tabela-Horário para Cursos de Computação da UFES

Danilo Erler Lima[1], **Arthur Roberto Barboza Maciel**[1], **Maria Claudia Silva Boeres**[1], **Edmar Hell Kampke**[2]

[1]Departamento de Informática – Universidade Federal do Espirito Santo (UFES)
Vitória – ES – Brazil

{danilo.e.lima,arthur.maciel}@edu.ufes.br, boeres@inf.ufes.br

[2]Departamento de Computação – Universidade Federal do Espirito Santo (UFES)
Alegre – ES – Brazil

edmar.kampke@ufes.br

Abstract. *This paper addresses the University Timetabling Problem in the context of computer science courses at UFES. In addition to the formulation as an optimization problem, an algorithm resulting from the adaptation of the Adaptive Large Neighborhood Search metaheuristic is proposed to solve it. The objective is to find a feasible solution, which is one that respects the constraints that cannot be violated, and minimizes violations of the constraints whose satisfaction is desirable. In order to validate the proposed algorithm, a set of instances with real data for the problem was used in the computational experiments. The results obtained demonstrate that the proposed algorithm finds solutions, on average, 88% better than the solutions currently generated.*

Resumo. *Neste trabalho é abordado o Problema de Tabela-Horário de Universidades no contexto dos cursos da área de computação da UFES. Além da formulação como um problema de otimização, um algoritmo resultante da adaptação da meta-heurística Adaptive Large Neighborhood Search, é proposto para resolvê-lo. O objetivo é encontrar uma solução viável, ou seja, que respeite as restrições que não podem ser violadas, e que minimize as violações das restrições cuja satisfação é desejável. Com o propósito de validar o algoritmo proposto foi utilizado nos experimentos computacionais, um conjunto instâncias com dados reais para o problema. Os resultados obtidos demonstram que o algoritmo proposto encontra soluções, em média, 88% melhores que as soluções geradas atualmente.*

1. Introdução

Os Problemas de Otimização Combinatória (POC) são comuns em diversas áreas e estão presentes em diversas atividades cotidianas. Esses problemas podem ser representados como a minimização ou maximização de uma função matemática, também denominada função objetivo. As variáveis da função objetivo são denominadas de variáveis de decisão e devem obedecer a certas restrições [Goldbarg and Luna 2005]. Os Problemas de Tabela-Horário (PTH) são um subconjunto importante dos POC, cujas aplicações são

variadas, como exemplo, na elaboração das escalas de trabalho de funcionários, no agendamento de partidas para campeonatos esportivos e na elaboração de horários escolares [Babaei et al. 2015].

No caso específico dos PTH da área educacional, o problema consiste em alocar um conjunto de aulas em um número pré-determinado de dias e horários, satisfazendo restrições definidas pela universidade (ou escola) [Kampke et al. 2015]. As restrições definidas pela instuição de ensino, juntamente com a função objetivo, formam o que chamamos de formulação do PTH daquela instituição. Entretanto, em alguns casos não é possível encontrar uma solução do PTH que atenda todas as restrições consideradas na formulação. Por isso, o conjunto de restrições é particionado normalmente em dois grupos: Restrições Fortes (RFt) e Restrições Fracas (RFc) [Santos and Souza 2007]. O conjunto RFt é composto pelas restrições que não podem ser violadas. Se uma tabela-horário não viola nenhuma restrição forte, ela é considerada uma solução viável. Já as restrições do conjunto RFc são aquelas cuja satisfação é desejável, mas caso não seja possível atendê-las, a solução não é inviabilizada. Além disso, segundo [Schaerf 1999] os PTH educacionais se subdividem em três categorias: Problemas de Tabela-Horário de Escolas (PTHE), Problemas de Tabela-Horário de Universidades (PTHU) e Problemas de Tabela-Horário de Exames (PTHEx).

O PTHU geralmente tem mais restrições que o PTHE e o PTHEx, sendo classificado NP-completo para a maioria das formulações [Schaerf 1999]. Isso significa que a solução ótima do PTHU só pode ser encontrada rapidamente para instâncias muito pequenas, o que não é a realidade da maioria das universidades brasileiras.

Na literatura diversos trabalhos apresentam métodos de solução para o PTHU de várias universidades brasileiras. Como exemplo, o trabalho de [Moreira et al. 2016], que apresenta a formulação do PTHU para o Departamento de Computação do *campus* de Alegre-ES da Universidade Federal do Espírito Santo (UFES). Nessa formulação são descritas sete restrições fortes e oito restrições fracas, sendo algumas delas bem específicas à realidade daquela instituição. Já no trabalho de [Costa et al. 2017], que foi realizado no Instituto de Matematica e Estatística da Universidade do Estado do Rio de Janeiro (UERJ), foi proposto um método de solução para o PTHU da instituição, que explora técnicas dos métodos *Simulated Annealing* (SA) e *Late Acceptance Hill Climbing*. O trabalho se caracteriza por implementar o conceito de separar as turmas por turnos (matutino, vespertino e noturno). No trabalho de [Sales 2015] uma formulação matemática completa para o PTHU da Universidade Federal de Santa Maria (UFSM) é apresentado. Nesta formulação é usado o conceito de proximidade de salas devido o tamanho do *campus*. Em outras palavras, a solução do problema tenta minimizar a distância percorrida pelos professores, por isso as salas de aula usadas pelo mesmo professor devem ser próximas.

Portanto, existem diferentes formulações para o PTHU, já que as restrições consideradas variam entre as universidades. Nesse caso, uma dificuldade adicional surge ao comparar a qualidade dos resultados obtidos pelos métodos de solução propostos para cada formulação. Diante desse cenário e com o objetivo de incentivar o estudo de diferentes abordagens sobre o PTHU, foram realizadas quatro edições (2002, 2007, 2011 e 2019) do campeonato internacional de tabela-horário (do inglês, *International Timetabling Competition* - ITC).

O ITC realizado em 2007 (ITC-2007) abordou o PTHU em três diferentes formulações [Di Gaspero et al. 2007]. Uma dessas formulações considera o currículo dos cursos, e por isso é denominada *Curriculum-Based Course Timetabling* (CB-CTT). A formulação CB-CTT é a que mais se aproxima da realidade das universidades brasileiras.

O trabalho de [Müller 2009] foi o vencedor do ITC-2007, na formulação CB-CTT, com um algoritmo *Iterative Forward Search*, que usa *Conflict-based Statistics* para gerar a solução inicial e *Hill Climbing* combinado com *Great Deluge* e SA para refinamento da solução. Após o ITC-2007, diversos autores também apresentaram métodos de solução para a formulação CB-CTT. Dentre esses, o algoritmo *Adaptive Large Neighborhood Search* (ALNS), desenvolvido por [Kiefer et al. 2017], foi o que até o momento apresentou as melhores soluções para o CB-CTT.

Diante do exposto, a motivação deste trabalho consistiu em definir uma formulação única e abrangente, baseada na formulação CB-CTT do ITC-2007 e de outros trabalhos da literatura, para os seis cursos de graduação na área de computação da UFES, ofertados nos três *campi* da universidade, com características bem distintas entre si. Essa formulação é apresentada na Seção 2. Além dela, na Seção 3 é apresentada a aplicação de um algoritmo construtivo (Seção 3.2), que gera uma solução inicial para o método ALNS (Seção 3.3). O ALNS obtém como resposta final uma solução para essa formulação do PTHU, utilizando instâncias com dados reais de cursos da área de computação da UFES. Os dados necessários, assim como a forma de construção dessas instâncias, são apresentados na Seção 3.1 e os resultados obtidos com a aplicação dos algoritmos às instâncias, na Seção 4. Por fim, as conclusões e trabalhos futuros são apresentados na Seção 5.

2. Modelo proposto para os cursos de computação da UFES

O PTHU baseado em currículos (CB-CTT), apresentado no ITC-2007 [Di Gaspero et al. 2007], consiste na alocação semanal de um conjunto de aulas de várias disciplinas, considerando um número de salas e de períodos de aulas. Os conflitos entre aulas das disciplinas são determinados de acordo com o currículo. Assim, um currículo é definido como um conjunto de disciplinas que possuem alunos em comum, e por isso duas disciplinas de um mesmo currículo não podem ter aulas alocadas no mesmo horário.

Uma solução (ou tabela-horário) da formulação CB-CTT representa a atribuição de horário e sala a todas as aulas de todas disciplinas da universidade. Essa formulação considera um conjunto de restrições, que são impostas às soluções.

Portanto, neste trabalho é proposta uma adaptação da formulação CB-CTT do ITC-2007 para os cursos de Computação da UFES, uma vez que esta formulação é a que mais se aproxima da realidade da maioria das universidades brasileiras. O objetivo é que a formulação seja abrangente o suficiente para representar o melhor possível, as características dos seis cursos de computação, a saber: Ciência da Computação e Engenharia de Computação oferecidos no turno diurno, tanto no *campus* situado em Vitória-ES quanto em São Mateus-ES e Ciência da Computação no turno diurno e Sistemas de Informação, no turno noturno, no *campus* de Alegre-ES.

A primeira etapa para a construção do modelo foi a aquisição das informações relativas aos cursos. Para isso, foram realizadas a análise das matrizes curriculares e de ofertas semestrais das disciplinas, além de entrevistas com os coordenadores. Em seguida,

tomando como base a formulação CB-CTT do ITC-2007 e também trabalhos correlatos na literatura, iniciou-se a construção da formulação do PTHU para o contexto dos cursos de computação da UFES, definindo-se a função objetivo e o conjunto de restrições.

O modelo construído é denominado COMP-UFES, e nele foram definidas oito restrições fortes e seis fracas. As quatro primeiras restrições fortes (**RFt1** a **RFt4**) consistem em restrições que também são pertencentes à formulação CB-CTT. Além dessas, outras três (**RFt5** a **RFt7**) são consideradas no modelo COMP-UFES. A oitava restrição forte é considerada na formulação CB-CTT do ITC-2007 como fraca. No entanto, o resultado das entrevistas realizadas com os coordenadores dos cursos indicou a sugestão unânime de utilização dessa restrição como forte. A seguir são apresentadas as restrições fortes do modelo COMP-UFES:

- **RFt1.** *Aulas*: Todas as aulas das disciplinas devem ser alocadas em horários diferentes.
- **RFt2.** *Conflitos*: Aulas de disciplinas do mesmo currículo ou lecionadas pelo mesmo professor devem ser alocadas em horários diferentes.
- **RFt3.** *Ocupação de Sala*: Duas aulas (de quaisquer disciplinas) não podem ocupar uma sala no mesmo horário.
- **RFt4.** *Disponibilidade*: Uma aula não pode ser alocada num horário em que a disciplina é indisponível.
- **RFt5.** *Horários Fixos*: Algumas disciplinas precisam ter suas aulas alocadas em horários pré-determinados.
- **RFt6.** *Tipo de Ambiente*: Disciplinas com aulas práticas precisam ter suas aulas alocadas em laboratórios, enquanto que disciplinas teóricas devem ter suas aulas alocadas em salas comuns.
- **RFt7.** *Obrigatórias*: As disciplinas obrigatórias de um currículo precisam estar alocadas no turno (matutino, vespertino ou noturno) do curso.
- **RFt8.** *Capacidade da Sala*: O número de alunos inscritos na disciplina deve ser menor ou igual ao número de assentos da sala em que a aula for alocada.

Em relação ao conjunto de restrições fracas do COMP-UFES, apenas as restrições **RFc1** e **RFc2** foram aproveitadas da formulação CB-CTT. As restrições **RFc3** e **RFc4** foram definidas a partir das entrevistas com os coordenadores de curso. Já as duas últimas restrições fracas (**RFc5** e **RFc6**) foram elaboradas tomando por base o trabalho de [Moreira et al. 2016] e as características observadas nos cursos da área de computação da UFES. As seis restrições fracas do modelo COMP-UFES são apresentadas a seguir:

- **RFc1.** *Aulas Isoladas*: Aulas do mesmo currículo devem ser alocadas em horários subsequentes.
- **RFc2.** *Estabilidade de Sala*: Todas as aulas de uma disciplina devem ser alocadas na mesma sala.
- **RFc3.** *Optativas*: As disciplinas optativas de um currículo devem ser alocadas preferencialmente no turno do curso.
- **RFc4.** *Estabilidade de Horário*: Todas as aulas de uma disciplina devem ser alocadas em dias diferentes, mas no mesmo horário.
- **RFc5.** *Janelas de Professores*: As aulas de disciplinas de um mesmo professor devem ser alocadas em horários adjacentes.
- **RFc6.** *Sobrecarga de Trabalho*: Deve-se evitar que professores em um dia ministrem aulas nos três turnos, ou que professores que ministrem aulas no turno noturno em um dia, ministrem aulas no turno matutino do dia seguinte.

Se uma solução do COMP-UFES não satisfaz alguma restrição forte, dizemos que é inviável, o que significa, na prática, que não pode ser usada. Caso contrário, a solução é viável. Como se trata de um problema de otimização, cujo interesse é obter a melhor solução que satisfaz às restrições do problema, toda solução viável S é avaliada por uma função objetivo f cujo valor $f(S)$ reflete sua qualidade. Como o COMP-UFES é um problema de minimização, f representa o número de violações (conflitos). Cada restrição fraca violada por uma solução, aumenta o seu valor de f de acordo com o peso da restrição. Assim, a melhor solução para o problema é aquela que é viável e que possui o menor valor possível de f. Desta forma, a função objetivo f para uma solução viável S da formulação COMP-UFES é dada por $f(S) = \sum_{i=1}^{6} \omega_i \cdot |RFc_i|$, sendo $|RFc_i|$ o número de violações da restrição fraca i de S e ω_i, o peso atribuído a ela.

Uma diferença na utilização da função f na formulação COMP-UFES, em relação à formulação CB-CTT, é que os pesos ω_i, $i = 1, \ldots, 6$, não são fixos, mas ajustáveis para a realidade dos cursos de cada *campus*. Como exemplo, no *campus* de Vitória-ES, há aulas apenas nos turnos matutino e vespertino, e por isso a restrição **RFc6** tem peso nulo ($\omega_6 = 0$). Já no *campus* de Alegre-ES, há aulas nos três turnos (matutino, vespertino e noturno) e por isso **RFc6** terá um peso que reflete a importância que essa restrição tem perante as demais restrições fracas. É importante destacar ainda outras diferenças entre as duas formulações. No modelo COMP-UFES foram adicionadas as definições de turnos e aulas-fixas, bem como a possibilidade de uma disciplina ter mais de um professor responsável. Da mesma forma, a formulação CB-CTT não apresenta diferenciação entre salas e laboratórios, e entre disciplinas obrigatórias e optativas. No entanto, o resultado das entrevistas com os coordenadores de curso, indica claramente a necessidade de tais diferenciações no modelo COMP-UFES.

3. Instâncias e algoritmos de solução para o COMP-UFES

Esta seção aborda as instâncias consideradas para o problema COMP-UFES, que foram construídas a partir de dados reais, assim como os algoritmos propostos para resolver o problema. Na Subseção 3.1, são apresentados os padrões de arquivos texto contendo as informações fornecidas como entrada para o problema COMP-UFES e os resultados gerados pelos algoritmos de solução, cujos pseudocódigos são descritos nas Subseções 3.2 e 3.3. Os algoritmos de solução implementados neste trabalho incluem um algoritmo construtivo, que gera uma solução inicial para o problema COMP-UFES e o algoritmo *Adaptive Large Neighborhood Search* (ALNS), que utiliza a solução inicial como base na busca de soluções melhores.

3.1. Construção das Instâncias para o COMP-UFES baseada em dados reais

Neste trabalho, foram construídas instâncias para o problema COMP-UFES, baseadas no padrão das instâncias existentes para a formulação CB-CTT do ITC-2007. Essas instâncias foram desenvolvidas a partir das informações coletadas das matrizes curriculares dos cursos, bem como das características específicas de cada curso, identificadas por meio de entrevistas realizadas com os coordenadores. Apesar de todos os cursos analisados serem de Computação e pertencerem à mesma instituição, eles estão localizados em *campi* diferentes, cada um com matrizes curriculares distintas e características diferentes na forma de ofertar disciplinas.

Para representar uma instância do problema COMP-UFES, foi construída a instância *Toy-UFES* a partir da instância *Toy* do ITC-2007 [Di Gaspero et al. 2007]. O

padrão do arquivo texto que representa esta instância e é utilizado como entrada para os algoritmos de solução, tem suas informações dispostas em uma sequencia de linhas, divididas por assunto, em nove partes: cabeçalho, disciplinas, salas, laboratórios, turnos, currículos, aulas-fixas, indisponibilidades e pesos. O cabeçalho é composto por dez linhas, cada uma contendo, respectivamente, o nome da instância, o número de disciplinas, salas, laboratórios, dias, períodos por dia, turnos, currículos, aulas-fixas e indisponibilidades. Na parte das disciplinas, cada linha contém o nome de uma disciplina, a quantidade de professores da disciplina, seguida pelos nomes dos professores que a lecionam, quantidade de aulas na semana, quantidade de alunos e o tipo de ambiente em que suas aulas terão que ser alocadas (S-Sala ou L-Laboratório). A terceira e quarta partes apresentam informações, respectivamente, das salas e laboratórios. Cada linha representa uma sala (laboratório) e possui o seu nome seguido de sua capacidade (número de assentos). Já na quinta parte são apresentadas as informações sobre os turnos, sendo que cada linha possui o nome do turno e o intervalo de abrangência entre os períodos de um dia. As informações sobre os currículos compõem a sexta parte do arquivo. Cada currículo é representado por uma linha contendo o seu nome, a quantidade de turnos seguido pelos nomes dos turnos em que as disciplinas do currículo podem ter suas aulas alocadas, o número de disciplinas obrigatórias presentes no currículo seguido dos nomes de cada disciplina, e da mesma forma as disciplinas optativas. A parte seguinte apresenta as disciplinas que possuem aulas fixas. Na penúltima parte são listadas as indisponibilidades das disciplinas, uma por linha, contendo o nome da disciplina, o dia e horário em que a disciplina não pode ser lecionada. Por fim, a última parte apresenta em uma única linha seis números que representam os pesos das restrições fracas (**RFc1** a **RFc6**).

O padrão de arquivo texto utilizado para representar a saída dos algoritmos é formado por uma sequencia de linhas, exibindo para cada disciplina, o dia, horário e sala alocados para cada uma de suas aulas, representando uma solução do problema COMP-UFES, exibida na forma de tabela-horário na Figura 1a. A tabela-horário é organizada pelos horários (linhas) e dias (colunas), sendo que para cada dia há uma subdivisão em salas e/ou laboratórios disponíveis para realização das aulas. Assim, cada aula é representada pelo nome da disciplina, alocada em um *timeslot* (posição) da tabela. A visualização da solução gerada é mais intuitiva a partir da Figura 1a. Por exemplo, a disciplina CG possui uma aula alocada no horário 2, do dia 1, na sala rB.

3.2. Algoritmo Construtivo

Para abordar o problema de alocação de aulas, inicia-se a solução por meio de um algoritmo construtivo. O algoritmo segue uma sequência de etapas projetadas para garantir que todas as aulas sejam alocadas, de modo que não infrinjam as restrições fortes. Inicialmente, as aulas que possuem horários fixos são alocadas. Em seguida, as aulas restantes são alocadas de acordo com um critério de priorização, baseado na quantidade de *timeslots* disponíveis, garantindo que disciplinas com menos opções de alocação sejam alocadas primeiro. O Algoritmo 1 apresenta o pseudocódigo do algoritmo construtivo.

O algoritmo recebe como entrada o conjunto D de disciplinas a serem alocadas, o conjunto R de salas e laboratórios disponíveis para a realização das aulas e o conjunto A com as informações dos *timeslots* – sem conflitos – das aulas fixas das disciplinas. A solução S a ser retornada como resposta pelo Algoritmo 1 é inicialmente vazia (linha 1). No laço das linhas 2 a 4 todas as aulas fixas $a \in A$ são alocadas em S. Após isso, é gerada uma lista de aulas ainda não alocadas em S, que é representada pelo conjunto C na linha

Algoritmo 1: Algoritmo Construtivo

Entrada: D = {conjunto de disciplinas},
R = {conjunto de salas e laboratórios},
A = {conjunto de aulas fixas das disciplinas}
Saída: Solução S

```
1  S ← ∅;
2  para a ∈ A faça
3  |  S ← AlocaAulaFixa(S, R, a);
4  fim para
5  C ← GeraListaAulasNaoAlocadas(S, D);
6  enquanto |C| > 0 faça
7  |  para c ∈ C faça
8  |  |  tsd_c ← CalculaQtdTimeSlotsDisponiveis(S,R,c);
9  |  fim para
10 |  OrdenaListaAulasNaoAlocadas(C);
11 |  c' ← ObtemPrimeiroElemento(C);
12 |  ts ← EscolheTimeslotAleatorio(S,R,c');
13 |  S ← AlocaAula(S, R, c', ts);
14 |  C ← C − {c'} ;
15 fim enquanto
```

5. Em seguida, iterativamente, cada aula $c' \in C$ é alocada em S, como pode ser observado no laço das linhas 6 a 15. Em cada iteração desse laço, é calculado o valor tsd_c para cada aula $c \in C$ (linhas 7 a 9). O valor de tsd_c representa o número de *timeslots* disponíveis para alocação da aula c na solução parcial S. É importante destacar que na função *CalculaQtdTimeSlotsDisponiveis*(S,R,c) (linha 8) serão contabilizados apenas os *timeslots* cuja possível alocação de c não viole as restrições fortes do problema. A partir disso, a lista C é ordenada crescentemente pelos valores de tsd_c (linha 10), com o intuito de alocar prioritariamente as aulas que possuem menos *timeslots* disponíveis. Dessa maneira, após a ordenação, na linha 11 é obtida a aula $c' \in C$ que possui menor valor de tsd_c. Nesse momento, é selecionado aleatoriamente um *timeslot* ts dentre os $tsd_{c'}$ disponíveis para a aula c' (linha 12). Enfim a aula c' é alocada no *timeslot* ts dentro da solução S (linha 13) e removida da lista C (linha 14). O procedimento iterativo termina quando o conjunto C estiver vazio, o que indica que todas as aulas de todas as disciplinas do conjunto D foram alocadas em S, ou seja, que S é uma solução (tabela-horário) completa e viável para o COMP-UFES, e por isso está apta a ser enviada ao algoritmo ALNS, apresentado na próxima seção, para uma busca de possíveis melhorias.

3.3. Algoritmo ALNS para o COMP-UFES

Neste trabalho foi implementada a meta-heurística *Adaptive Large Neighborhood Search* (ALNS) proposta por [Ropke and Pisinger 2006], para a resolução do COMP-UFES.

A meta-heurística ALNS utiliza uma função adaptativa probabilística, que seleciona, baseado em pesos, um método de destruição e um método de reconstrução de soluções. O peso de cada método (heurística), depende da pontuação obtida pelo método nas iterações passadas. Assim, a cada iteração, a solução atual é parcialmente destruída e reconstruída, gerando uma nova solução, possibilitando a exploração da vizinhança de soluções. Essa nova solução é aceita de acordo com critérios similares ao do algoritmo *Simulated Annealing*, que permite soluções de qualidade inferior. Esse processo é repe-

Dias	Dia 1			Dia 2		
Salas	rA	rB	IC	rA	rB	IC
Horário 1		CD	PG	AB	CD	PG
Horário 2	AB	CG	PA		CD	PA
Horário 3	GI		PA	AB		PA
Horário 4		GI	PG	GI		
Horário 5		CD			GI	PG

(a) Solução Inicial do COMP-UFES

Dias	Dia 1			Dia 2		
Salas	rA	rB	IC	rA	rB	IC
Horário 1		CD	PG	AB	CD	PG
Horário 2	AB	CG	PA		CD	PA
Horário 3	GI		PA	AB		PA
Horário 4		GI	PG	GI		
Horário 5		CD			GI	PG

(b) Destruição: desmonte da solução

Dias	Dia 1			Dia 2		
Salas	rA	rB	IC	rA	rB	IC
Horário 1		CD	PG	AB	CD	PG
Horário 2	AB	CG	PA		CD	PA
Horário 3	GI		PA			PA
Horário 4			PG	GI		
Horário 5		CD			GI	

(c) Destruição: solução incompleta

Dias	Dia 1			Dia 2		
Salas	rA	rB	IC	rA	rB	IC
Horário 1		CD	PG	AB	CD	PG
Horário 2	AB	CG	PA	AB	CD	PA
Horário 3	GI		PA			PA
Horário 4	GI		PG	GI		
Horário 5		CD	PG		GI	

(d) Reconstrução: solução

Figura 1. Estratégias de Destruição e Reconstrução

tido até que uma condição de parada seja satisfeita, que neste trabalho foi definido, como tempo máximo de execução.

No algoritmo ALNS implementado neste trabalho foram desenvolvidas duas heurísticas de destruição (D_1 e D_2) e duas de reconstrução (R_1 e R_2). As heurísticas D_1 e D_2 destroem parcialmente uma solução, ao retirar (desalocar) 15% das aulas alocadas. Já as heurísticas R_1 e R_2 reinserem (realocam) as aulas removidas. A diferença entre D_1 e D_2 está no fato que D_1 é totalmente aleatória, ou seja, as aulas a serem retiradas da solução são escolhidas aleatoriamente, enquanto que na heurística D_2 é usada uma estratégia gulosa, em que as aulas que mais impactam o valor de f são escolhidas e desalocadas. Uma diferença similar ocorre entre R_1 e R_2, já que R_1 escolhe aleatoriamente os *timeslots* em que as aulas removidas serão realocadas, e R_2 usa uma estratégia parcialmente gulosa e aleatória, pois para cada aula removida são escolhidos aleatoriamente cinco *timeslots* vazios, e dentre esses, o que proporciona o menor incremento em f é o escolhido para a aula ser realocada.

A Figura 1 mostra um exemplo de aplicação dessas estratégias na solução gerada pelo algoritmo construtivo (Figura 1a). A Figura 1b, representa a fase de destruição, na qual 15% das aulas alocadas são escolhidas para serem removidas da solução (*timeslots* em destaque laranja). A Figura 1c mostra a solução incompleta após a aplicação da operação de destruição, com os *timeslots* outrora ocupados, mas agora vazios (destaque vermelho). Por fim, a Figura 1d representa a solução após a reconstrução, uma vez que as aulas removidas foram reinseridas na solução (*timeslots* em destaque verde).

4. Experimentos Computacionais e Resultados

Os experimentos computacionais foram realizados utilizando um conjunto de cinco instâncias, sendo uma delas a instância *Toy-UFES* e as outras quatro com dados reais dos semestres letivos 2023/2 e 2024/1 dos cursos de Ciência da Computação e Sistemas de informação do *campus* de Alegre-ES e Ciência da Computação e Engenharia de Computação do *campus* de Vitória-ES. Além disso, foi definido o tempo máximo de execução do algoritmo ALNS em 180 segundos para cada execução em cada instância.

O algoritmo ALNS foi executado cinco vezes em cada uma das cinco instâncias do COMP-UFES, tendo sido desenvolvido na linguagem de programação C e compilado com o compilador *GNU gcc compiler*. Os experimentos computacionais foram realiza-

dos em um ambiente com microprocessador Intel Core i3-9100H de 3.6 GHz, 16 GB de memória RAM, utilizando o sistema operacional Ubuntu 20.04.4 LTS.

Os resultados obtidos nos experimentos computacionais são resumidos na Tabela 1. A primeira e segunda colunas da Tabela 1 apresentam os nomes das instâncias e o valor de f para as soluções construídas manualmente pelos coordenadores de curso. As quatro colunas seguintes trazem a melhor solução (f_{best}) e a solução média (f_{avg}) encontradas respectivamente pelos algoritmos construtivo e ALNS. Por fim, as duas últimas colunas indicam as distâncias dos resultados obtidos pelo ALNS em relação àqueles obtidos pelo algoritmo construtivo e ALNS, calculadas respectivamente pelas fórmulas: $gap_C = |f_{bestA} - f_{bestC}|/f_{bestC}$ e $gap_M = |f_{bestA} - f_{Manual}|/f_{Manual}$, sendo f_{bestA} e f_{bestC} os valores de f_{best}, respectivamente, do ALNS e do algoritmo construtivo.

Tabela 1. Resultados obtidos pelos algoritmos construtivo e ALNS

Instâncias	f_{Manual}	Construtivo		ALNS			
		f_{best}	f_{avg}	f_{best}	f_{avg}	$gap_C(\%)$	$gap_M(\%)$
Toy-UFES	—	9	17,6	0	0	—	—
Alegre-ES (2023/2)	1175	1355	1479	265	318	80,44	77,45
Alegre-ES (2024/1)	1515	1385	1826	160	173	88,45	89,44
Vitória-ES (2023/2)	2310	1290	1414	110	115	91,47	95,24
Vitória-ES (2024/1)	1680	1185	1348	170	189	85,65	89,88

Os resultados apresentados ilustram a superioridade do ALNS, que melhora em mais de 77% as soluções alcançadas das outras formas, isto é, pelo algoritmo construtivo e manualmente. No caso das instâncias do *campus* de Alegre-ES, notou-se que na instância $2023/2$ a solução manual é melhor do que a obtida pelo algoritmo construtivo. Isso ocorreu pois no algoritmo construtivo a solução é obtida de forma parcialmente aleatória, enquanto que na manual os coordenadores de curso, ao alocarem as aulas na tabela-horário, já evitam a violação de certas restrições. Já no caso das instâncias de Vitória-ES, observou-se que tanto a solução manual quanto as soluções construídas pelo algoritmo construtivo, foram impactadas principalmente pelas restrições **RFc4** – Estabilidade de Horário e **RFc5** – Janelas de Professores. Entretanto, o ALNS foi eficiente em todas as instâncias em reduzir o número de violações dessas restrições.

5. Conclusão e Trabalhos Futuros

Neste trabalho foi abordado o PTH para os cursos de computação da UFES. Considerando a realidade distinta dos três *campi* da UFES, que oferecem cursos de computação, inicialmente foi definida uma formulação única para o problema. A formulação COMP-UFES foi elaborada com base na formulação CB-CTT do ITC-2007 e de outros trabalhos da literatura que abordam o PTH em universidades brasileiras. A definição das restrições fortes e fracas da formulação foram realizadas a partir de entrevistas com os coordenadores de curso, que são os atuais responsáveis pela confecção semestral das tabelas-horário dos seus respectivos cursos.

Além da formulação, também foi desenvolvido um método de solução para o COMP-UFES, dividido em duas etapas. Na primeira etapa, um algoritmo construtivo gera uma solução inicial a ser submetida em uma segunda etapa, ao algoritmo ALNS que tenta melhorar a solução construída. Experimentos computacionais foram conduzidos em

cinco instâncias, sendo quatro delas com dados reais, e mostraram que foi possível encontrar soluções, em média, 88% melhores que as soluções construídas manualmente pelos coordenadores de curso. Como trabalhos futuros pretende-se o aprimoramento do algoritmo construtivo e desenvolvimento de novas heurísticas de destruição e reconstrução para o ALNS, bem como obter dados reais do *campus* de São Mateus-ES e dados dos próximos semestres letivos.

Referências

Babaei, H., Karimpour, J., and Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43 – 59.

Costa, D. H. S., Coelho, I. M., and Pinto, P. E. D. (2017). Programaçao de horários e alocaçao de salas de aula no ime/uerj com simulated annealing e lahc. In *Anais do XLIX SBPO*, Rio de Janeiro-RJ. SOBRAPO.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). In *Proceedings of The International Conference on Automated Planning and Scheduling*, pages 1–5, Providence, Rhode Island, USA. ICAPS Incorporation.

Goldbarg, M. C. and Luna, H. P. L. (2005). *Otimização Combinatória e programação linear: modelos e algoritmos*. Elselvier, Rio de Janeiro, RJ, Brasil, 2 edition.

Kampke, E. H., Rocha, W. S., Boeres, M. C. S., and Rangel, M. C. (2015). A GRASP algorithm with path relinking for the university courses timetabling problem. In *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, volume 3, pages 1081–1087, São Carlos, SP, Brasil. SBMAC.

Kiefer, A., Hartl, R. F., and Schnell, A. (2017). Adaptive large neighborhood search for the curriculum-based course timetabling problem. *Annals of Operations Research*, 252(2):255–282.

Moreira, L. V., Monteiro, R. C., Kampke, E. H., and Mauri, G. R. (2016). Metaheurística GRASP para o Problema de Tabela-horário de Disciplinas do Departamento de Computação do CCA-UFES. In *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, pages 2171–2182, Rio de Janeiro, RJ, Brasil. SOBRAPO.

Müller, T. (2009). ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1):429–446.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Sales, E. S. (2015). Problema de alocação de salas e a otimização dos espaços no centro de tecnologia da ufsm. Mestrado em administração, Universidade Federal de Santa Maria.

Santos, H. G. and Souza, M. J. F. (2007). Programação de horários em instituições educacionais: formulações e algoritmos. *XXXIX SBPO-Simpósio Brasileiro de Pesquisa Operacional*, (1):2827–2882.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.