Walace de Souza Rocha

Algoritmo GRASP para o Problema de Tabela-horário de Universidades

Vitória - ES, Brasil

28 de Fevereiro de 2013

Walace de Souza Rocha

Algoritmo GRASP para o Problema de Tabela-horário de Universidades

Dissertação apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Espírito Santo.

Orientador:
Maria Claudia Silva Boeres

Co-orientador:
Maria Cristina Rangel

DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória - ES, Brasil

28 de Fevereiro de 2013

Dissertação de Projeto Final de Graduação sob o título *"Algoritmo GRASP para o Problema de Tabela-horário de Universidades"*, defendida por Walace de Souza Rocha e aprovada em 28 de Fevereiro de 2013, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos professores:

Profa. Dr. Maria Claudia Silva Boeres
Orientadora

Profa. Dr. Maria Cristina Rangel
Co-orientadora

Prof. Dr. Fulano de Tal
Universidade Federal do Espírito Santo

Resumo

Escreva aqui o texto do seu resumo.

Abstract

Write here the English version of your “Resumo”.

Dedicatória

Dedico este trabalho a ...

Agradecimentos

Agradeço a ...

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos p. 11

1 Introdução p. 12

1.1 Apresentação do problema . . . . . p. 13

1.2 Motivação para o problema . . . . . p. 13

1.3 Objetivos deste trabalho . . . . . p. 13

1.4 Organização do texto . . . . . p. 14

2 Estado da Arte p. 15

2.1 Introdução . . . . . p. 16

2.2 Conceitos e definições . . . . . p. 16

2.3 Abordagens existentes . . . . . p. 17

2.4 Trabalhos relacionados . . . . . p. 19

3 Proposta do trabalho p. 20

3.1 Formulação do ITC-2007 . . . . . p. 21

3.2 Algoritmo GRASP . . . . . p. 21

3.2.1 Estrutura básica . . . . . p. 21

3.2.2 Hibridização com *Path-Relinking* . . . . . p. 24

4 Resultados Computacionais p. 27

4.1 Descrição das instâncias utilizadas . . . . . p. 28
4.2 Detalhes de implementação . . . . . p. 28
4.3 Sintonia da Meta-heurística . . . . . p. 28
4.4 Análise dos resultados . . . . . p. 29

5 Conclusões e trabalhos futuros p. 30

5.1 Conclusões . . . . . p. 31

Referências Bibliográficas p. 32

Anexo A – Ferramentas utilizadas p. 36

Lista de Figuras

Lista de Tabelas

4.1 Tabela com informações sobre cada instância do ITC-2007 . . . . . p. 29

Lista de Algoritmos

1 Estrutura básica do algoritmo GRASP . . . . . p. 22

2 Algoritmo guloso aletório para construção da solução inicial do GRASP . . . . p. 23

3 Algoritmo *Path-Relinking* . . . . . p. 25

4 Algoritmo GRASP . . . . . p. 26

1 Introdução

Neste capítulo são apresentados o objetivo desta dissertação e a estrutura da mesma.

1.1 Apresentação do problema

O problema de tabela-horário consiste em alocar um conjunto de aulas em um número pré-determinado de horários, satisfazendo diversas restrições envolvendo professores, alunos e o espaço físico disponível. A solução manual deste problema não é uma tarefa trivial e as instituições de ensino precisam resolvê-lo anualmente ou semestralmente. Nem sempre a alocação manual é satisfatória, por exemplo, quando um aluno não consegue matricular em duas disciplinas porque elas são alocadas no mesmo horário.

Por esta razão, atenção especial tem sido dada a solução automática de tabela-horário. Nos últimos cinquenta anos, começando com [Gotlieb 1962], este problema ganhou grande destaque na área de otimização combinatória, tendo diversos trabalhos publicados.

1.2 Motivação para o problema

O problema de tabela-horário está entre os mais difíceis da área de otimização combinatória. Em [Schaerf 1995] pode ser visto que ele é classificado como NP-completo. Assim, a solução exata só pode ser garantida para instâncias bem pequenas, que não correspondem às instâncias reais da maioria das instituições de ensino.

Existe uma necessidade de propor algoritmos cada vez mais eficientes que produzam tabelas-horário satisfatórias em um tempo viável, independente do tamanho da instância. Devido à complexidade do problema, métodos exaustivos são descartados. Diferentes meta-heurísticas tem sido aplicadas devido ao fato de serem relativamente simples de implementar e produzirem bons resultados. Dentre as meta-heurísticas que já foram aplicadas ao problema, existe um esforço em aplicar melhorias para conseguir melhores resultados. Há algumas meta-heurísticas que ainda não foram exploradas no problema, o que deixa uma incógnita quanto à sua eficiência.

1.3 Objetivos deste trabalho

O objetivo principal deste trabalho é resolver o problema de tabela-horário de universidades usando a meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedures*). Pelo fato de existirem diversas formulações para o problema, será escolhida uma que tem sido bastante utilizada na área: a que é proposta no ITC-2007 (*International Timetabling Competition - 2007*) [PATAT 2008]. A razão principal desta escolha é facilitar a comparação dos resultados com outros algoritmos propostos na literatura.

Para atingir este objetivo será necessário estudar a formulação do ITC-2007 e propor uma implementação eficiente dos dados. Além de aplicar o GRASP para o problema, pretende-se implementar algumas melhorias que já foram propostas na literatura e que visam melhorar a versão básica do algoritmo.

Vale ressaltar que o ITC-2007 dividiu o campeonato em três *tracks*. Cada *track* possui uma formulação específica e seu próprio conjunto de instâncias. O *track* 1 é uma formulação específica para aplicação de exames finais. Os *tracks* 2 e 3 tratam a alocação semanal das aulas de uma universidade. A diferença básica é que o *track* 2, chamado de *post enrolment* (pós matrícula) trata o problema considerando os dados de matrícula dos alunos, enquanto que no 3 as informações levam em conta os currículos que compõe a universidade.

Será adotado neste trabalho a formulação do *track* 3 por ser considerada, dentre as três, a mais próxima do que acontece na prática nas universidades brasileiras. Esta formulação será explicada em detalhes no capítulo 3.

Deseja-se por fim coletar na literatura resultados obtidos para o problema com diferentes técnicas de solução, a fim de comparar com o algoritmo proposto.

1.4 Organização do texto

No capítulo 2 é apresentado o estado da arte listando as principais técnicas de solução do problema. No capítulo 3 são apresentados a formulação do problema segundo o ITC-2007, o algoritmo GRASP e sua aplicação no problema. No capítulo 4 são apresentados detalhes de implementação, instâncias de testes e os resultados obtidos. No capítulo 5 são listadas as conclusões obtidas no trabalho e enumerados alguns trabalhos futuros.

2 Estado da Arte

"*Navegar é preciso, viver não.*"

Luís de Camões

Neste capítulo são apresentados ....

2.1 Introdução

O problema de tabela-horário não possui uma formulação única. Como pode ser visto em [Schaerf 1995], ao longo do tempo surgiram diversas modelagens. Essa variedade surgiu pelo fato das restrições do problema serem específicas a determinada instituição de ensino. Alguns *benchmarks* foram criados, como é o caso do ITC-2007 [PATAT 2008]. A maior contribuição do *benchmark* é fornecer um conjunto de dados para que os diferentes pesquisadores possam comparar suas técnicas de solução.

Apesar das diferentes formulações, os problemas de tabela-horário possuem uma característica em comum: a separação das restrições em dois grupos. São fortes ou fracas.

As restrições fortes são aquelas que não podem ser violadas. Elas restringem o conjunto de soluções para impedir certas situações irreais, como por exemplo, alunos assistindo mais de uma aula no mesmo tempo ou uma sala sendo alocada para mais de uma aula no mesmo horário. Se uma tabela-horário não viola nenhuma restrição forte ela é dita ser uma solução viável.

As restrições fracas são aquelas que não interferem na viabilidade da solução, mas refletem certas preferências das instituições. É desejável, por exemplo, que um aluno não tenha aulas isoladas durante o dia ou que aulas da mesma disciplina sejam lecionadas na mesma sala. As restrições fracas podem possuir pesos diferentes.

O objetivo do problema é encontrar uma tabela-horário viável e que minimize a quantidade de violações fracas, portanto, um problema de minimização.

2.2 Conceitos e definições

As diversas formulações do problema apresentadas na literatura variam devido à instituição envolvida (escola ou universidade) e também pelo conjunto de restrições que são abordadas. Contudo, todas as formulações podem ser classificadas em uma das três classes principais [Schaerf 1995]:

- **Tabela-horário de Escola** Programação semanal para as turmas da escola, evitando que professores lecionem para duas turmas ao mesmo tempo ou que turmas assistam mais de uma aula no mesmo horário.
- **Tabela-horário de Universidade** Programação semanal das aulas das disciplinas universitárias, minimizando a sobreposição de aulas que tenham alunos em comum.

- **Tabela-horário de Exames** Programação para aplicação dos exames das disciplinas, evitando sobreposição de exames das disciplinas que tenham alunos em comum, e espalhando os exames para os alunos o máximo possível.

A tabela-horário de exames é usada somente nos finais de bimestre, trimestre ou semestre, enquanto as duas primeiras durante todo o período letivo. A diferença principal do problema no âmbito de escolas e universidades é que no segundo não há o conceito de turma, já que cada aluno escolhe o conjunto de disciplinas que irá cursar. Já nas escolas (primárias e secundárias) todos alunos matriculados numa turma cursam as mesmas disciplinas.

2.3 Abordagens existentes

As técnicas usadas para resolver o problema de tabela-horário são bastante diversificadas. As mais antigas, as heurísticas construtivas [Junginger 1986, Papoulias 1980], foram baseadas no modo humano de resolver o problema: as aulas eram alocadas uma a uma até finalizar a construção da tabela. Conflitos eram resolvidos realizando trocas de aulas.

[Neufeld e Tartar 1974] propôs uma redução do problema ao problema de coloração de grafos. As aulas são representadas como vértices do grafo. Arestas conectam aulas que não podem ser alocadas no mesmo horário. A coloração do grafo resultante é então transformada na tabela-horário: cada cor representa um horário de aula. Propostas similares a esta foram aplicadas em [Werra 1985, Selim 1988].

[Ostermann e Werra 1982] reduziram o problema de tabela-horário ao problema de fluxo de redes. As aulas são representadas pelos vértices. Uma rede é criada para cada horário e o fluxo na rede identifica as aulas que são lecionadas no mesmo horário. Posteriormente, [Werra 1985] usou uma idéia similar, mas cada fluxo representa uma classe e os vértices são horários e professores. Outras abordagens com fluxo de redes foram usadas em [Dinkel, Mote e Venkataramanan 1989, Chahal e Werra 1989].

De uns anos para cá as técnicas de solução para o problema têm recaído basicamente em três grupos: programação matemática, programação em lógica e principalmente meta-heurísticas.

Na área de programação matemática, bons resultados têm sido obtidos com programação inteira. Em [Lach e Lubbecke 2010] pode ser vista uma formulação completa do problema. Ela foi executada com CPLEX9 [IBM 2012] e conseguiu soluções com respostas bem próximas às melhores obtidas na competição ITC-2007. Em [Burke et al. 2008] pode ser vista outra formulação com algumas relaxações. Com um tempo de execução aproximado de quinze minu-

tos no CPLEX10, esse algoritmo conseguiu encontrar a solução ótima para duas instâncias da competição. Uma grande contribuição destes dois trabalhos é que eles forneceram limites inferiores para a quantidade de violações das restrições fracas para cada instância do ITC-2007. Em [Filho e Lorena 2006] pode ser vista uma modelagem em programação inteira mais específica para escolas brasileiras de ensino fundamental.

Alguns resultados relevantes também tem sido encontrados com programação em lógica. [Achá e Nieuwenhuis 2010] apresenta uma formulação usando *MaxSAT* em que se conseguiu melhorar quase metade das respostas que eram conhecidas à época para as instâncias do ITC-2007. [Gueret et al. 1995] e [Goltz e Matzke 1999] adotam formulações diferentes do ITC-2007, mas destacam como programação em lógica combinada com programação por restrições podem implementar modelos bem flexíveis, em que restrições podem ser adicionadas, modificadas ou excluídas com pouca alteração de código-fonte.

Pode ser visto em [Lewis 2007] que grande parte dos trabalhos recentes na área tem usado meta-heurísticas, tanto pela simplicidade quanto pelos bons resultados alcançados. Meta-heurísticas são estruturas gerais de algoritmos que podem ser adaptados para diferentes problemas de otimização necessitando em geral pouca implementação específica [Lewis 2007]. *Simulated Annealing*, Algoritmo Genético e Busca Tabu são as mais aplicadas. Há propostas com meta-heurísticas menos conhecidas, como a Busca de Harmonia [Al-Betar e Khader 2012].

Em [Elmohamed, Coddington e Fox 1998] foram investigadas diversas abordagens do *Simulated Annealing*. Dentre as configurações possíveis, os melhores resultados foram obtidos com resfriamento adaptativo, reaquecimento e um algoritmo baseado em regras para gerar uma boa solução inicial. [Ceschia, Gaspero e Schaerf 2011] usa SA para resolver a formulação *track 2* do ITC-2007. Neste trabalho foram obtidas boas respostas para as instâncias, e em alguns casos, foram conhecidas respostas melhores que às conhecidas à época.

Algoritmos Genéticos também foram propostos para o problema. O cromossomo representa a tabela-horário. Cada posição da tabela corresponde a um gene. A partir daí são definidas as operações genéticas de seleção, cruzamento e mutação, que visam gerar indivíduos cada vez melhores. Um indivíduo com bom valor *fitness* representa uma tabela-horário bem avaliada de acordo com a função objetivo associada ao problema. [Erben e Keppler 1995] aplicou esta técnica para tratar o problema de tabela-horário de universidades com muitas restrições, obtendo resultados promissores. Em [Kanoh e Sakamoto 2008] pode ser visto uma abordagem também para universidades, em que se utiliza uma base de conhecimento para gerar a população inicial. Essa base de conhecimento é formada com informações de tabelas-horário de anos anteriores. [Massoodian e Esteki 2008] propôs um algoritmo genético híbrido para resolver a formulação

do ITC-2007. O algoritmo foi chamado de híbrido porque além de aplicar os operadores genéticos, uma busca local era aplicada nos melhores indivíduos ao final de cada geração.

Um algoritmo mais simples que o genético, mas que também tem sido aplicado é a Busca Tabu. Em [Elloumi et al. 2008] é apresentado uma algoritmo para resolver o problema de tabela-horário de uma universidade tunisiana. [Santos, Ochi e Souza 2005] aplicou a técnica no âmbito de escolas primárias e obteve respostas melhores que outros algoritmos propostos com Busca Tabu. Em [Souza, Maculan e Ochi 2004] a Busca Tabu é aplicada como fase de busca local do GRASP, numa implementação também voltada para escolas.

Devido à dificuldade de otimização do problema, alguns autores têm proposto técnicas híbridas, em que são usadas mais de uma meta-heurística ou aplicadas de forma diferente. Exemplos deste tipo de implementação podem ser visto em [Massoodian e Esteki 2008] em que o algoritmo genético é combinado com um algoritmo de busca local para melhorar a qualidade das soluções. Em [Kostuch 2006] há um algoritmo para o ITC-2002 que constrói a tabela-horário em três etapas. Na primeira é usada um algoritmo de coloração de grafos para se obter uma solução inicial viável. Na segunda e terceira etapas aplica-se *Simulated Annealing*, mas cada uma usando uma estrutura de vizinhança diferente. Como dito anteriormente, em [Souza, Maculan e Ochi 2004] a Busca Tabu é inserida dentro da meta-heurística GRASP para melhorar a fase de busca local desta.

2.4 Trabalhos relacionados

O objetivo deste trabalho é propor um novo algoritmo usando a meta-heurística GRASP para o problema de tabela-horário de universidades. O GRASP já foi implementado no problema, mas para tabela-horário de escolas. Como mostrada na seção anterior, [Souza, Maculan e Ochi 2004] usou o GRASP juntamente com busca Tabu. Outra implementação do GRASP pode ser vista em [Vieira, Rafael e Scaraficci 2010], onde foi implementada a versão básica do algoritmo e uma estratégia de intensificação.

A formulação adotada neste trabalho é a do ITC-2007. Como foi visto na seção 2.3 muitos trabalhos foram propostos para resolver esta formulação. Mas nenhuma conseguiu obter soluções ótimas para todas as instâncias, o que gera uma motivação para continuar buscando novos resultados. Os melhores resultados encontrados até agora estão em [Lach e Lubbecke 2010, Burke et al. 2008, Achá e Nieuwenhuis 2010, Ceschia, Gaspero e Schaerf 2011].

3 Proposta do trabalho

"Nada se cria, nada se perde, tudo se transforma."

Lavousier

Neste capítulo é apresentado a proposta do trabalho ...

3.1 Formulação do ITC-2007

3.2 Algoritmo GRASP

Um problema de otimização combinatória pode ser definido por um conjunto finito $E = 1,...,n$, um conjunto de soluções viáveis $F \subseteq 2^E$ e uma função objetivo $f : 2^E \rightarrow \mathbb{R}$, todos definidos para cada problema específico. Considerando a versão de minimização, o objetivo consiste em encontrar uma solução ótima $S^* \in F$ tal que $f(S^*) \leq f(S), \forall S \in F$.

As meta-heurísticas são algoritmos gerais que orientam uma forma de explorar o conjunto de soluções para encontrar a solução ótima. A solução ótima não é garantida, mas elas em geral conseguem boas soluções.

A meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedures* - Procedimentos de busca aleatória, adaptativa e gulosa) foi introduzida por [Feo e Resende 1989] para tratar o problema de cobertura de conjuntos. De lá para cá ela já foi aplicada em vários problemas de otimização como conjunto independente máximo [Feo, Resende e Smith 1994], problema quadrático de alocação [Li, Pardalos e Resende 1994], satisfabilidade [Resende e Feo 1996], planarização de grafos [Resende e Ribeiro 1997], roteamento de circuitos virtuais [Resende e Ribeiro 2003], entre outros.

3.2.1 Estrutura básica

O algoritmo GRASP é um procedimento multi-iterativo. Cada iteração constrói uma solução inicial e aplica busca local para melhorá-la. A resposta final é a melhor obtida dentre as iterações. O algoritmo 1 apresenta o pseudo-código genérico do GRASP. Ao fim da fase de construção inicial, pode ocorrer de a solução obtida ser inviável. Por isso um passo intermediário é previsto para reparar a solução, tornando-a viável.

Algoritmo 1: Estrutura básica do algoritmo GRASP

Entrada: MaxIter

Saída: Solução S^*

```
1  f* ← ∞ ;
2  para i ← 1 até MaxIter faça
3      S ← GeraSolucaoInicial() ;
4      se S é inviável então
5          ReparaSolucao(S);
6      fim
7      S ← BuscaLocal(S) ;
8      se f(S) < f* então
9          S* ← S ;
10         f* ← f(S) ;
11     fim
12 fim
```

O método de construção da solução inicial do GRASP é guloso aleatório e visa produzir um conjunto diversificado de soluções iniciais de boa qualidade para a busca local. Algoritmos totalmente aleatórios conseguem essa diversificação, mas as soluções em geral são ruins considerando a função objetivo. Por outro lado, algoritmos gulosos tendem a gerar soluções de melhor qualidade, mas eles não conseguem produzir soluções diferentes já que a construção é sempre feita por escolhas gulosas.

O algoritmo 2 ilustra genericamente o procedimento de construção inicial. Ele recebe como parâmetro um conjunto de elementos que irão compor a solução. Inicialmente todos os elementos são candidatos a entrar na solução. Cada iteração escolhe um novo elemento para ser incorporado à solução. Essa escolha é gulosa aleatória. Primeiramente todos os candidatos são avaliados por uma função $g(c)$ para medir o custo de adicionar o candidato c à solução parcial S. Com base no menor e maior custo são escolhidos os candidatos mais bem avaliados para compor a lista restrita de candidatos - LRC. Um dos candidatos é escolhido aleatoriamente para entrar na solução e é retirado do conjunto de candidatos. O procedimento termina quando todos os elementos foram incorporados à solução.

O parâmetro $\alpha(0 \leq \alpha \leq 1)$ regula se o algoritmo será mais guloso ou mais aleatório. Quando α é mais próximo de zero somente os elementos com baixo custo irão entrar na LRC. Este comportamento produz soluções de boa qualidade porém pouco diversificadas. Com α mais próximo de um, elementos com custo mais alto poderão também entrar na LRC. Isto in-

troduz mais aleatoriedade à solução, mas em compensação se perde em qualidade. O ideal é encontrar um valor intermediário que permita diversificação sem prejudicar muito a qualidade de solução que será passada para a fase seguinte de busca local.

Algoritmo 2: Algoritmo guloso aletório para construção da solução inicial do GRASP

Entrada: E = {conjunto discreto finito}

Saída: Solução S

1 $S \leftarrow \emptyset$;
2 $C \leftarrow E$;
3 **enquanto** $|C| > 0$ **faça**
4 Para todo $c \in C$ computar o valor da função gulosa $g(c)$;
5 $c^{min} \leftarrow min\{g(c)|c \in C\}$;
6 $c^{max} \leftarrow max\{g(c)|c \in C\}$;
7 $LRC \leftarrow \{c \in C|g(c) \leq c^{min} + \alpha(c^{max} - c^{min})\}$;
8 Escolha aleatoriamente $c^{'} \in LRC$;
9 $S \leftarrow S \bigcup \{c^{'}\}$;
10 $C \leftarrow C - \{c^{'}\}$;
11 **fim**

Para encontrar boas soluções, uma meta-heurística precisa ter duas características importantes: diversificação e intensificação. A primeira se refere à capacidade de explorar bem o espaço de soluções e não ficar preso a mínimos ou máximos locais. A intensificação é necessária para explorar bem uma região de mínimo local, dado que o mínimo global necessariamente também é um mínimo local.

No GRASP a diversificação é feita pela independência das iterações e pela aleatoriedade introduzida na solução inicial. Assim a busca local irá trabalhar em diferentes soluções.

A intensificação é feita pela busca local. Nesta fase a solução inicial é melhorada explorando sua vizinhança na busca de soluções melhores. O GRASP não especifica qual a estratégia de busca local deve ser utilizada. Em [Feo, Resende e Smith 1994] a busca local implementada é conhecida como *Hill Climbing*. É uma estratégia simples em que se explora a vizinhança enquanto são encontradas soluções melhores. Já em [Souza, Maculan e Ochi 2004] por exemplo, é usado busca Tabu.

3.2.2 Hibridização com *Path-Relinking*

A heurística *Path-Relinking* (religamento de caminhos) foi originalmente proposta por [Glover 1996] como uma estratégia de intensificação na busca Tabu. A primeira proposta de uso do *Path-Relinking* no GRASP foi feita por [Laguna e Martí 1999]. Essa hibridização tenta resolver uma deficiência do GRASP que é ausência de memorização entre as iterações.

A idéia básica do *Path-Relinking* é traçar um caminho ligando duas soluções, que são chamadas de inicial e alvo. Para gerar este caminho, sucessivamente são inseridos atributos da solução alvo na solução inicial para que ela fique a cada iteração mais parecida com a solução alvo. A cada atributo inserido uma solução diferente é obtida. A melhor solução obtida no caminho é a resposta do algoritmo. Desta forma, o religamento de caminhos pode ser visto como uma estratégia que procura incorporar atributos de soluções de alta qualidade. O pseudo-código do algoritmo 3 ilustra o PR aplicado ao par de soluções x_i (inicial) e x_a (alvo).

O procedimento inicia computando a diferença simétrica $\Delta(x_i, x_a)$ entre as duas soluções, isto é, o conjunto de movimentos necessários para alcançar x_a a partir de x_i. Um caminho de soluções é gerado ligando x_i e x_a. A melhor solução x^* no caminho é retornada pelo algoritmo. Em cada iteração, o procedimento examina todos os movimentos $m \in \Delta(x, x_a)$ a partir da solução corrente x e seleciona aquele que resulta no custo de solução menor, isto é, aquele que minimiza $f(x \oplus m)$, onde $x \oplus m$ é a solução resultante da aplicação do movimento m à solução x. O melhor movimento m^* é feito, produzindo a solução $x \oplus m^*$. O conjunto de movimentos possíveis é atualizado. Se for o caso, a melhor solução x^* é atualizada. O procedimento termina quando x_a é alcançada, ou seja, $\Delta(x, x_a) = \emptyset$.

Algoritmo 3: Algoritmo *Path-Relinking*

Entrada: Solução inicial x_i, solução alvo x_a

Saída: Melhor solução x^* no caminho de x_i para x_a

$f^* \leftarrow min\{f(x_i), f(x_a)\}$;
$x^* \leftarrow argmin\{f(x_i), f(x_a)\}$;
$x \leftarrow x_i$;
Compute as diferenças simétricas $\Delta(x, x_a)$;
enquanto $\Delta(x, x_a) \neq \emptyset$ **faça**
    $m^* \leftarrow argmin\{f(x \oplus m) : m \in \Delta(x, x_a)\}$;
    $\Delta(x \oplus m^*, x_a) \leftarrow \Delta(x, x_a) - \{m^*\}$;
    $x \leftarrow x \oplus m^*$;
    se $f(x) < f^*$ **então**
        $f^* \leftarrow f(x)$;
        $x^* \leftarrow x$;
    fim
fim

O *Path-Relinking* pode ser visto com uma estratégia de busca local mais restrita, onde os movimentos aplicados são mais específicos.

De acordo com [Resende e Ribeiro 2005], *Path-Relinking* é uma melhoria ao algoritmo básico do GRASP, proporcionando melhoras tanto no tempo quanto na qualidade de solução.

Ainda de acordo com [Resende e Ribeiro 2005], existem duas estratégias diferentes de aplicar o PR no GRASP:

- PR é aplicado entre todos os pares de soluções elite. Pode ser feito periodicamente durante as iterações do GRASP ou no final da execução como uma pós-otimização.
- PR é aplicado como estratégia de intensificação em cada mínimo local obtido logo após a fase de busca local.

As duas estratégias podem ser combinadas.

Algoritmo 4: Algoritmo GRASP

Entrada: Instância p, MaxIter, TamPool

Saída: Solução S^*

```
1  f* ← ∞ ;
2  Pool ← ϕ ;
3  para i ← 1 até MaxIter faça
4      S ← GeraSolucaoInicial() ;
5      S ← BuscaLocal(S) ;
6      se i > TamPool então
7          Selecione aleatoriamente S_elite ∈ Pool ;
8          S ← PathRelinking(S, S_elite) ;
9      fim
10     se f(S) < f* então
11         S* ← S ;
12         f* ← f(S) ;
13     fim
14     AtualizaPool(S, Pool) ;
15 fim
```

4 Resultados Computacionais

“*Nada se cria, nada se perde, tudo se transforma.*”

Lavousier

Neste capítulo é apresentado os resultados computacionais ...

4.1 Descrição das instâncias utilizadas

As instâncias utilizadas foram as mesmas submetidas aos competidores do ITC-2007. São 21 instâncias ao todo, com grau de dificuldade variado. A organização garante que existe solução viável para todas as instâncias, fato que foi comprovado nos testes. Mas nada foi informado sobre a quantidade de violações fracas em cada instância. Em [PATAT 2008] podem ser obtidas todas as instâncias.

Na tabela 4.1 são apresentados os dados mais relevantes de cada instância. A quantidade de horários de aula numa semana não varia muito de instância para instância. A coluna conflitos conta a quantidade de pares de aula que não podem ser alocadas no mesmo horário (mesma disciplina, mesmo currículo ou mesmo professor) dividido pelo total de pares distintos de aula. A disponibilidade mede percentualmente a quantidade de horários que são disponíveis para as aulas, levando-se em conta as restrições de indisponibilidade que são informadas no arquivo de entrada.

A quantidade de currículos e disciplinas tem grande impacto no tempo de execução do algoritmo, pois são mais aulas para fazer a contagem total de violações. A quantidade de conflitos e disponibilidade influencia na dificuldade de encontrar uma solução viável, pois quanto mais conflitos e menos disponibilidade, menos horários existirão para alocar aula sem violar as restrições fortes. Além de dificultar a viabilidade no momento de geração da solução inicial, os conflitos e as disponibilidades dificultam a exploração da vizinhança na busca local, dado que muitas trocas acabam sendo descartadas por introduzirem violações das restrições fortes.

4.2 Detalhes de implementação

4.3 Sintonia da Meta-heurística

Duas estruturas de vizinhança foram usadas neste trabalho para a fase de busca local: *MOVE* e *SWAP*. Em [Ceschia, Gaspero e Schaerf 2011] essas duas estruturas também são usadas, só que aplicadas de maneira diferente. Na geração do vizinho sempre é usado o *MOVE*. Em alguns casos ocorre *SWAP* juntamente com o *MOVE*. Essa probabilidade de ocorrer o *SWAP* foi parametrizada. Experimentalmente foi escolhido um valor de 40%.

Testes mostraram que aplicar *MOVE* e *SWAP* separadamente é mais eficiente. Foi obser-

Instância	Currículos	Salas	Disciplinas	Horários por dia	Dias	Conflitos	Disponibilidade
comp01	14	6	30	6	5	13.2	93.1
comp02	70	16	82	5	5	7.97	76.9
comp03	68	16	72	5	5	8.17	78.4
comp04	57	18	79	5	5	5.42	81.9
comp05	139	9	54	6	6	21.7	59.6
comp06	70	18	108	5	5	5.24	78.3
comp07	77	20	131	5	5	4.48	80.8
comp08	61	18	86	5	5	4.52	81.7
comp09	75	18	76	5	5	6.64	81
comp10	67	18	115	5	5	5.3	77.4
comp11	13	5	30	9	5	13.8	94.2
comp12	150	11	88	6	6	13.9	57
comp13	66	19	82	5	5	5.16	79.6
comp14	60	17	85	5	5	6.87	75
comp15	68	16	72	5	5	8.17	78.4
comp16	71	20	108	5	5	5.12	81.5
comp17	70	17	99	5	5	5.49	79.2
comp18	52	9	47	6	6	13.3	64.6
comp19	66	16	74	5	5	7.45	76.4
comp20	78	19	121	5	5	5.06	78.7
comp21	78	18	94	5	5	6.09	82.4

Tabela 4.1: Tabela com informações sobre cada instância do ITC-2007

vado que separadamente eles produzem melhores soluções e de forma mais rápida. Cada um ocorre com a mesma probabilidade de 50%.

- Temperatura inicial do SA para busca local

4.4 Análise dos resultados

5 Conclusões e trabalhos futuros

"*Nada se cria, nada se perde, tudo se transforma.*"

Lavousier

Neste capítulo é apresentado as conclusões e alguns trabalhos futuros ...

5.1 Conclusões

Alguns itens interessantes para a conclusão de um projeto de graduação

Qual foi o resultado do seu trabalho? melhora na área, testes positivos ou negativos? Você acha que o mecanismo gerado produziu resultados interessantes? Quais os problemas que você encontrou na elaboração do projeto? E na implementação do protótipo? Que conclusão você tirou das ferramentas utilizadas? (heurísticas, prolog, ALE, banco de dados). Em que outras áreas você julga que este trabalho seria interessante de ser aplicado? Que tipo de continuidade você daria a este trabalho? Que tipo de conhecimento foi necessário para este projeto de graduação? Para que serviu este trabalho na sua formação?

Referências Bibliográficas

[Achá e Nieuwenhuis 2010]ACHá, R. A.; NIEUWENHUIS, R. Curriculum-based course timetabling with sat and maxsat. *Annals of Operations Research*, Springer Netherlands, p. 1–21, 2010. ISSN 0254-5330. 10.1007/s10479-012-1081-x. Disponível em: <http://dx.doi.org/10.1007/s10479-012-1081-x>.

[Al-Betar e Khader 2012]AL-BETAR, M.; KHADER, A. A harmony search algorithm for university course timetabling. *Annals of Operations Research*, Springer Netherlands, v. 194, p. 3–31, 2012. ISSN 0254-5330. 10.1007/s10479-010-0769-z. Disponível em: <http://dx.doi.org/10.1007/s10479-010-0769-z>.

[Burke et al. 2008]BURKE, E. K. et al. A branch-andcut procedure for the udine course timetabling. In: *Problem, Proceedings of the 7th PATAT Conference, 2008.* [S.l.: s.n.], 2008.

[Ceschia, Gaspero e Schaerf 2011]CESCHIA, S.; GASPERO, L. D.; SCHAERF, A. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, v. 39, n. 7, p. 1615–1624, 2011. ISSN 0305-0548.

[Chahal e Werra 1989]CHAHAL, N.; WERRA, D. D. An interactive system for constructing timetables on a pc. *European Journal of Operational Research*, Elsevier, v. 40, n. 1, p. 32–37, 1989.

[Dinkel, Mote e Venkataramanan 1989]DINKEL, J.; MOTE, J.; VENKATARAMANAN, M. Or practice - an efficient decision support system for academic course scheduling. *Operations Research*, INFORMS, v. 37, n. 6, p. 853–864, 1989.

[Elloumi et al. 2008]ELLOUMI, A. et al. A tabu search procedure for course timetabling at a tunisian university. In: *Proceedings of the 7th PATAT Conference, 2008.* [S.l.: s.n.], 2008.

[Elmohamed, Coddington e Fox 1998]ELMOHAMED, M. A. S.; CODDINGTON, P.; FOX, G. A comparison of annealing techniques for academic course scheduling. In: *Lecture Notes in Computer Science*. [S.l.: s.n.], 1998.

[Erben e Keppler 1995]ERBEN, W.; KEPPLER, J. *A Genetic Algorithm Solving a Weekly Course-Timetabling Problem*. 1995.

[Feo e Resende 1989]FEO, T.; RESENDE, M. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, v. 8, p. 67–71, 1989.

[Feo, Resende e Smith 1994]FEO, T.; RESENDE, M.; SMITH, S. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, v. 42, p. 860–878, 1994. Disponível em: <http://www.research.att.com/ mgcr/doc/gmis.ps.Z>.

[Filho e Lorena 2006]FILHO, G. R.; LORENA, L. An integer programming model for the school timetabling problem. In: CITESEER. *XIII CLAIO: Congreso Latino-Iberoamericano de Investigación Operativa*. [S.l.], 2006.

[Glover 1996]GLOVER, F. Tabu search and adaptive memory programing - advances, applications and challenges. In: *Interfaces in Computer Science and Operations Research*. [S.l.]: Kluwer, 1996. p. 1–75.

[Goltz e Matzke 1999]GOLTZ, H.-J.; MATZKE, D. University timetabling using constraint logic programming. In: *PRACTICAL ASPECTS OF DECLARATIVE LANGUAGES*. [S.l.]: Springer-Verlag, 1999. p. 320–334.

[Gotlieb 1962]GOTLIEB, C. C. The construction of class-teacher time-tables. In: *IFIP Congress*. [S.l.: s.n.], 1962. p. 73–77.

[Gueret et al. 1995]GUERET, C. et al. *Building University timetables using Constraint Logic Programming*. 1995.

[IBM 2012]IBM. *IBM ILOG CPLEX Optimization Studio*. 2012. Disponível em: <http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/>.

[Insecure org 2002]Insecure org. *The Network Explotation Tool and Security Scanner*. 2002. URL: http://www.insecure.org/nmap. Last Visited: 22/07/2002.

[Junginger 1986]JUNGINGER, W. Timetabling in Germany – A survey. *Interfaces*, v. 16, n. 4, p. 66–74, 1986.

[Kanoh e Sakamoto 2008]KANOH, H.; SAKAMOTO, Y. Knowledge-based genetic algorithm for university course timetabling problems. *Int. J. Know.-Based Intell. Eng. Syst.*, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 12, n. 4, p. 283–294, dez. 2008. ISSN 1327-2314. Disponível em: <http://dl.acm.org/citation.cfm?id=1460198.1460201>.

[Kostuch 2006]KOSTUCH, P. *The University Course Timetabling Problem with a 3-phase approach*. 2006.

[Lach e Lubbecke 2010]LACH, G.; LUBBECKE, M. E. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 2010.

[Laguna e Martí 1999]LAGUNA, M.; MARTÍ, R. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, v. 11, p. 44–52, 1999.

[Lewis 2007]LEWIS, R. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, v. 30, n. 1, p. 167–190, 2007.

[Li, Pardalos e Resende 1994]LI, Y.; PARDALOS, P.; RESENDE, M. A greedy randomized adaptive search procedure for the quadratic assignment problem. In: PARDALOS, P.; WOLKOWICZ, H. (Ed.). *Quadratic assignment and related problems*. American Mathematical Society, 1994, (DIMACS Series on Discrete Mathematics and Theoretical Computer Science, v. 16). p. 237–261. Disponível em: <http://www.research.att.com/ mgcr/doc/grpqap.ps.Z>.

[Massoodian e Esteki 2008]MASSOODIAN, S.; ESTEKI, A. A hybrid genetic algorithm for curriculum based course timetabling. In: *Proceedings of the 7th PATAT Conference, 2008*. [S.l.: s.n.], 2008.

[Neufeld e Tartar 1974]NEUFELD, G. A.; TARTAR, J. Graph coloring conditions for the existence of solutions to the timetable problem. *Commun. ACM*, ACM, New York, NY, USA, v. 17, n. 8, p. 450–453, ago. 1974. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/361082.361092>.

[Ostermann e Werra 1982]OSTERMANN, R.; WERRA, D. Some experiments with a timetabling system. *Operations-Research-Spektrum*, Springer-Verlag, v. 3, p. 199–204, 1982. ISSN 0171-6468. Disponível em: <http://dx.doi.org/10.1007/BF01719788>.

[Papoulias 1980]PAPOULIAS, D. B. The assignment-to-days problem in a school time-table, a heuristic approach. *European Journal of Operational Research*, v. 4, n. 1, p. 31–41, 1980. Disponível em: <http://EconPapers.repec.org/RePEc:eee:ejores:v:4:y:1980:i:1:p:31-41>.

[PATAT 2008]PATAT. *International Timetabling Competition*. 2008. URL: http://www.cs.qub.ac.uk/itc2007.

[Resende e Feo 1996]RESENDE, M.; FEO, T. A GRASP for satisfiability. In: JOHNSON, D.; TRICK, M. (Ed.). *The Second DIMACS Implementation Challenge*. American Mathematical Society, 1996, (DIMACS Series on Discrete Mathematics and Theoretical Computer Science, v. 26). p. 499–520. Disponível em: <http://www.research.att.com/ mgcr/doc/grpsat-long.ps.Z>.

[Resende e Ribeiro 1997]RESENDE, M.; RIBEIRO, C. A GRASP for graph planarization. *Networks*, v. 29, p. 173–189, 1997. Disponível em: <http://www.research.att.com/ mgcr/doc/gmpsg.ps.Z>.

[Resende e Ribeiro 2003]RESENDE, M.; RIBEIRO, C. A GRASP with path-relinking for private virtual circuit routing. *Networks*, v. 41, n. 1, p. 104–114, 2003.

[Resende e Ribeiro 2005]RESENDE, M. G. C.; RIBEIRO, C. C. *GRASP with Path-Relinking: Recent Advances and Applications*. 2005.

[Saint Corporation 2002]Saint Corporation. *Security Administrator's Integrated Tool*. 2002. URL: http://www.saintcorporation.com/index.html. Last Visited: 22/07/2002.

[Santos, Ochi e Souza 2005]SANTOS, H. G.; OCHI, L. S.; SOUZA, M. J. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 10, dez. 2005. ISSN 1084-6654. Disponível em: <http://doi.acm.org/10.1145/1064546.1180621>.

[Schaerf 1995]SCHAERF, A. A survey of automated timetabling. *ARTIFICIAL INTELLIGENCE REVIEW*, v. 13, p. 87–127, 1995.

[Selim 1988]SELIM, S. M. Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. *Comput. J.*, Oxford University Press, Oxford, UK, v. 31, n. 1, p. 76–82, fev. 1988. ISSN 0010-4620. Disponível em: <http://dx.doi.org/10.1093/comjnl/31.1.76>.

[Souza, Maculan e Ochi 2004]SOUZA, M. J. F.; MACULAN, N.; OCHI, L. S. Metaheuristics. In: RESENDE, M. G. C.; SOUSA, J. P. de; VIANA, A. (Ed.). Norwell, MA, USA: Kluwer Academic Publishers, 2004. cap. A GRASP-tabu search algorithm for solving school timetabling problems, p. 659–672. ISBN 1-4020-7653-3. Disponível em: <http://dl.acm.org/citation.cfm?id=982409.982441>.

[The Nessus Project 2002]The Nessus Project. *The Nessus Project*. 2002. URL: http://www.nessus.org. Last Visited: 22/07/2002.

[Vieira, Rafael e Scaraficci 2010]VIEIRA, A.; RAFAEL, M.; SCARAFICCI, A. *A GRASP Strategy for a More Constrained School Timetabling Problem*. 2010.

[Werra 1985]WERRA, D. de. An introduction to timetabling. *European Journal of Operational Research*, v. 19, n. 2, p. 151–162, 1985. Disponível em: <http://EconPapers.repec.org/RePEc:eee:ejores:v:19:y:1985:i:2:p:151-162>.

ANEXO A – Ferramentas utilizadas

Foi feita uma análise de algumas ferramentas que são muito usadas por atacantes (hackers) para a confecção de ataques. Estas ferramentas são muito úteis em vários aspectos, tais como: (1) o levantamento de informações sobre o alvo, (2) que tipo de serviços estão disponíveis no alvo, (3) quais as possíveis vulnerabilidades do alvo, entre outras informações. As ferramentas analisadas foram o *nmap* [Insecure org 2002], o *nessus* [The Nessus Project 2002], o *saint* [Saint Corporation 2002], além de alguns comandos de sistemas operacionais (UNIX-Like e Windows-Like) usados para rede, tais como o *ping, nslookup e whois*.