Solução de um problema de *timetabling* de universidades com o algoritmo dos casais estáveis

Eric Prates dos Santos[1], Fernando Sarturi Prass[2]

[1]Bacharelando em Ciência da Computação – Centro Universitário Franciscano – Santa Maria – RS – Brasil

[2]Mestre em Ciências da Computação – Universidade Federal de Santa Catarina (UFSC)

ericprates@unifra.edu.br, fprass@gmail.com

Abstract. *The stable marriage problem is a combinatorial optimization problem solved by the deferred acceptance algorithm- This paper reports the creation of a prototype to test its applicability in solving the university timetabling problem, modeled as a combination of resources and events. A literature review examining the theoretical aspects of both problems is presented as well as a discussion of the results. It is shown a many-to-one Gale-Shapley algorithm can generate valid timetables.*

Resumo. *O problema dos casais estáveis, conforme definido por Gale e Shapley (1962), é um problema de otimização combinatória resolvido pelo algoritmo de aceitação diferida. Este trabalho relata a criação de um protótipo que testa sua aplicabilidade na solução do problema de timetabling de universidades, modelado como uma combinação de recursos e eventos. Uma revisão bibliográfica examinando os aspectos teóricos de ambos os temas é apresentada, bem como uma discussão dos resultados obtidos. É mostrado que a variante poligâmica do algoritmo de Gale e Shapley consegue gerar tabelas de horários válidas.*

1. Introdução

Em Matemática, alguns dos problemas estudados não dizem respeito a valores, numéricos ou geométricos, nem pressupõem conhecimento em Cálculo [Gale e Shapley 1962]. Considere o problema de formar casais entre um grupo de amigos. Tomando-se um número de homens e mulheres solteiros, certamente eles terão seus possíveis pares em uma ordem de preferência; esta ordem será ao menos qualitativa, abstrata. Se há a pretensão de formar casais para todos, é necessário garantir que os mesmos não se desfaçam porque há a possibilidade de alguém abandonar seu par em nome de outro(a) mais bem colocado(a) em sua lista de preferência. Apenas conhecendo tais preferências, será possível formar ao menos um arranjo de casais para este grupo de amigos, garantindo que nenhum par se dissolva?

[Gale e Shapley 1962] apresentaram este problema como "o problema dos casais estáveis", um problema combinatório [Farczadi *et al.* 2014], determinístico polinomial[1]

1 Um problema decisório que pode ser resolvido por uma máquina de Turing determinística em um número de passos restrito por uma função polinomial do tamanho da entrada; problemas desta natureza pertencem à categoria elementar "P" [Cook 2000].

[Ronn 1990], *strategy-proof*[2] [Liu e Chiu 2010], que consiste em encontrar um arranjo estável entre elementos de dois conjuntos e é alvo de grande atenção acadêmica, especialmente no estudo da complexidade de suas muitas variações [Farczadi *et al.* 2014; Irving 2008; Iwama e Miyazaki 2008].

O algoritmo de aceitação diferida que resolve este problema é aplicado, com modificações, na seleção de estudantes para programas de residência, estágios, faculdades e escolas primárias e secundárias em diversos países [Biró 2007; Cechlárová *et al.* 2015], podendo ser aplicado de um modo geral a qualquer mercado de dois lados. O mesmo tem encontrado espaço para aplicações em campos como detecção de código clonado [Alhakami *et al.* 2014], agendamento de táxis [Bai *et al.* 2013] e alocação de recursos para as futuras redes 5G de celular [Hasan e Hossain 2015].

Por sua vez, a automação de horários por computador, chamada comumente de *timetabling,* é um problema enfrentado continuamente no meio educacional com uma diversidade de formulações possíveis que torna-se complexo dadas as restrições apresentadas no caso particular [Gröbner *et al.* 2003; Pillay 2014; Schmidt e Strohlein 1980]. Numa formulação geral, entende-se este problema como a alocação de professores e disciplinas a serem ministradas em determinados grupos de horários, satisfazendo-se certas restrições [Kristiansen e Stidsen 2013; Willemen 2002].

As principais soluções computacionais publicadas nas últimas duas décadas para uma automação eficiente de horários ou advém de sistemas *expert* [Smolira *et al.* 2003], ou aplicam técnicas de otimização de busca como pesquisa tabu, resfriamento simulado e algoritmos genéticos[3] [Moreira 2008], ou são parte da experimentação em torno da programação lógica com restrições [Pillay 2014]. Dado que a construção de uma tabela de horários para uma universidade é um problema, de modo geral, NP-completo[4] [Amorim 2012; Pillay 2014], é fundamental modelar o mesmo de maneira a obter uma solução em tempo computacional hábil [Willemen 2002].

Se entende-se que o problema da construção de tabelas de horários é um problema de alocação, isto é, um problema combinatório, levanta-se então a hipótese: pode o algoritmo que resolve o problema dos casais estáveis ser aplicado à automação de horários, seja com ou sem modificações? Se não pode, por que razão?

Este trabalho relata a modelagem, desenvolvimento e testes de um protótipo de programa para automação de horários de universidades, formulando um caso genérico de alocação de professores, disciplinas, turmas e horários com algumas restrições comumente encontradas em universidades e apresentadas pela literatura. Mostra-se que o algoritmo de aceitação diferida pode ser satisfatoriamente aplicado para resolução de um problema de *timetabling* com baixo custo computacional e poucas quebras de restrições fracas.

---

2 "Um mecanismo é strategy-proof se apresenta incentivos para que agentes revelem suas verdadeiras preferências enquanto participam do processo de alocação" [Alcalde e Barbera 1994].

3 Mais informações sobre estas técnicas, estudadas na área de Inteligência Artificial, estão disponíveis nos capítulos 2, 3 e 10 de [Glover e Kochenberger 2003].

4 Um problema decisório é NP-completo se e somente se todos os outros problemas em NP podem ser reduzidos a ele em tempo polinomial [Cook 2000; Goldreich 2008].

1.1. Como este trabalho está dividido

No capítulo 1, apresentou-se uma introdução ao tema a ser abordado. O restante está assim dividido:

O capítulo 2 apresenta o problema dos casais estáveis, sua definição matemática bem como sua prova e discute quais as aplicações modernas do algoritmo que resolve o mesmo através de uma revisão bibliográfica.

O capítulo 3 traz um contraponto com exemplos de soluções encontradas para o problema da automação de horários, apontando sua complexidade e formulando um caso genérico de *timetabling* universitária, com suas características e restrições.

No capítulo 4, propõe-se o desenvolvimento de um protótipo para resolver tal caso com o uso do algoritmo de aceitação diferida, com definição e justificativa das metodologias utilizadas.

Por fim, no capítulo 5 discute-se a modelagem e implementação deste protótipo avaliando-se os resultados obtidos com o mesmo baseado em um caso de teste real, seguindo-se o capítulo 6 com as conclusões alcançadas.

2. O problema dos casais estáveis

Considere-se dois conjuntos com homens e mulheres solteiros e um arranjo de casais entre eles. Quando formados estes casais, eles só permanecerão juntos se nenhum dos homens ou mulheres preferir outra mulher ou homem àquele ao qual foi assinalado e, ao mesmo tempo, for preferido por esta ou este. Um conjunto de casamentos é dito instável "se sob ele há, pelo menos, um homem e uma mulher que não estão casados mas que preferem um ao outro do que a seus parceiros atuais" [Gale e Shapley 1962]. Perguntam os autores: é sempre possível encontrar um arranjo estável de casamentos dados dois conjuntos de tamanho arbitrário com suas listas de preferência?

2.1. Enunciação matemática do problema (teorema)

Tome-se dois conjuntos finitos disjuntos A e B. Atribua-se a cada elemento destes conjuntos uma lista $\succ_e$ contendo elementos do conjunto oposto, estritamente ordenados por preferência. Existe pelo menos um conjunto de pares (a, b) disjuntos tal que não haja quaisquer dois elementos em pares diferentes que simultaneamente prefiram-se a seus pares assinalados (regra da estabilidade) [Gale e Shapley 1962; Iwama e Miyazaki 2008].

2.2. Prova construtiva: o algoritmo de aceitação diferida

Os passos definidos para a solução deste problema são, matematicamente, os que seguem [Gale e Shapley 1962; Roth 2008].

(Passo 0). Tome-se as listas de preferências de cada elemento em ambos os conjuntos. Escolha-se um conjunto, A, e assinale-se um par, sugerido, (a, b) onde $a \in A$, $b \in B$ e b é o primeiro elemento na lista de preferências $_a$.

(Passo 1). Tome-se os pares sugeridos e aceite-se, para cada b, somente aquele onde a é o elemento mais bem colocado na lista de preferências $?_b$. Resta uma lista de

pares, tentativos, (a, b).

(Passo 2). Assinale-se, então, um novo par sugerido (a, b) a cada a ∈ A que não tenha um par tentativo assinalado, onde b ∈ B e é o elemento seguinte na lista de preferências $?_a$. Tome-se os novos pares sugeridos e, caso a ∈ A (sugerido) seja um elemento mais bem colocado na lista de preferências $?_b$ do que seu par (tentativo) atual, forma-se um novo par, tentativo, (a, b).

(Passo 3). Se ainda há elementos a ∈ A sem par tentativo assinalado e que não foram assinalados a todos os elementos de B em algum momento, repete-se o passo 2. Caso contrário, obtém-se um conjunto de pares (a, b) que obedecem à regra da estabilidade.

Este algoritmo, em sua analogia com arranjos de casamentos, constitui uma prova construtiva válida para o problema. Ele é comumente chamado algoritmo de aceitação diferida, visto que o conjunto que recebe as propostas para formar pares não necessariamente as aceita de primeiro momento, mas aguarda por melhores propostas até o fim do algoritmo [Roth 2008].

Pode ser demonstrado que:

- O algoritmo de aceitação diferida sempre termina, e retorna arranjos estáveis de homens e mulheres [Gale e Shapley 1962; Roth 2008];
- Uma instância do problema dos casais tem ao menos uma solução encontrada por este algoritmo [Gale e Shapley 1962; Iwama e Miyazaki 2008];
- A solução encontrada é ótima para o lado que faz as propostas de casamento [Iwama e Miyazaki 2008];
- O procedimento é *strategy-proof* pelo menos para os elementos de um de seus conjuntos e roda em tempo polinomial (quadrático) [Bai *et al.* 2013; Gale e Shapley 1962; Liu e Chiu 2010].

2.3. Aplicações modernas do problema dos casais estáveis

Dada sua natureza genérica [Roth 2008], o algoritmo de aceitação diferida aparece em trabalhos com aplicações diversas à apresentada originalmente. Destaca-se a seguir dois exemplos de aplicações recentes do mesmo.

[Alhakami *et al.* 2014] investigam a possibilidade de adaptação do algoritmo para uso na detecção de código-fonte clonado. Apresentando diferentes métricas de qualidade de software, os autores propõem que as mesmas sejam aplicadas a fragmentos de código-fonte, construindo assim uma lista de preferências estrita entre dois conjuntos baseada na similaridade daqueles. Adaptando o algoritmo de aceitação diferida com uma estratégia própria, relaciona-se também múltiplas porções de código a uma única porção, obtendo-se resultados na detecção de clones "com modificações além das sintáticas", uma solução que "vence competitivamente algumas estratégias já existentes em vários aspectos".

Outro problema é o roteamento de táxis. Em locais onde a maioria dos táxis trabalham de maneira independente, estabelece-se uma competição entre os diferentes motoristas, ou por definição, um jogo não-cooperativo. Este problema é solucionado

quando cada jogador adota uma estratégia e nenhum deles obtém resultado final melhor por mudar a sua, o que se traduz, no roteamento de táxis, em fazer uma combinação estável de táxis e passageiros. Em [Bai *et al.* 2013], utiliza-se precisamente da natureza *strategy-proof* do algoritmo de aceitação diferida para encontrar, com baixo custo computacional, uma combinação estável de táxis e passageiros na rua de modo que nenhum motorista encontre melhor escolha do que a rota e passageiro que lhe foram assinalados.

Outros exemplos dignos de nota são a alocação de recursos para as futuras redes 5G de celular, atualmente em desenvolvimento [Hasan e Hossain 2015], e extensões do problema de alocação de residentes de Medicina [Roth 2008]. Apresentadas estas pesquisas, fica claro que o algoritmo de aceitação diferida pode ser aplicado a problemas combinatórios genéricos, estendendo-se a áreas como a teoria dos jogos, engenharia de software, etc. Deduz-se como características compartilhadas entre os problemas: que envolvem diferentes "mercados" (conjuntos de entidades e dados); que pode ser definida uma lista de preferências qualitativa ou quantitativa entre os membros de diferentes lados do problema; que os arranjos retornados pelo algoritmo são estáveis.

3. Abordagens atuais no problema da automação de horários

A criação de tabelas de horários (*timetabling*) é uma atividade relevante a muitas áreas da vida humana que demanda grande esforço otimizatório. A área educacional está constantemente preocupada com este problema, que é descrito de formas diferentes de acordo com uma série de restrições, regras, custos etc, de tal modo que "é impossível escrever uma única formulação do problema" [Ceschia *et al.* 2012]. A tarefa confronta principalmente diversos conflitos de horários ao buscar combinar professores, salas, disciplinas e outros fatores numa grade geralmente semanal [Pillay 2014].

Algumas formulações deste problema são a automação de horários de universidades, a de escolas secundárias, e a automação da geração de um período de exames, tendo todas variáveis análogas [Gröbner *et al.* 2003; Moreira 2008; Pillay 2014]. Este trabalho limita-se a abordar o problema de *timetabling* universitário, não discriminando nem ignorando, entretanto, as definições e técnicas utilizadas para um ou outro caso, que foram analisadas livremente para traçar um panorama da automação de horários.

3.1. Formalização do problema e definição de um caso genérico

Numa abordagem computacional, a automação de horários está relacionada a problemas de agendamento sequencial [Moreira 2008], sendo classificada como um problema NP-difícil ou NP-completo de acordo com a natureza das restrições apresentadas [Pillay 2014].

Construir uma solução para automação de horários altamente especializada, talhada em conformidade com as restrições de um caso, implica que a terminologia definida será extremamente particular ao caso abordado e coloca em risco a obtenção de resultados aceitáveis quando da aplicação da mesma solução e suas estruturas de dados a problemas apenas ligeiramente diferentes. Com esta argumentação, em [Gröbner *et al.* 2003] apresenta-se uma terminologia universal para a formalização do problema, que

será utilizada a seguir para descrever um caso genérico de automação de horários para uma universidade.

Tome-se os elementos de um problema de alocação de horários em três categorias: eventos, recursos e restrições. Elementos a ser agendados podem ser recursos ou eventos. Recursos são entidades do problema que podem ligar-se entre si sem propriedades temporais. Um evento é um elemento com uma propriedade temporal (reunião, horário de aula). Uma restrição do problema pode ser forte ou fraca: restrições fortes jamais podem ser violadas, enquanto que uma solução é melhor que outra se violar menos restrições fracas. Uma alocação de horários, então, é uma sequência arbitrária de eventos que não viola nenhuma restrição forte [Gröbner *et al.* 2003].

Numa universidade arbitrária, por exemplo, recursos são professores, turmas e disciplinas. Horários de aula de um período de um determinado curso são eventos; a aula de sexta-feira, das sete às nove da manhã, para o segundo período do curso de Ciência da Computação é estritamente diferente da aula, no mesmo horário, para o sétimo período do curso, ou para o primeiro período do curso de Sistemas de Informação. Como restrições fortes, pode-se estabelecer [Schmidt e Strohlein 1980; Smolira *et al.* 2003]:

1. Nenhuma turma de alunos (grupo ao qual são ofertadas disciplinas de um mesmo período) deve ser assinalada a mais de uma disciplina no mesmo horário;
2. Nenhum professor deve ser assinalado a mais de uma disciplina no mesmo horário;
3. Uma disciplina deve ser assinalada ao número de horários requerido;
4. Nenhum professor deve ser assinalado a mais horários do que sua carga horária requer;
5. Nenhum professor deve ser assinalado a uma disciplina no turno da manhã sem que seja respeitado o descanso noturno de 11 horas após a última aula ministrada no turno da noite (por questões legais).

Como restrições fracas, sugere-se que [Bellio *et al.* 2014; Pillay 2014; Smolira *et al.* 2003]:

1. Nenhum professor deve ser assinalado a um horário livre entre duas disciplinas;
2. Nenhuma turma deve ser assinalada a um horário livre entre duas disciplinas;
3. Nenhuma turma deve ser assinalada a apenas um horário no dia;
4. Disciplinas do mesmo período devem ser alocadas em horários adjacentes.

Uma alocação de horários bem-sucedida para esta universidade, então, é um conjunto de disciplinas, professores, turmas e horários que não viole nenhuma das restrições fortes dadas, buscando ao mesmo tempo violar o mínimo possível de restrições fracas.

3.2. Técnicas geralmente utilizadas

Dois estudos recentes sobre o problema de automação de horários, [Pillay 2014] e

[Kristiansen e Stidsen 2013], trazem um resumo das técnicas aplicadas ao longo das últimas décadas para solução do mesmo. Em especial, algoritmos genéticos, pesquisa tabu e resfriamento simulado têm sido explorados.

Algoritmos genéticos buscam representar dados de um problema como indivíduos de uma população, simulando a reprodução e mutação dos mesmos e a seleção dos indivíduos mais apropriados para a próxima geração. Por tratar-se de método estocástico sem garantia de parada, a avaliação das soluções geradas não é óbvia, e uma condição de parada satisfatória deve ser planejada para o algoritmo com a observação do melhoramento genético obtido a cada geração [Moreira 2008].

Meta-heurísticas de busca, como resfriamento simulado e pesquisa tabu, também são apontadas como soluções possíveis para o problema, sendo geralmente aplicadas em conjunto para acelerar a geração de um resultado viável. [Bellio *et al.* 2014] obtiveram soluções melhores que metade dos casos na literatura com tempos de execução razoáveis, mas como a maioria das pesquisas desta natureza, resolvem apenas um problema de *timetable* bem definido e carecem de teste geral [Pillay 2014].

Também podem ser citadas as abordagens de sistemas *expert*, isto é, *softwares* que imitam o comportamento de um usuário [Gervás e Miguel 1999], e a programação por restrições lógicas, que modela os problemas partindo diretamente de suas restrições [Pillay 2014].

Conclui-se que as técnicas mais utilizadas em problemas de automação de horários são algoritmos de otimização de busca. Isto ressalta o fato de que resolver um problema de agendamento sequencial como esse é, intrinsecamente, resolver um problema de otimização. A eficiência da solução apresentada em cada pesquisa, por sua vez, está ligada à construção da solução de forma a minimizar as quebras de restrições.

4. Um protótipo para automação de horários utilizando o algoritmo de aceitação diferida

Apresentado o problema dos casais estáveis e o problema de *timetabling,* e relacionada a natureza combinatória de ambos, relata-se a seguir o desenvolvimento de um protótipo onde o algoritmo de aceitação diferida foi aplicado para gerar tabelas de horários de uma universidade.

No contexto universitário, coordenadores de curso geralmente são os responsáveis pela criação de uma tabela de horários que satisfaça as restrições apresentadas. Cabe aos mesmos a decisão de qual professor selecionar para ministrar uma disciplina, e a alocação de salas para diferentes turmas, ficando em aberto a tarefa da alocação de horários para cada disciplina de cada período. Apresenta-se a seguir requisitos e metodologia de desenvolvimento de um protótipo de *software* que atende estas necessidades de negócio, isto é, que gera automaticamente uma tabela de horários satisfatória conhecidos os horários dos professores e o professor assinalado a cada disciplina. O usuário final deste sistema é um coordenador de curso.

4.1. Requisitos de software

A execução do software está condicionada aos seguintes requisitos:

- Permitir o gerenciamento dos diversos cursos disponíveis em uma determinada instituição;
- Permitir o gerenciamento de professores e associação ao mesmos de dados pessoais, uma carga horária, uma lista de disciplinas e ao menos um curso de afiliação;
- Permitir o gerenciamento de disciplinas a serem ofertadas para cada curso, fazendo distinção entre disciplinas de mesmo nome associadas a diferentes cursos, bem como disciplinas de mesmo nome e curso associadas a diferentes períodos e disciplinas de mesmo nome, curso e período associadas a diferentes turmas;
- Permitir o gerenciamento dos horários disponíveis de cada professor cadastrado, bem como a geração de uma tabela de horários para cada curso, ou para todos os seus semestres ou para semestres pares ou ímpares, obedecendo às restrições fortes e minimizando as quebras de restrições fracas, elencadas da seção 3.1.

4.2. Metodologia de desenvolvimento

O protótipo foi codificado em linguagem *Java*, com uso de um banco de dados relacional *MySQL.* A metodologia de desenvolvimento foi a *Cowboy* com técnicas ágeis.

Java é uma linguagem de programação de código aberto apontada como uma das mais ensinadas para programadores iniciantes em universidades, destacando-se pela disponibilidade de ambientes de desenvolvimento integrado (IDE) amigáveis para o usuário, exigência de bons hábitos de programação e demanda na indústria [Farooq *et al.* 2014], características que satisfazem as necessidades de produção do protótipo sugerido. *MySQL*, por sua vez, é um sistema de gerenciamento de banco de dados que satisfaz, frente a outras alternativas do mercado, critérios básicos de seleção como custo, funcionalidade, facilidade de uso e segurança [Denton e Peace 2003].

Dada a necessidade de desenvolvimento "solitário" do protótipo, utilizou-se a metodologia *Cowboy* apresentada em [Hollar 2006], um processo de desenvolvimento criado ao redor da ideia do programador dito *cowboy:* aquele que é obrigado por circunstâncias a programar sozinho e acaba não adotando qualquer das metodologias formais geralmente adotadas por um time. A metodologia sugere a adoção das seguintes práticas ágeis: programação por ciclos, com cada ciclo adicionando novas características ao *software* e com duração não superior a duas semanas; criação de um *backlog*, ou registro de cada iteração do projeto; simplicidade dos artefatos produzidos, que devem relacionar-se a uma necessidade explícita do negócio, incluindo documentação, que pode ser derivada da documentação do próprio código-fonte; programação em ambiente de desenvolvimento integrado, seguindo boas práticas de escrita do código, realizando refatoração imediatamente quando necessário e utilizando testes para verificar o comportamento das unidades do *software*.

5. Modelagem do problema, implementação e discussão dos resultados

Duas principais modelagens foram testadas durante o desenvolvimento do protótipo. A

primeira é representada pela Figura 1 e daqui por diante referenciada como *modelagem naïve*. Nela propõe-se que tuplas professor-disciplina (homens), previamente definidas pelo usuário, proponham casamento a tuplas professor-dia-horário (mulheres). As listas de preferência de um professor-disciplina, neste caso, são todos os professor-dia-horário de mesmo professor, ordenados por dia e horário. Professor-dia-horários, por sua vez, tem em sua lista de preferências disciplinas ministradas por aquele professor, ordenadas aleatoriamente. Uma tabela de horários, nesta modelagem, é representada por um conjunto de casamentos de professor-disciplina com tantos professor-dia-horário, de mesmo professor, quanto satisfaçam o número de créditos da disciplina.

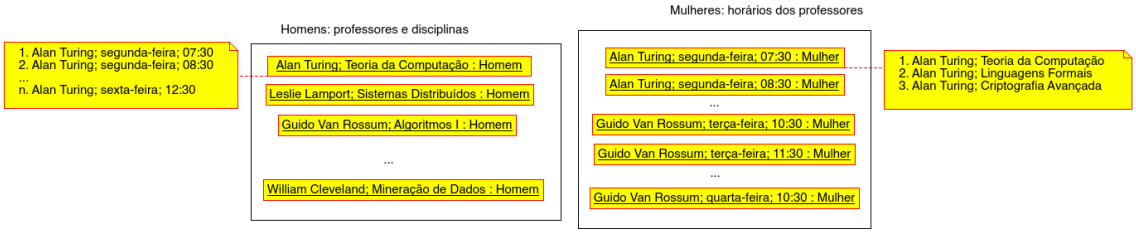

Figura 1: Modelagem naïve

Na segunda modelagem realizada, tuplas professor-disciplina (homens) previamente definidas são selecionadas em grupos, propondo casamento a tuplas professor-dia-horário livre (mulheres), sendo o objetivo realizar um casamento relativo a cada semestre até esgotar-se o número de semestres do curso, e eliminando-se da lista de horários disponíveis dos professores aqueles casados em casamentos anteriores. Esta modelagem será referenciada como *modelagem de muitos casamentos* e pode ser intuída na Figura 2.

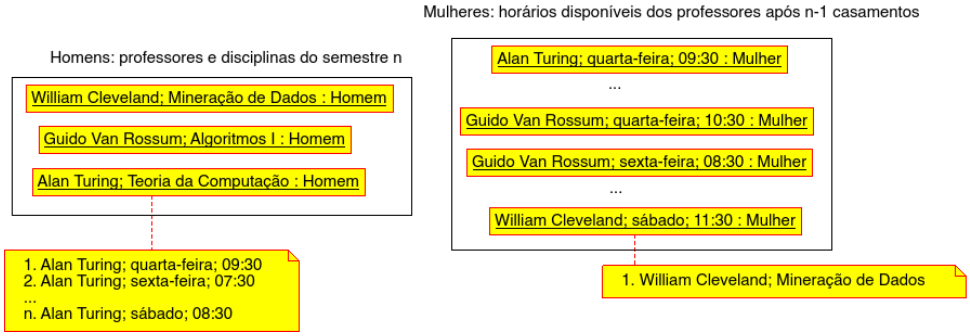

Figura 2: Modelagem de muitos casamentos

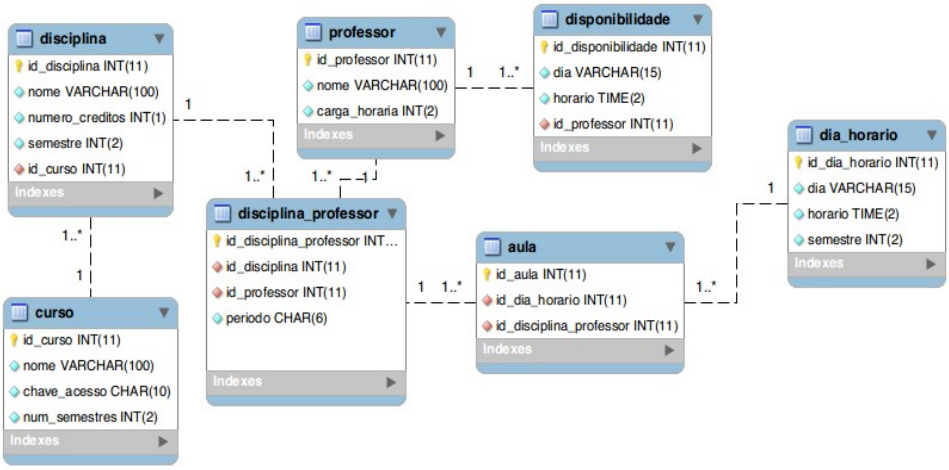

Figura 3: Modelo de dados implementado no protótipo desenvolvido

Buscando dar suporte ao protótipo desejado, foi desenvolvido e implementado o modelo de dados da Figura 3. Neste modelo estão representados os elementos do problema genérico a ser resolvido. O mesmo não se encontra na terceira forma normal [Kent 1983] de forma a preservar a simplicidade do projeto, dado que o objetivo principal estabelecido aqui é testar a aplicabilidade do algoritmo de aceitação diferida ao problema de *timetabling*, não desenvolver um sistema de gestão.

A rotina de aceitação diferida implementada pode ser visualizada na Figura 4, e seu fluxo de forma macro na Figura 5. A mesma é invocada sobre uma lista de homens e mulheres, cada objeto contendo suas listas de preferência, e implementa o algoritmo de Gale e Shapley em uma variante poligâmica para os homens, conservando todas as suas propriedades.

```
public void casar() {
    do {
        contagemPropostas = 0;
        for (Homem h : listaHomens) {
            if (h.casado != -1 && h.proximaProposta < h.listaPreferencias.size()) {
                contagemPropostas++;
                int idMulherCortejada = h.listaPreferencias.get(h.proximaProposta).idLista;
                System.out.println("Homem " + h.eu.getNome() + " , " + h.disciplina.getNome() + " propoe casamento a "
                        + h.listaPreferencias.get(h.proximaProposta).eu.getDia()
                        + h.listaPreferencias.get(h.proximaProposta).eu.getHorario().toString().substring(10, 19));
                boolean resposta = listaMulheres.get(idMulherCortejada).analisarProposta(h);
                if (resposta) {
                    h.esposas.add(h.listaPreferencias.get(h.proximaProposta));
                    if (h.esposas.size() == h.numeroEsposas) {
                        h.casado = -1;
                    } else {
                        h.casado = 1;
                    }
                    if (listaMulheres.get(idMulherCortejada).mudouIdeia) {
                        Homem homemRejeitado = listaHomens.get(listaMulheres.get(idMulherCortejada).idMaridoAnterior);
                        homemRejeitado.esposas.remove(h.listaPreferencias.get(h.proximaProposta));
                        System.out.println(homemRejeitado.eu.getNome() + " , " +
                                homemRejeitado.disciplina.getNome() + " foi rejeitado.");
                        if (homemRejeitado.casado == -1) {
                            if (homemRejeitado.numeroEsposas == 1) {
                                homemRejeitado.casado = 0;
                            } else {
                                homemRejeitado.casado = 1;
                            }
                        }
                    }
                }
                h.proximaProposta++;
            }
        }
    } while (contagemPropostas > 0);
}
```

Figura 4: Rotina de aceitação diferida implementada no protótipo

O desenvolvimento do protótipo passou por seis iterações de duas semanas, partindo da definição de um modelo de dados inicial e construção das classes para solução do problema e terminando em testes e ajustes do algoritmo de aceitação diferida para um caso de teste. As principais dificuldades encontradas durante o desenvolvimento foram a estimativa de esforço e a priorização das tarefas do *backlog*, além do constante refinamento do modelo de dados de forma a atender necessidades descobertas durante as iterações, culminando nas duas principais modelagens comentadas.

Figura 5: Fluxograma do algoritmo de aceitação diferida

A Figura 6 apresenta o horário real, criado por um humano, para o curso de Ciência da Computação do Centro Universitário Franciscano para o segundo semestre de 2015. Esta tabela de horários foi escolhida como caso de teste pois representa um problema pequeno, respeita dentre suas restrições fortes e fracas as mesmas definidas no problema genérico da seção 3.1 e sofre dependências de professores de outros cursos daquela instituição.

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
2º Manhã	1º) 07:40 – 08:30	Física Aplicada à Informática	Matemática Discreta	Física Aplicada à Informática		Língua Brasileira de Sinais
	2º) 08:30 – 09:20	Física Aplicada à Informática	Matemática Discreta	Física Aplicada à Informática		Língua Brasileira de Sinais
	3º) 09:35 – 10:25	Cálculo II	Algoritmos e Programação II	Cálculo II	Algoritmos e Programação II	Matemática Discreta
	4º) 10:25 – 11:15	Cálculo II	Algoritmos e Programação II	Cálculo II	Algoritmos e Programação II	Matemática Discreta
	5º) 11:15 – 12:05	Organização de Computadores			Organização de Computadores	
	6º) 12:05 – 12:55	Organização de Computadores			Organização de Computadores	
Professor		Marcelo Menna Barreto Schwarcke/ Tiago Martins da Silva/ Alessandro Andre Mainardi de Oliveira	Leonardo Dalla Porta/ Ana Paula Canal	Marcelo Menna Barreto Schwarcke/ Tiago Martins da Silva	Ana Paula Canal/ Alessandro Andre Mainardi de Oliveira	Arlei Peripolli/ Leonardo Dalla Porta

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
4º Manhã	1º) 07:40 – 08:30	Probabilidade e Estatística	Pesquisa e Ordenação	Linguagem de Programação II		Sistemas Operacionais
	2º) 08:30 – 09:20	Probabilidade e Estatística	Pesquisa e Ordenação	Linguagem de Programação II		Sistemas Operacionais
	3º) 09:35 – 10:25	Sistemas Digitais	Sistemas Digitais	Sistemas Operacionais	Redes de Computadores	Redes de Computadores
	4º) 10:25 – 11:15	Sistemas Digitais	Sistemas Digitais	Sistemas Operacionais	Redes de Computadores	Redes de Computadores
	5º) 11:15 – 12:05			Probabilidade e Estatística	Pesquisa e Ordenação	Ling. de Programação II
	6º) 12:05 – 12:55			Probabilidade e Estatística	Pesquisa e Ordenação	Ling. de Programação II
Professor		Leandro Ribeiro Fontoura/ Alessandro Andre Mainardi de Oliveira	Alexandre de Oliveira Zamberlan/ Alessandro Andre Mainardi de Oliveira	Reiner Franthesco Perozzo/ Ana Paula Canal/ Leandro Ribeiro Fontoura	Sylvio Andre Garcia Vieira/ Alexandre de Oliveira Zamberlan	Ana Paula Canal/ Sylvio Andre Garcia Vieira / Reiner Franthesco Perozzo

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
6º Manhã	1º) 07:40 – 08:30	Optativa II: Desenvolvimento Web	Metodologia Científica	Requisitos de Software	Compiladores	Computação Gráfica
	2º) 08:30 – 09:20	Optativa II: Desenvolvimento Web	Metodologia Científica	Requisitos de Software	Compiladores	Computação Gráfica
	3º) 09:35 – 10:25	Optativa II: Desenvolvimento Web	Computação Gráfica	Requisitos de Software	Compiladores	Otimização Computacional
	4º) 10:25 – 11:15	Otimização Computacional	Computação Gráfica	Requisitos de Software	Compiladores	Otimização Computacional
	5º) 11:15 – 12:05	Otimização Computacional	Banco de Dados II			Banco de Dados II
	6º) 12:05 – 12:55		Banco de Dados II			Banco de Dados II
Professor		Ricardo Frohlich da Silva/ Mirkos Ortiz Martins	Alessandro Andre Mainardi de Oliveira/ Guilherme Chagas Kurtz/ Gustavo Stangherlin Cantarelli	Fernando Sarturi Prass	Mirkos Ortiz Martins	Guilherme Chagas Kurtz/ Mirkos Ortiz Martins/ Gustavo Stangherlin Cantarelli

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
8º Manhã	1º) 07:40 – 08:30	Tolerância a Falhas	Antropologia e Cosmovisão Franciscana		Empreendedorismo	
	2º) 08:30 – 09:20	Tolerância a Falhas	Antropologia e Cosmovisão Franciscana		Empreendedorismo	
	3º) 09:35 – 10:25	Tolerância a Falhas	Antropologia e Cosmovisão Franciscana		Legislação em Informática	Interfaces Humano Computador
	4º) 10:25 – 11:15		Antropologia e Cosmovisão Franciscana	Optativa IV: Tópicos Avançados em Eletrônica	Legislação em Informática	Interfaces Humano Computador
	5º) 11:15 – 12:05			Optativa IV: Tópicos Avançados em Eletrônica		Interfaces Humano Computador
	6º) 12:05 – 12:55			Optativa IV: Tópicos Avançados em Eletrônica		
Professor		Guilherme Chagas Kurtz	Solanje de Moraes Dejeanne	Alessandro Andre Mainardi de Oliveira	Fernando Sarturi Prass/ Manuela Pereira Savio	Iara Carnevale de Almeida

Figura 6: Horário real do curso de Ciência da Computação para o segundo semestre de 2015

Durante o desenvolvimento experimentou-se duas principais estratégias para definição do número de esposas para as tuplas professor-disciplina. Na primeira, a codificação obedeceu a modelagem *naïve* e o banco de dados foi populado com todos os horários disponíveis dos professores de acordo com a divisão de períodos de aula da instituição. Estes testes demonstraram que a modelagem *naïve* é insuficiente para gerar uma tabela de horários válida, pois quebra a restrição forte 1 ao alocar disciplinas de mesmo semestre no mesmo horário. Foi adotada, então, a modelagem de muitos casamentos para os testes subsequentes.

Neste ponto, satisfez-se todas as restrições fortes, mas observou-se que compor as tuplas professor-horário para cada período de aula da instituição leva a muitas quebras da restrição fraca 4, como pode ser observado na Figura 7: quando da presença de disciplinas com número ímpar de créditos dentro de um semestre, induz-se a alocação de horários órfãos, o que é inconveniente.

```
********** RESULTADOS:
Professor Mirkos Martins , Otimização Computacional casado com:
quinta às  09:30:00 , quinta às  10:30:00 , quinta às  11:30:00 , quinta às  12:30:00 ,
Professor Alessandro André , Metodologia Científica casado com:
quinta às  08:30:00 , sexta às  07:30:00 ,
Professor Guilherme Kurtz , Computação Gráfica casado com:
terça às  12:30:00 , quarta às  11:30:00 , quarta às  12:30:00 , quinta às  07:30:00 ,
Professor Gustavo Cantarelli , Banco II casado com:
terça às  08:30:00 , terça às  09:30:00 , terça às  10:30:00 , terça às  11:30:00 ,
Professor Mirkos Martins , Compiladores casado com:
segunda às  10:30:00 , segunda às  11:30:00 , segunda às  12:30:00 , terça às  07:30:00 ,
Professor Ricardo Frohlich , Optativa II casado com:
segunda às  07:30:00 , segunda às  08:30:00 , segunda às  09:30:00 ,
Professor Fernando Prass , Requisitos de Software casado com:
quarta às  07:30:00 , quarta às  08:30:00 , quarta às  09:30:00 , quarta às  10:30:00 ,
```

Figura 7: Registro de execução do sistema alocando horários para o sexto semestre

Adotou-se então uma nova estratégia de alocação de horários, registrando blocos de dois horários para representar a disponibilidade de cada professor, limitando assim o número de esposas para cada tupla professor-disciplina para um número sempre par. A Figura 8 representa a tabela de horários gerada pelo protótipo codificado obedecendo à modelagem de muitos casamentos com esta estratégia de alocação de créditos. Quando comparada à Figura 6, assume-se que: os mesmos professores seriam alocados às disciplinas ofertadas que no horário real; cada turma tem sala e laboratório próprios para as disciplinas teóricas e práticas; horários de professores oriundos de outros cursos são desconsiderados na contagem de restrições fracas. Dirimidas estas questões, observa-se que não houve quebra de restrições fortes, e apenas quatro quebras de restrições fracas, concluindo-se que o algoritmo de aceitação diferida implementado resolve o problema de *timetabling* definido.

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
2º Manhã	1º) 07:40 – 08:30	Física Aplicada à Informática	Matemática Discreta	Física Aplicada à Informática		Língua Brasileira de Sinais
	2º) 08:30 – 09:20	Física Aplicada à Informática	Matemática Discreta	Física Aplicada à Informática		Língua Brasileira de Sinais
	3º) 09:35 – 10:25	Cálculo II	Algoritmos e Programação II	Cálculo II		Matemática Discreta
	4º) 10:25 – 11:15	Cálculo II	Algoritmos e Programação II	Cálculo II		Matemática Discreta
	5º) 11:15 – 12:05	Algoritmos e Programação II	Organização de Computadores	Organização de Computadores		
	6º) 12:05 – 12:55	Algoritmos e Programação II	Organização de Computadores	Organização de Computadores		

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
4º Manhã	1º) 07:40 – 08:30	Probabilidade e Estatística	Sistemas Operacionais	Linguagem de Programação II	Pesquisa e Ordenação	
	2º) 08:30 – 09:20	Probabilidade e Estatística	Sistemas Operacionais	Linguagem de Programação II	Pesquisa e Ordenação	
	3º) 09:35 – 10:25	Redes de Computadores	Linguagem de Programação II	Pesquisa e Ordenação	Sistemas Digitais	
	4º) 10:25 – 11:15	Redes de Computadores	Linguagem de Programação II	Pesquisa e Ordenação	Sistemas Digitais	
	5º) 11:15 – 12:05	Redes de Computadores	Sistemas Operacionais	Probabilidade e Estatística	Sistemas Digitais	
	6º) 12:05 – 12:55	Redes de Computadores	Sistemas Operacionais	Probabilidade e Estatística	Sistemas Digitais	

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
6º Manhã	1º) 07:40 – 08:30	Optativa II: Desenvolvimento Web	Compiladores	Requisitos de Software	Computação Gráfica	Metodologia Científica
	2º) 08:30 – 09:20	Optativa II: Desenvolvimento Web	Compiladores	Requisitos de Software	Computação Gráfica	Metodologia Científica
	3º) 09:35 – 10:25	Optativa II: Desenvolvimento Web	Banco de Dados II	Requisitos de Software	Otimização Computacional	
	4º) 10:25 – 11:15		Banco de Dados II	Requisitos de Software	Otimização Computacional	
	5º) 11:15 – 12:05	Compiladores	Banco de Dados II	Computação Gráfica	Otimização Computacional	
	6º) 12:05 – 12:55	Compiladores	Banco de Dados II	Computação Gráfica	Otimização Computacional	

Semestre/ Turno	Período/ Horário	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
8º Manhã	1º) 07:40 – 08:30	Interfaces Humano Computador	Antropologia e Cosmovisão Franciscana	Optativa IV: Tópicos Avançados em Eletrônica	Empreendedorismo	
	2º) 08:30 – 09:20	Interfaces Humano Computador	Antropologia e Cosmovisão Franciscana		Empreendedorismo	
	3º) 09:35 – 10:25	Interfaces Humano Computador	Antropologia e Cosmovisão Franciscana	Tolerância a Falhas	Legislação em Informática	
	4º) 10:25 – 11:15		Antropologia e Cosmovisão Franciscana		Legislação em Informática	
	5º) 11:15 – 12:05	Optativa IV: Tópicos Avançados em Eletrônica	Tolerância a Falhas			
	6º) 12:05 – 12:55	Optativa IV: Tópicos Avançados em Eletrônica	Tolerância a Falhas			

Figura 8: Horário gerado pelo protótipo. Apenas as restrições fracas 2 e 4 são quebradas.

Destacam-se as seguintes particularidades:

- A implementação de listas de preferência sorteadas por horário para os professores tende a gerar horários adjacentes, fazendo cumprir a restrição fraca 1 e resultando em dias de aula livres para os alunos (observar o quarto semestre da Figura 8 em relação ao horário real na Figura 6);
- O principal problema levantado pela modelagem de muitos casamentos foi a divisão dos horários para disciplinas de número ímpar de créditos, dado que estas geraram todas as quebras de restrição fraca do problema. Pode ser demonstrado que a introdução de muitas disciplinas desta natureza aumenta as quebras destas restrições, podendo levar a uma contradição na geração de uma tabela de horários em relação à carga horária dos professores;

- A composição das tuplas professor-horário para a modelagem de muitos casamentos pode ser adotada tanto em termos de períodos de aula como blocos de horários, sendo necessária portanto uma revisão das restrições fracas para definir se um horário de aula representa um ou dois períodos reais, um problema intimamente relacionado com a questão do ponto anterior.

6. Conclusões

O problema dos casais estáveis é satisfatoriamente resolvido com o uso do algoritmo de aceitação diferida ou uma variação do mesmo. Embora trate-se de um problema de formulação simples, seu modelo é replicado em diversas áreas da vida moderna que exijam combinação estável entre entidades. *Timetablin*g é um problema de tal natureza, onde conjuntos de entidades ligadas entre si devem ser, por sua vez, ligadas a diferentes eventos de forma consistente, satisfazendo-se determinadas restrições.

Não havendo uma solução óbvia, amplamente testada e respaldada para este problema, este trabalho modelou, produziu e experimentou o algoritmo de aceitação diferida para solução do mesmo, concluindo que o mesmo pode resolver um problema genérico com poucas quebras de restrições fracas em tempo de execução insignificante em um computador moderno, conforme sugerido em [Prates e Prass 2015]. Das questões não resolvidas pelo problema e modelagem propostos, destacam-se a divisão dos horários por períodos de aula, que no caso teste problematizou disciplinas de número ímpar de créditos causando as únicas quebras de restrição fraca.

Para trabalhos futuros, sugere-se: implementação do algoritmo de aceitação diferida para o casamento de professores e disciplinas previamente à modelagem apresentada neste trabalho; investigação e testes da complexidade da solução quando gerando horários para diversos cursos que compartilham professores (sugere-se tentar solucionar com a utilização de uma árvore de dependências); revisão da definição do problema ou da modelagem da técnica de forma a corrigir, ou desenho de um algoritmo próprio para corrigir as quebras de restrição fraca encontradas.

Bibliografia

Alcalde, J. and Barbera, S. (1994). Top dominance and the possibility of strategy-proof stable solutions to matching problems. *Economic Theory*, v. 4, p. 417–435.

Alhakami, H., Chen, F. and Janicke, H. (2014). An Extended Stable Marriage Problem Algorithm for Clone Detection. *International Journal of Software Engineering and Applications (IJSEA)*, v. 5, n. 4, p. 103–122.

Amorim, G. S. (2012). Proposta de Implementação de uma Ferramenta para Solucionar Problemas de Timetable da UNIFRA. Centro Universitário Franciscano.

Bai, R., Li, J., Atkin, J. A. and Kendall, G. (2013). A novel approach to independent taxi scheduling problem based on stable matching. *Journal of the Operational Research Society*, v. 65, n. 10, p. 1–10.

Bellio, R., Ceschia, S., Gaspero, L. Di, Schaerf, A. and Urli, T. (2014). Feature-based

tuning of simulated annealing applied to the curriculum-based course timetabling problem.

Biró, P. (2007). The stable matching problem and its generalizations : an algorithmic and game theoretical approach. Budapesti Műszaki és Gazdaságtudományi Egyetem.

Cechlárová, K., Fleiner, T., Manlove, D. F. and Mcbride, I. (2015). Stable matchings of teachers to schools.

Ceschia, S., Di Gaspero, L. and Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, v. 39, p. 1615–1624.

Cook, S. (2000). The P versus NP problem. *The Clay Mathematical Institute; The Millennium Prize Problem*, p. 1–12.

Denton, J. W. and Peace, a G. (2003). Selection and Use of MySQL in a Database Management Course. *Journal of Information Systems Education*, v. 14, n. 4, p. 401–407.

Farczadi, L., Georgiou, K. and Jochen, K. (2014). Stable marriage with general preferences. . http://arxiv.org/abs/1407.1853v2.

Farooq, M. S., Khan, S. A., Ahmad, F., Islam, S. and Abid, A. (2014). An evaluation framework and comparative analysis of the widely used first programming languages. *PLoS ONE*, v. 9, n. 2, p. 1–25.

Gale and Shapley (1962). College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, v. 69, n. 1, p. 9–15.

Gervás, P. and Miguel, B. S. (1999). Sequential Building of Constrained Timetables Using Rule-Based Heuristics : An Expert System for Automated Timetabling at UEM.

Glover, F. and Kochenberger, G. A. [Eds.] (2003). *Handbook of Metaheuristics*. New York, N.Y.: Kluwer Academic Publishers.

Goldreich, O. (2008). *Computational Complexity: a Conceptual Perspective*. New York, N.Y.: Cambridge University Press.

Gröbner, M., Wilke, P. and Büttcher, S. (2003). A Standard Framework for Timetabling Problems. *Practice and Theory of Automated Timetabling*. Berlin: Springer-Verlag. p. 24–38.

Hasan, M. and Hossain, E. (2015). Distributed Resource Allocation in 5G Cellular Networks. In: Vanithamby, R.; Telwar, S.[Eds.]. . *Towards 5G: Applications, Requirements and Candidate Technologies*. Wiley. p. 1–26.

Hollar, A. B. (2006). Cowboy: An Agile Programming Methodology for a Solo Programmer. Virginia Commonwealth University.

Irving, R. W. (2008). Stable matching problems with exchange restrictions. *Journal of Combinatorial Optimization*, v. 16, p. 344–360.

Iwama, K. and Miyazaki, S. (2008). A survey of the stable marriage problem and its variants. *Proceedings - International Conference on Informatics Education and Research for Knowledge-Circulating Society, ICKS 2008*, n. i, p. 131–136.

Kent, W. (1983). A simple guide to five normal forms in relational database theory. *Communications of the ACM*, v. 26, n. 2, p. 120–125.

Kristiansen, S. and Stidsen, T. R. (2013). A Comprehensive Study of Educational Timetabling - a Survey. *Department of Management Engineering, Technical University of Denmark*, n. November.

Liu, J. and Chiu, D. M. (2010). Reciprocating Preferences Stablize Matching: College Admissions Revisited.

Moreira, J. J. (2008). A system for automatic construction of exam timetable using genetic algorithms. *Tékhne-Revista de Estudos Politécnicos*, v. VI, p. 1–12.

Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research*, v. 218, n. 1, p. 261–293.

Prates, E. and Prass, F. S. (2015). Proposta de aplicação do problema dos casais estáveis à automação de horários (timetabling). In *Anais do XIII Simpósio de Informática do Centro Universitário Franciscano*.

Ronn, E. (1990). NP-complete stable matching problems. *Journal of Algorithms*, v. 11, p. 285–304.

Roth, A. E. (2008). Deferred acceptance algorithms: History, theory, practice, and open questions. *International Journal of Game Theory*, v. 36, p. 537–569.

Schmidt, G. and Strohlein, T. (1980). Timetable construction - an annotated bibliography. *The Computer Journal*, v. 23, n. 4, p. 307–316.

Smolira, M., Cytawa, J. and Michalak, L. (2003). Evolutionary algorithms for timetable problems. v. 1, p. 245–250.

Willemen, R. (2002). *School timetable construction; algorithms and complexity*. Eindhoven: Technische Universiteit Eindhoven.