Atividade Extra 1 - Stable Matching

Felipe Campolina Soares de Paula[1]

[1]ICEI - Pontifícia Universidade Católica de Minas Gerais

1. Introdução

O problema do *Stable Matching* (ou *Casamento Estável*) é uma questão clássica em teoria dos jogos e otimização, com amplas aplicações em diversos campos, incluindo economia, informática e teoria das escolhas [Gale and Shapley 1962]. Este problema visa encontrar uma correspondência estável entre dois conjuntos de elementos, onde uma correspondência é considerada estável se não houver dois elementos que prefeririam estar emparelhados um com o outro do que com seus parceiros atuais.

Mais formalmente, dado dois conjuntos A e B de tamanho n, e dadas as preferências de cada elemento de A e B sobre os elementos do outro conjunto, o objetivo é encontrar uma correspondência $M \subseteq A \times B$ tal que cada elemento de A e de B é emparelhado exatamente uma vez, e a correspondência é estável.

Uma correspondência M é estável se não existir um par $(a, b) \notin M$ tal que a prefere b a seu parceiro atual em M e b prefere a a seu parceiro atual em M. Em termos matemáticos, para todo $(a, b) \notin M$:

$$\text{Se } a \text{ prefere } b \text{ a } M(a) \text{ e } b \text{ prefere } a \text{ a } M(b), \text{ então } M \text{ é instável.}$$

Para resolver este problema, David Gale e Lloyd Shapley propuseram em 1962 o algoritmo conhecido como *Gale-Shapley algorithm* ou algoritmo dos *Proposals and Rejections*. Este algoritmo garante a existência de uma correspondência estável para qualquer conjunto de preferências. O processo inicia com todos os elementos de um conjunto (por exemplo, A) propondo-se aos seus parceiros preferidos no outro conjunto (B). Cada elemento de B avalia as propostas recebidas, aceita a melhor e rejeita as demais. Os elementos rejeitados de A continuam propondo-se a suas próximas preferências até que todos estejam emparelhados de forma estável.

A estrutura deste texto está organizada da seguinte forma: na Seção 2, discutimos em detalhes o princípio do *Stable Matching* e conceitos fundamentais para essa discussão. Na Seção 3, fornecemos um exemplo ilustrativo do funcionamento do algoritmo de Gale-Shapley. Na Seção 4, exploramos algumas das principais aplicações práticas deste algoritmo. Finalmente, na Seção 5, analisamos o custo computacional do algoritmo e discutimos suas implicações para problemas de grande escala.

2. Princípios e Conceitos

Para formular a questão, podemos imaginar um grupo de estudantes de ciências da computação que estão se candidatando a estágios de verão em várias empresas. Cada candidato tem uma ordem de preferência em relação às empresas, e cada empresa, após receber as candidaturas, forma uma ordem de preferência em relação aos candidatos.

Gale e Shapley consideraram os problemas que poderiam surgir nesse processo na ausência de um mecanismo que mantivesse o status quo. Por exemplo, se um candidato

aceitar uma oferta de uma empresa, mas depois receber uma oferta de uma empresa que prefere mais, ele pode rescindir a primeira oferta, causando um efeito dominó de rejeições e novas ofertas.

Para resolver isso, Gale e Shapley desenvolveram o algoritmo dos *Proposals and Rejections*, que funciona da seguinte maneira:

1. **Inicialização**: Todos os elementos de um conjunto (por exemplo, A) começam sem pares.
2. **Proposta**: Cada elemento de A tenta formar um parte ao elemento mais preferido de B que ainda não o rejeitou.
3. **Avaliação**: Cada elemento de B considera todas as propostas recebidas e mantém a melhor proposta (de acordo com suas preferências), rejeitando as outras.
4. **Iteração**: Os elementos rejeitados de A continuam propondo-se a seus próximos preferidos em B até que todos os elementos estejam emparelhados de forma estável.

O algoritmo garante que uma correspondência estável será sempre encontrada, independentemente das preferências individuais dos participantes. A estabilidade aqui significa que não haverá dois elementos que prefeririam estar juntos do que com seus parceiros atuais, evitando assim qualquer incentivo para mudanças nos pares formados.

2.1. Conceitos Fundamentais

1. **Casamento Perfeito (Perfect Matching)**: Um casamento perfeito ocorre quando todos os participantes estão emparelhados monogamicamente. Em termos mais específicos, cada homem é emparelhado com exatamente uma mulher e cada mulher é emparelhada com exatamente um homem. No contexto dos estudantes de Ciência da Computação e estágios, isso significa que cada estudante recebe exatamente uma oferta de estágio e cada empresa contrata exatamente um estagiário.
2. **Estabilidade (Stability)**: Estabilidade significa que não há incentivo para que algum par de participantes interfira no emparelhamento atual agindo conjuntamente. Formalmente, em um emparelhamento M, um par não emparelhado (m, w) é instável se o homem m e a mulher w preferem estar um com o outro do que com seus parceiros atuais. Por exemplo, se um estudante preferir uma empresa àquela para a qual foi designado e essa empresa também preferir esse estudante ao seu estagiário atual, eles têm um incentivo para formar um novo par, o que torna a situação instável.
3. **Casamento Estável (Stable Matching)**: Um casamento estável é um casamento perfeito onde não há pares instáveis. Isso garante que nenhum estudante e nenhuma empresa prefeririam estar emparelhados um com o outro em vez de com seus parceiros designados.
4. **Problema do Casamento Estável (Stable Matching Problem)**: Dado um conjunto de listas de preferências de n homens e n mulheres, ou, no contexto dos estágios, n estudantes e n empresas, o objetivo é encontrar um casamento estável se ele existir. Isso significa encontrar uma maneira de emparelhar todos os estudantes com empresas de forma que nenhum estudante e nenhuma empresa prefiram mudar sua correspondência.

Estes conceitos fundamentais são essenciais para entender o problema do *Stable Matching* e suas aplicações práticas. Eles ajudam a garantir que o processo de emparelhamento seja auto-regulável e estável, evitando caos e insatisfação entre os participantes.

3. Funcionamento e Exemplo

O problema do *Stable Matching* tem várias aplicações práticas em diferentes contextos. Uma aplicação interessante é no processo de alocação de alunos de Ciência da Computação da PUC Minas a orientadores de TCC. Neste cenário, cada aluno tem suas próprias preferências em relação aos professores, e cada professor tem suas próprias especialidades e limites de orientação.

3.1. Exemplo Fictício: Alocação de Alunos de Ciência da Computação a Orientadores de TCC

Imagine um grupo de alunos de Ciência da Computação da PUC Minas que precisam escolher orientadores de TCC entre um grupo de professores, como Max, Silvio, Wallison, Henrique, entre outros. Cada aluno tem interesses específicos em áreas como Inteligência Artificial, Redes de Computadores, Sistemas Distribuídos e Desenvolvimento de Software. Da mesma forma, cada professor tem especialidades nessas áreas e pode orientar um número limitado de alunos.

Cada aluno cria uma lista de preferências com base nas áreas de especialidade dos professores e na sua afinidade pessoal. Por exemplo, o aluno A1 pode preferir o professor Max em primeiro lugar, seguido pelo professor Henrique, enquanto o aluno A2 pode preferir o professor Silvio e depois o professor Wallison.

Os professores, por sua vez, também criam listas de preferências com base nos interesses dos alunos e no potencial de pesquisa que cada aluno apresenta. Por exemplo, o professor Max pode preferir orientar o aluno A1, seguido pelo aluno A3, enquanto o professor Henrique pode preferir o aluno A2 e depois o aluno A4.

3.2. Funcionamento

Utilizando o algoritmo de Gale-Shapley, o processo de alocação segue os seguintes passos:

1. **Inicialização**: Todos os alunos e professores começam sem par.

2. **Proposta**: Cada aluno propõe-se ao professor mais preferido de sua lista que ainda não o rejeitou.

3. **Avaliação**: Cada professor considera todas as propostas recebidas e mantém as melhores propostas (de acordo com suas preferências e limites de orientação), rejeitando as outras.

4. **Iteração**: Os alunos rejeitados continuam propondo-se aos próximos preferidos em suas listas até que todos os alunos estejam emparelhados de forma estável.

Preferências de Alunos

	1ª Preferência	2ª Preferência	3ª Preferência
A1	Max	Henrique	Silvio
A2	Silvio	Wallison	Henrique
A3	Max	Wallison	Henrique
A4	Henrique	Max	Silvio

Preferências de Professores

	1ª Preferência	2ª Preferência	3ª Preferência
Max	A1	A3	A4
Silvio	A2	A1	
Wallison	A3	A2	
Henrique	A4	A2	A1

O algoritmo inicia com todos os alunos sem orientação. Cada aluno propõe-se ao seu professor mais preferido:

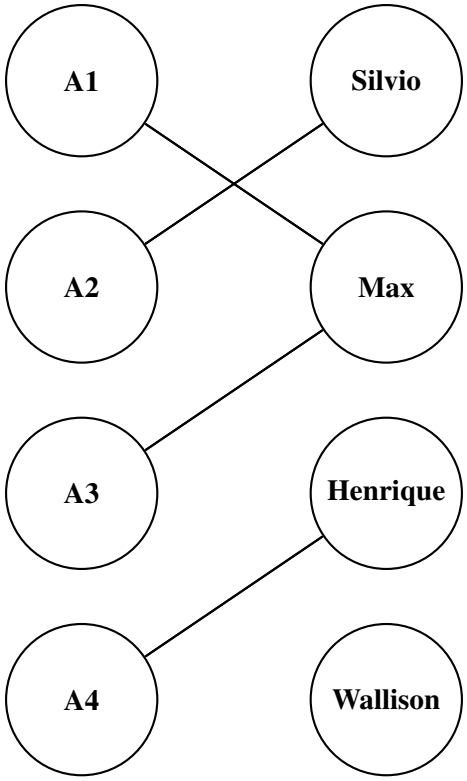

Figure 1. Propostas Iniciais

Assim , o Max, recebendo propostas de A1 e A3, prefere manter A1 e rejeita A3. Silvio aceita A2. Henrique aceita A4. A3, rejeitado por Max, propõe-se a Wallison, que aceita A3. Ficando igual a figura abaixo:

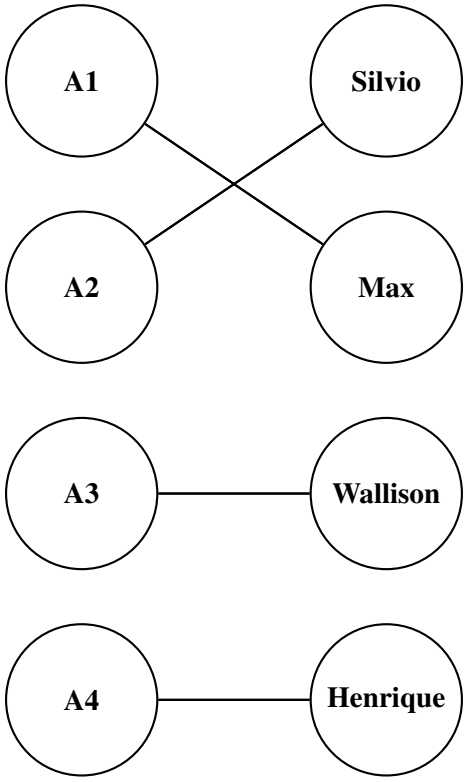

Figure 2. Após Algortímo

Este emparelhamento é estável porque nenhum aluno e professor preferem estar emparelhados com outra pessoa ao invés do seu par atual.

Este exemplo ilustra como o algoritmo de Gale-Shapley pode ser utilizado para resolver problemas de alocação de recursos de maneira estável e eficiente, garantindo que as preferências de todos os participantes sejam consideradas.

3.3. Algoritmo de Alocação de Alunos a Orientadores de TCC

Segue abaixo o pseudo código do exemplo:

Entrada: Lista de preferências dos alunos, Lista de preferências dos professores

Saída: Pareamentos entre alunos e orientadores de TCC

Algorithm 1 Alocação de Alunos a Orientadores de TCC

```
Inicialize todos os alunos e professores como livres.
while existir um aluno livre A que ainda não propôs a todos os seus professores preferidos do
    Selecione o professor P que é a próxima preferência de A e ainda não foi proposto por A.
    if P estiver livre then
        A e P tornam-se pareados.
    else
        Se P já estiver pareado com outro aluno A'
        if P prefere A a A' then
            A e P tornam-se pareados.
            A' torna-se livre.
        else
            A' e P permanecem pareados.
        end if
    end if
end while
```

Este algoritmo recebe como entrada as listas de preferências dos alunos e dos professores e retorna os pareamentos entre os alunos e os orientadores de TCC. Ele assegura que cada aluno seja pareado com um orientador de acordo com suas preferências, considerando também as preferências dos orientadores.

4. Aplicações

O problema do *Stable Matching* tem várias aplicações práticas em diferentes contextos. Abaixo são apresentados alguns exemplos conhecidos na literatura:

4.1. National Resident Matching Program (NRMP)

Uma das aplicações mais notáveis do algoritmo de Gale-Shapley é o National Resident Matching Program (NRMP) nos Estados Unidos, que emparelha médicos residentes com hospitais. Este programa utiliza um algoritmo baseado em *Stable Matching* para garantir que os médicos e os hospitais sejam emparelhados de maneira estável, evitando conflitos e insatisfações [Roth 1990].

4.2. Alocação de Estudantes a Escolas

Outra aplicação importante do *Stable Matching* é a alocação de estudantes a escolas. Em várias cidades ao redor do mundo, os sistemas escolares utilizam algoritmos de *Stable Matching* para emparelhar estudantes com escolas com base nas preferências dos estudantes e nas prioridades das escolas. Este método ajuda a garantir que o processo de alocação seja justo e eficiente [Abdulkadiroğlu and Sönmez 2003].

4.3. Mercados de Trabalho

Alguns mercados de trabalho também se beneficiam do algoritmo de Gale-Shapley. Em certos setores, as empresas e os candidatos a emprego são emparelhados utilizando princípios de *Stable Matching* para assegurar que os candidatos sejam colocados em empresas que correspondam às suas preferências e qualificações, e vice-versa [Hatfield and Milgrom 2005].

4.4. Doação de Órgãos

O *Stable Matching* também é utilizado em sistemas de doação de órgãos. Algoritmos de correspondência estável ajudam a emparelhar doadores de órgãos com receptores de forma eficiente e justa, levando em consideração fatores como compatibilidade médica e urgência do caso [Roth et al. 2005].

4.5. Sistemas de Recomendação

Em sistemas de recomendação, como os utilizados por plataformas de streaming e de e-commerce, o *Stable Matching* pode ser usado para emparelhar usuários com conteúdos ou produtos que atendam melhor às suas preferências. Este tipo de aplicação melhora a experiência do usuário, oferecendo recomendações personalizadas [Deshpande and Karypis 2004].

Estes exemplos ilustram a versatilidade e a importância do *Stable Matching* em uma ampla gama de aplicações práticas, demonstrando como os princípios teóricos podem ser aplicados para resolver problemas reais de maneira eficiente e justa.

5. Custo e Complexidade Computacional

O algoritmo de Gale-Shapley, utilizado para resolver o problema de *Stable Matching*, é notável não apenas pela sua eficácia, mas também pela sua eficiência computacional. A seguir, discutimos a complexidade do algoritmo e algumas de suas variantes, bem como seus custos computacionais.

5.1. Complexidade do Algoritmo de Gale-Shapley

O algoritmo original de Gale-Shapley tem uma complexidade de tempo de $O(n^2)$, onde n é o número de elementos em cada conjunto. Esta complexidade surge do fato de que, no pior caso, cada participante pode fazer uma proposta a cada outro participante uma vez, resultando em $n \times n = n^2$ operações [Gale and Shapley 1962]. A complexidade de espaço do algoritmo também é $O(n^2)$, pois é necessário armazenar as listas de preferências e o estado das propostas.

5.2. Otimizações e Variantes do Algoritmo

Várias otimizações e variantes do algoritmo de Gale-Shapley foram propostas para lidar com situações específicas e melhorar a eficiência em certos contextos:

1. **Matching com Restrições**: Em certos casos, como no emparelhamento de doadores e receptores de órgãos, é necessário considerar restrições adicionais (como compatibilidade médica). Algoritmos especializados, como o de Roth et al. (2005), foram desenvolvidos para lidar com essas restrições, mantendo a complexidade computacional administrável [Roth et al. 2005]. Esses algoritmos frequentemente utilizam técnicas de programação inteira e combinatória para incorporar restrições adicionais.

2. **Emparelhamento com Contratos**: Quando os emparelhamentos envolvem contratos com múltiplos atributos, a complexidade pode aumentar. Hatfield e Milgrom (2005) propuseram um algoritmo eficiente para o emparelhamento com contratos, que lida com contratos multi-atributo e mantém a estabilidade. A complexidade pode aumentar para $O(n^3)$ em alguns casos devido à necessidade de considerar as diferentes combinações de contratos possíveis [Hatfield and Milgrom 2005]. Este algoritmo expande o modelo tradicional de *Stable Matching* para incluir dimensões adicionais de decisão.
3. **Algoritmos Aproximados**: Para problemas de grande escala, onde a complexidade quadrática pode ser impraticável, algoritmos aproximados que encontram emparelhamentos quase estáveis em tempo subquadrático foram desenvolvidos. Por exemplo, Deshpande e Karypis (2004) desenvolveram algoritmos baseados em técnicas de aprendizado de máquina e heurísticas que podem encontrar soluções aproximadas mais rapidamente [Deshpande and Karypis 2004]. Esses algoritmos são úteis em aplicações como sistemas de recomendação, onde a velocidade de resposta é crucial.

References

Abdulkadiroğlu, A. and Sönmez, T. (2003). School choice: A mechanism design approach. *American Economic Review*, 93(3):729–747.

Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177.

Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15.

Hatfield, J. W. and Milgrom, P. R. (2005). Matching with contracts. *American Economic Review*, 95(4):913–935.

Roth, A. E. (1990). New physicians: A natural experiment in market organization. *Science*, 250(4987):1524–1528.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2005). Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188.