Uso dos Algoritmos Simulated Annealing e Random Search no Problema de 3-Satisfatibilidade Booliana (3-SAT)

Ediana da Silva de Souza[1], **Êrica Peters do Carmo**[1]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brasil

{edianadasilvadesouza,ericapetersc}@gmail.com

Resumo. *O uso de algoritmo de busca tem sido amplamente utilizado em diferentes áreas, visto que através destes é possível reduzir o custo de recurso e de tempo para solucionar um problema. Isto é ainda mais oportuno para problemas que possuem grandes espaços de busca e são muito complexos para serem resolvidos de forma rotineira. Desta forma, este artigo tem como objetivo descrever o comportamento dos algoritmos Simulated Annealing e Random Search quando aplicados no problema 3-SAT. O estudo da literatura realizado indica que o comportamento do Simulated Annealing se mostra mais eficiente que o do Random Search por possuir uma maior robustez. Como resultado deste trabalho identifica-se que os algoritmos obtiveram resultados semelhantes quando analisado as cláusulas do problema 3-SAT. Porém, quando estudado critérios mais profundos dos algoritmos, como variância, desvio padrão e convergência, percebe-se que o Simulated Annealing obteve um desempenho mais significativo. Além disso, as autoras deste acreditam que este último seja um algoritmo mais confiável devido à sua robustez.*

1. Introdução

A resolução de problemas por meio dos algoritmos de busca é dado como uma importante técnica da área de Inteligência Artificial [Udemy 2018]. Uma das classes deste tipo de algoritmo são as buscas heurísticas, na qual utiliza informações prévias sobre o problema a ser resolvido para chegar ao melhor resultado possível. Neste sentido têm-se o Simulated Annealing, pertencente também a classe de algoritmos de trajetória, possui como objetivo ir melhorando a solução inicial para o problema em que foi aplicado através de um comportamento inspirado na natureza [Tomazi et al. 2016]. Já o algoritmo Random Search é um algoritmo randômico, onde as soluções são todas dadas de maneira aleatória [Schumer and Steiglitz 1968]. Dado este escopo, o modelo de problema abordado neste trabalho descreve a implementação destes dois algoritmos como solucionador de instâncias do 3-SAT.

1.1. Justificativa e Motivação

A complexidade de um problema está diretamente relacionada ao tamanho do seu espaço de busca correspondente [Direne 2018]. Dado este cenário, este trabalho justifica-se pela busca de resultados eficientes através da aplicabilidade dos algoritmos Simulated Annealing e Random Search no problema 3-SAT, sendo este considerado um problema complexo e com grande espaço de busca. Logo, a utilização destes algoritmos visam reduzir este espaço.

1.2. Objetivo

O objetivo deste trabalho foi encontrar soluções eficientes para problemas com grandes espaços de busca, como representado pelo 3-SAT. Para isso, implementou-se os algoritmos Simulated Annealing e Random Search e testou-os com três instâncias do problema, onde estas continham diferentes números de variáveis e cláusulas, sendo cada uma delas testadas dez vezes pelos algoritmos. Posteriormente, verificou-se qual dos algoritmos possuía o melhor comportamento através dos gráficos de convergência e desvio padrão.

1.3. Organização do Texto

Este trabalho está estruturado da seguinte forma: a seção 2 conceitua o problema 3-SAT e caracteriza os algoritmos Simulated Annealing e Random Search. A seção 3 apresenta os métodos e justificativas de desenvolvimento do algoritmo implementado. A seção 4 fornece descrições sobre os experimentos realizados no ambiente e os resultados alcançados. A seção 5 estuda os resultados obtidos. Finalmente, conclusões e direções de pesquisas futuras são apresentadas na seção 6.

2. Fundamentação Teórica

Nesta seção será conceituado o problema 3-SAT, bem como os elementos envolvidos para a solução de suas instâncias, dado pelos algoritmos Simulated Annealing e Random Search

2.1. 3-SAT

O primeiro problema identificado como pertencente à classe NP-Completo foi o SAT, problema de Satisfatibilidade Booleana [Sipser 2005]. Por ele estar nesta classe, significa que ainda não há um algoritmo determinístico que o resolva em tempo polinomial, isto porque o problema consiste em verificar se há uma atribuição de valores para as variáveis de uma fórmula lógica que tornem esta fórmula verdadeira, isto é, resulte em uma tautologia [Lasma 2010]. Dentro da classe dos problemas SAT, existe o problema 3-SAT. Ele é definido por três variáveis separadas pelo símbolo OU - formando cláusulas - onde cada cláusula é unida pelo símbolo E [Tovey 1984].

Como mencionado, neste trabalho as instâncias a serem resolvidas possuem tamanhos diferentes. De acordo com [Lasma 2010], o problema 3-SAT possui fases de transição, onde separa o problema em três tipos: os problemas que são facilmente resolvidos, pois são compostos por poucas cláusulas e poucos símbolos, os que são resolvidos mas demandam maior custo computacional e os problemas que são resolvidos em tempo exponencial.

2.2. Algoritmo Simulated Annealing

Simulated Annealing é um método para encontrar soluções satisfatórias para problemas de otimização difíceis [de Araujo 2001]. Foi proposto por Kirkpatrick, Gelatt e Vecchi em 1983 [Tomazi et al. 2016] e por possui um comportamento estocástico, facilitando uma análise teórica de sua convergência assintótica, tornou-se muito popular para os matemáticos [Aarts et al. 2005]. O Simulated Annealing foi baseado na ideia de moldagem de metais e vidros, onde são aquecidos a uma temperatura elevada e em seguida resfriados lentamente, ficando menos flexíveis conforme a temperatura vai diminuindo. Isto é, no

contexto do algoritmo, o processo de otimização é realizado por iterações, simulando os níveis de temperatura no resfriamento [Haeser and Ruggiero 2008].

O algoritmo possui o seguinte funcionamento: inicialmente uma solução aleatória é gerada e testada e, em cada iteração, o algoritmo pode substituir a solução atual por uma solução contida na vizinhança. Se a solução da vizinhança acarretar em um resultado melhor que o anterior, a solução é adotada como solução atual, se não, é escolhido de maneira probabilística e com base na temperatura se a solução deve ou não ser descartada. Quanto maior for a temperatura, maior a probabilidade de a solução atual ser substituída pela solução contida na vizinhança. A fórmula descrita está sendo representada abaixo, onde delta é a diferença entre a solução atual e a solução contida na vizinhança.

$$p = e^{\frac{-delta}{temperatura}}$$

À medida que o algoritmo progride, o valor da temperatura vai diminuindo, a probabilidade de mudança de solução consequentemente vai reduzindo e o algoritmo converge para uma solução final [Haeser and Ruggiero 2008]. É importante notar que as iterações podem acontecer mais de uma vez na mesma temperatura, sendo este fator chamado de equilíbrio térmico. Desta forma, conclui-se que a temperatura é o fator determinante neste algoritmo.

Um pseudocódigo para o Simulated Annealing, utilizado como base para a implementação do presente trabalho, é apresentado no Quadro 1.

```
Gerar solução inicial de maneira aleatória.
Testar solução inicial.
Definir número máximo de interações.
Definir temperatura inicial.
Definir número do equilíbrio térmico.
Enquanto interações for menor que número máximo de interações:
    Enquanto equilibro for menor que equilíbrio térmico:
        Gerar solução contida na vizinhança.
        Testar solução contida na vizinhança.
        Se solução contida na vizinhança apresentar melhores resultados que solução atual:
            Solução atual passa a ser a solução contida na vizinhança.
        Se não:
            Calcular probabilidade com base na temperatura.
                Se a probabilidade for alta:
                    Solução atual passa a ser a solução contida na vizinhança.
                Se não:
                    Ignorar.
    Diminuir temperatura.
    Aumentar número de interações.
    Zerar valor do equilíbrio.
Retornar solução final.
```

Quadro 1. Pseudocódigo Simulated Annealing. Fonte: as autoras, 2019.

2.3. Algoritmo Random Search

Random Search é uma estratégia básica de pesquisa aleatória, onde seus resultados fornecem pequenas melhorias [Browlee 2015]. Isto acontece porque, neste método, a solução

testada em cada iteração do algoritmo é independente das soluções anteriores, ou seja, nas as leva em consideração, gerando soluções aleatórias em cada iteração [Browlee 2015]. Um pseudocódigo para o Random Search, utilizado como base para a implementação do presente trabalho, é apresentado no Quadro 2.

```
Gerar solução inicial de maneira aleatória.
Testar solução inicial.
Definir número máximo de interações.
Enquanto interações for menor que número máximo de interações:
        Gera nova solução aleatória.
        Testar nova solução aleatória.
        Se nova solução aleatória apresentar melhores resultados que solução inicial:
                Solução inicial passa a ser a nova solução aleatória.
        Se não:
                Ignorar.
Retornar solução final.
```

Quadro 2. Pseudocódigo Random Search. Fonte: as autoras, 2019.

3. Metodologia de Desenvolvimento

Conforme descrito na seção 1.2, o objetivo do programa desenvolvido é a implementação dos algoritmos Simulated Annealing e Random Search para encontrarem uma atribuição de variáveis que resolvam instâncias do problema 3-SAT. Assim, essa seção descreve a implementação da solução proposta para esse problema.

O programa (disponível em: `https://bit.ly/2rnhtdc`) foi implementado em Python e utiliza a biblioteca matplotlib para criação dos gráficos de convergência e desvio padrão. O link também disponibiliza os arquivos de entrada.

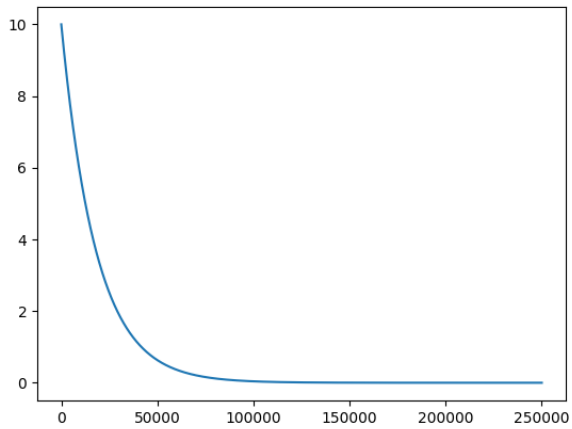


Figura 1. Comportamento da função de resfriamento. Fonte: as autoras, 2019.

Em sua execução, o programa pode lê os arquivos 'arquivo1.txt', 'arquivo2.txt' ou 'arquivo3.txt' do diretório em que está sendo executado. Esses arquivos contém as

instâncias do problema 3-SAT, sendo que os arquivos são composto, respectivamente de: 91 cláusulas e 20 variáveis, 430 cláusulas e 100 variáveis e 1065 cláusulas e 250 variáveis. As cláusulas são expressas por números, por exemplo: *26 -99 7 0*, onde o número 0 representa a conjunção (símbolo E), o sinal negativo representa a negação de uma variável, os números identificam as variáveis e, cada variável está ligada pela disjunção (símbolo OU).

A partir deste arquivo de entrada, o programa executa as funções *simulatedAnnealing* e *randomSearch*. Definiu-se o número de dez execuções, com duzentos e cinquenta mil iterações cada uma, para ambos os algoritmos. No algoritmo Simulated Annealing estabeleceu-se, depois de testes empíricos, o valor do equilíbrio térmico como sete e o valor inicial (máximo) de temperatura como dez. A escolha da função de redução de temperatura (resfriamento) se deu pela busca do padrão de convergência esperado. Utilizou-se a função (1), descrita abaixo, onde *T0* representa o valor inicial da temperatura, *Tn* representa o valor mínimo da temperatura (0.00001) e *i* representa o número de iterações. O comportamento da função pode ser visto na Figura 1:

$$Ti = T0 * (\frac{Tn}{T0})^{\frac{i}{N}} \tag{1}$$

O valor mínimo para a temperatura foi definido somente na fórmula de resfriamento, mas não definiu-se um valor mínimo no algoritmo do Simulated Annealing, pois pela função de resfriamento utilizada, a mesma nunca passará a ter um valor negativo. Além disso, a geração dos vizinhos foi feita da forma que se fizesse uma pertubação na solução atual, que trata-se da mudança do valor de uma variável escolhida aleatoriamente. Se essa essa perturbação acarretar em um resultado melhor que o anterior, ou seja, mais clausulas verdadeiras, a solução perturbada é adotada como solução atual, como explicado na seção 2.2.

Para geração dos gráficos de convergência utilizou-se, para cada iteração, o melhor resultado das dez execuções, sendo gerado um gráfico de convergência para cada arquivo lido. Para geração dos gráficos boxplot considerou-se o resultado final (número de cláusulas verdadeiras) de cada uma das dez execuções. Os resultados da média, desvio padrão e a variância também têm relação com os valores citados anteriormente

4. Descrição das Simulações e Resultados Obtidos

Dada as configurações dos algoritmos apresentada na seção anterior, a seção atual apresentará as simulações feitas para os três casos do 3-SAT. O Objetivo das simulações foi maximizar o número de cláusulas verdadeiras de acordo com valores *true* e *false* das variáveis que compõem as instâncias do problema. A Tabela 1 apresenta os resultados da média, variância de desvio padrão obtidos através das dez execuções de cada arquivo lido. A Tabela 2 apresenta os resultados obtidos nas dez execuções de cada arquivo lido por ambos os algoritmos. A Figura 2 apresenta os gráficos de convergência para as execuções com o Simulated Annealing. A Figura 3 apresenta os gráficos de convergência para as execuções com o Random Search. Por fim, a Figura 4 apresenta o boxsplot de ambos os algoritmos para os três arquivos.

Tabela 1. Resultados das médias, variâncias e desvios padrão dos algoritmos Simulated Annealing e Random Search. Fonte: as autoras, 2019.

Arquivo	Algoritmo	Média	Variância	Desvio Padrão
Arquivo 1	SA	90,9	0,09	0,3
	RS	90,9	0,09	0,3
Arquivo 2	SA	403,3	1,81	1,345
	RS	404	0,8	0,894
Arquivo 3	SA	977,1	16,09	4,011
	RS	981,8	16,96	4,118

(a) Resultados das dez execuções com o algoritmo **Simulated Annealing**. Fonte: as autoras, 2019.

Arquivo	Execução	Resultado
Arquivo 1	1	91
	2	91
	3	91
	4	91
	5	91
	6	91
	7	91
	8	91
	9	91
	10	90
Arquivo 2	1	403
	2	402
	3	404
	4	406
	5	404
	6	402
	7	405
	8	402
	9	402
	10	403
Arquivo 3	1	973
	2	976
	3	978
	4	974
	5	973
	6	974
	7	975
	8	981
	9	982
	10	985

(b) Resultados das dez execuções com o algoritmo **Random Search**. Fonte: as autoras, 2019.

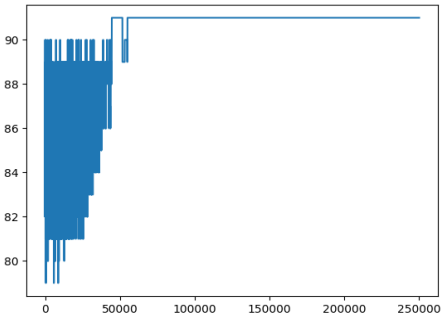
Arquivo	Execução	Resultado
Arquivo 1	1	91
	2	90
	3	91
	4	91
	5	91
	6	91
	7	91
	8	91
	9	91
	10	91
Arquivo 2	1	404
	2	405
	3	403
	4	404
	5	404
	6	405
	7	404
	8	402
	9	404
	10	405
Arquivo 3	1	985
	2	989
	3	978
	4	978
	5	978
	6	979
	7	989
	8	981
	9	980
	10	981

Tabela 2. Resultados das execuções com os algoritmos Simulated Annealing e Random Search. Fonte: as autoras, 2019.

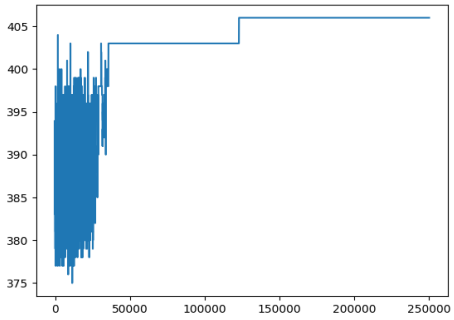
5. Análise dos Resultados Obtidos

Analisando os resultados gerados, pode-se perceber que os melhores resultados dos algoritmos, para cada um dos três arquivos foram:

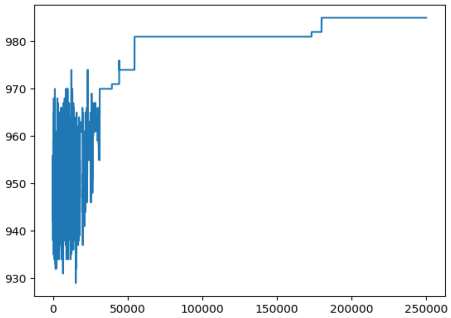
- Arquivo 1: Simulated Annealing (melhor: 91, pior: 90) e Random Search (melhor: 91, pior: 90);



(a) Leitura do arquivo 1.



(b) Leitura do arquivo 2.



(c) Leitura do arquivo 3.

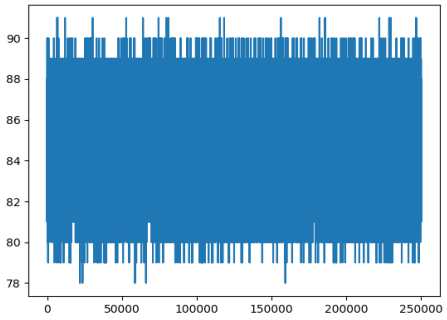
Figura 2. Gráficos de convergência utilizando Simulated Annealing. Fonte: as autoras, 2019.

- Arquivo 2: Simulated Annealing (melhor: 406, pior: 402) e Random Search (melhor: 405, pior: 402);
- Arquivo 3: Simulated Annealing (melhor: 985, pior: 973) e Random Search (melhor: 989, pior: 978);

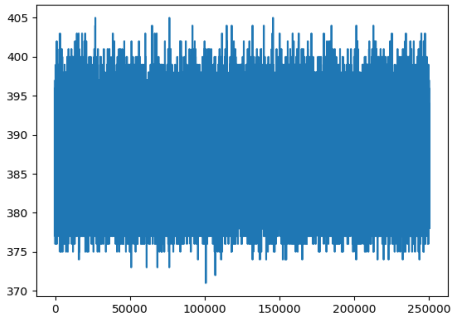
Tendo em vista que os arquivos tinham 91, 430 e 1065 cláusulas respectivamente, é possível perceber que apenas no arquivo 1 os algoritmos conseguiram alcançar o resultado ótimo.

Considerando a média, variância e desvio padrão, é possível perceber que para o arquivo 1 os algoritmos obtiveram os mesmos resultados. Para o arquivo 2, nota-se que o desempenho do Random Search foi melhor que o do Simulated Annealing, isto porque a média obteve um número maior e a variância e o desvio padrão foram menores. Estes dados mostram que, para a execução do arquivo 2, o Random Search gerou resultados mais próximos do ótimo, com resultando mais homogêneos e com variação menor. Por fim, para o arquivo 3, documento que dispunha de mais cláusulas, é possível perceber que, apesar de o Random Search alcançar resultados mais próximos do ótimo (visto que detém a maior média), a variância e o desvio padrão foram maiores que a do Simulated Annealing. Isto significa que os resultados foram mais discrepantes um dos outros.

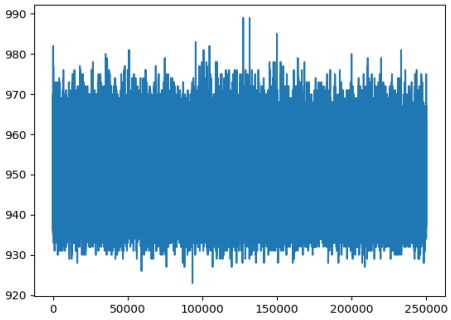
Tendo em vista os gráficos boxplot apresentados, nota-se que para o arquivo 1,



(a) Leitura do arquivo 1.



(b) Leitura do arquivo 2.



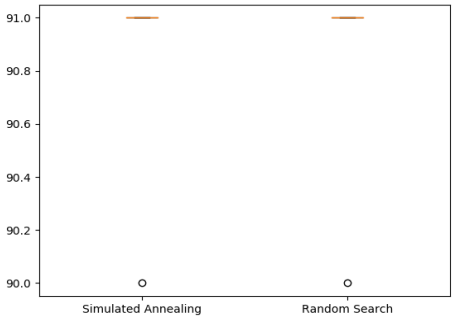
(c) Leitura do arquivo 3.

Figura 3. Gráficos de convergência utilizando Random Search. Fonte: as autoras, 2019.

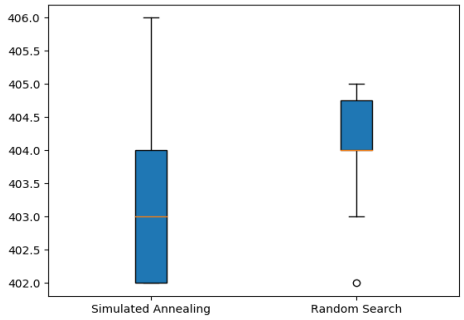
onde alcançou-se o resultado ótimo em ambos os algoritmos, os gráficos são iguais vazios. Para o arquivo 2, o boxplot mostra que os resultados do Random Search foram, em média, maiores que os resultados do Simulated Annealing, bem como menos distribuídos. Apesar desta característica, o Simulated Annealing teve o resultado mais próximo do ótimo, enquanto o Random Search teve o resultado menos próximo do ótimo. No arquivo 3 é possível perceber que os resultados se mostraram discrepantes em ambos os algoritmos, porém, o Simulated Annealing apresentar menor variância e desvio padrão (conforme descrito anteriormente).

Por fim, analisando os gráficos de convergência, é possível perceber que o Simulated Annealing apresenta um comportamento menos divergente que do Random Search. Isto acontece porque o Random Search age sobre um comportamento aleatório. Já o Simulated Annealing possui um comportamento de convergência, ou seja, inicialmente porta-se de maneira mais discrepante (baseado na alta temperatura), sendo que ao longo das execuções a discrepância diminuí (dada a função de resfriamento), convergindo para um resultado.

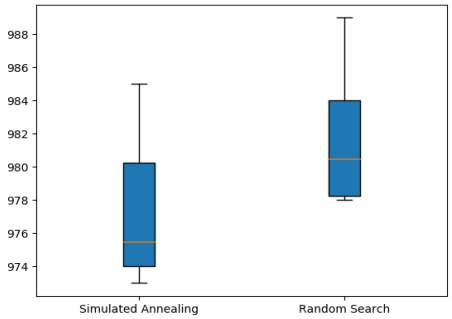
A partir destes resultados, entende-se a alta taxa de convergência do algoritmo Simulated Annealing. Essa característica faz com que, apesar de os algoritmos terem apresentados comportamentos semelhantes em alguns casos, a robustez do Simulated An-



(a) Boxsplot utilizando Simulated Annealing e Random Search na leitura do arquivo 1.



(b) Boxsplot utilizando Simulated Annealing e Random Search na leitura do arquivo 2.



(c) Boxsplot utilizando Simulated Annealing e Random Search na leitura do arquivo 3.

Figura 4. Boxsplot utilizando Simulated Annealing e Random Search. Fonte: as autoras, 2019.

nealing faz com que seja um algoritmo mais confiável.

6. Conclusões e Trabalhos Futuros

O presente trabalho teve como objetivo o estudo e a implementação dos algoritmos de busca Simulated Annealing e Randon Search para resolução de instância do problema de 3-satisfatibilidade booliana (3-SAT). A partir do desenvolvimento, foi possível realizar uma análise do comportamento dos algoritmos diante de mudanças do tamanho dos arquivos de entrada.

Conclui-se a eficácia e eficiência dos algoritmos implementados, dado a comprovação das características de completude e a otimalidade (unicamente na leitura do arquivo 1). Percebeu-se ainda que os algoritmos tiveram comportamentos semelhantes quando analisados diretamente os resultados das cláusulas, o que surpreendeu as autoras, visto que a robustez do Simulated Annealing é bem maior que a do Random Search. Porém, exclusivamente por esta característica, as autoras acreditam que o Simulated Annealing seja mais confiável. Além disso, quando analisado outros critérios, como variância, desvio padrão e convergência, é notório que o desempenho do Simulated Annealing é mais significativo.

Identifica que é passível como trabalhos futuros testes com condições de parada menor que 250 mil iterações e acima deste valor, pois percebeu-se que no arquivo 1 não é necessário um número tão grande de iterações, já no arquivo 3, talvez um critério de parada maior poderia melhorar a sua performance.

Referências

Aarts, E. H., Kordt, J. H., and van Laarhoven, P. J. (2005). Simulated annealing. In *Search methodologies*, pages 187–210. Springer.

Browlee, J. (2015). Clever algorithms: Nature-inspired programming recipes. `http://www.cleveralgorithms.com/nature-inspired/stochastic/random_search.html`. Acessado em: 27/10/2019.

de Araujo, H. A. (2001). Algoritmo simulated annealing: uma nova abordagem. Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Direne, A. I. (2018). Algoritmos de busca heurística (parte 1). `http://www.inf.ufpr.br/alexd/abh/abh1.pdf`. Acessado em: 27/10/2019.

Haeser, G. and Ruggiero, M. G. (2008). Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global. *Trends in Applied and Computational Mathematics*, 9(3):395–404.

Lasma, F. A. F. (2010). Análise de desempenho de algoritmos para o problema da satisfatibilidade booleana. Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras.

Schumer, M. A. and Steiglitz, K. (1968). Adaptive step size random search. *IEEE Transactions on Automatic Control*, 13(3):270–276.

Sipser, M. (2005). *Introdução à teoria da computação*. Cengage Learning, 1ª edition.

Tomazi, F., Bobsin, C., and dos Santos, J. V. C. (2016). Uma aplicação do algoritmo metaheurístico simulated annealing para o problema de seleção de contingências em análise de segurança de redes elétriccas. *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacionais (SBPO)*.

Tovey, C. A. (1984). A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89.

Udemy (2018). Inteligência artificial: Algoritmos inteligentes de busca. `https://www.udemy.com/course/inteligencia-artificial-algoritmos-inteligentes-de-busca/`. Acessado em: 06/10/2019.