Simulated Anneling, Greedy Search, Random Search Uma comparação e uma possivel solução para espressões 3 CNF-SAT extensas

Adilson Krischanski[1]

[1]Udesc CCT - Universidade do Estado de Santa Catarina - Campus de Ciencias Tecnologicas
Caixa Postal 89.219-710 – Joinville – SC – Brazil
Departamento de Ciencia da Computacao

{adilson.krischanski}@edu.udesc.br

Abstract. *In the present study, an experiment will be conducted to compare the optimization algorithms Simulated Annealing, Greedy Search, and Random Search. The aim is to find feasible combinations (in terms of time) and maximize the number of satisfied clauses in a Boolean expression. The objective is to investigate and analyze the performance of these algorithms in solving this particular problem type.*

Resumo. *No artigo a seguir segue o relato sobre um experimento para comparar os algoritmos de otimização Simulated Annealing, Greedy Search e Random Search, a fim de que seja possivel encontrar combinações viáveis (tempo) e assim maximizar a quantidade de cláusulas satisfeitas em uma expressão booleana. Analizando o desempenho desses algoritmos entre si na resolução dessa classe de problemas.*

1. Introducao

Problemas de otimização são situações onde buscamos encontrar uma possivel solução dentro de um vasto campo de busca. Esses problemas podem ser classificados como de maximização ou minimização de uma função objetivo, a qual está sujeita a um conjunto de restrições. O objetivo é encontrar possiveis valores para cada uma das variáveis de decisão que satisfação o valor da função objetivo. Tais problemas podem ser lineares, não lineares, discretos, contínuos ou combinatórios. Resolver problemas de otimização torna-se desafiador devido ao alto custo computacional dado o grande espaço de busca em algumas situações.

2. Problematica SAT

A problematica SAT (Satisfiability) é um dos problemas mais estudados na área de ciência da computação bem como teoria da complexidade. Ele pertence à classe de problemas NP-completos, ou seja, significa que não há um algoritmo conhecido que seja capaz de executar em tempo polinomial.

Para que solucionar o problema é necessario determinar se existe uma atribuição de valores verdadeiro ou falso para um conjunto de variáveis que satisfaça a expressao booleana desejada. A expressão booleana pode ser expressa de várias formas, sendo a forma mais comum a forma conjuntiva normal (CNF), onde a fórmula é uma conjunção de cláusulas, e cada cláusula é uma disjunção de literais (variáveis ou suas negações).

O principal objetivo da problematica é verificar se é possível encontrar uma atribuição de valores para as variáveis que torne toda a fórmula verdadeira, assim sendo atribuição, dizemos que a fórmula é satisfatível (uma tautologia), caso contrário, dizemos que a fórmula é insatisfatível.

3. Algoritmos

3.1. Simulated Anneling

O Simulated Annealing, é uma técnica de otimização estocástica utilizada em problemas que envolvem a busca por uma solução ótima em um espaço de busca muito grande ou complexo. Essa técnica foi inspirada pelo processo de recozimento muito utilizado na metalurgia para realizar a tempera de materias, nesse processo o material é aquecido e resfriado lentamente para reduzir a sua energia e aumentar a sua resistência. De forma analogo, o algorimo Simulated Annealing começa com uma solução inicial em alta temperatura (aleatória) e aprimora essa solução através da exploração do espaço de busca por soluções melhores(baixa temperatura), aceitando, em alguns casos, soluções piores do que a atual, para evitar a estagnação em um mínimo local. O método é baseado em uma probabilidade de aceitação de soluções piores que diminui com o tempo, assim como o resfriamento em um processo de recozimento, o que faz com que a solução final seja na em grande maioria a melhor encontrada durante a busca.

3.1.1. Temperatura

O controle da temperatura é um aspecto fundamental no algoritmo Simulated Annealing, pois é através da variação da temperatura que o algoritmo explora o espaço de busca de soluções de forma eficiente. A temperatura é a responsavel por controlar a probabilidade de aceitação de soluções com avaliação inferior a a atual, permitindo assim que o algoritmo possa escapar de atratores locais e alcançar soluções ótimas globais.

No início do algoritmo, a temperatura é alta, o que permite a aceitação de soluções piores com uma probabilidade muito alta, todavia Com o decorrer do tempo a temperatura é reduzida, e então a probabilidade de aceitar soluções piores diminui, o que permite ao algoritmo possar convergir para uma solução ótima.

Um dos grandes desafios na implementação do Simulated Annealing é determinar a taxa de resfriamento adequada, que controla a redução da temperatura ao longo do tempo. Se a taxa de resfriamento for muito rápida, o algoritmo pode não ter tempo suficiente para explorar todo o espaço de busca, enquanto que, se for muito lenta, o algoritmo pode levar muito tempo para convergir para uma solução ótima. Portanto, encontrar a taxa de resfriamento adequada é uma questão crítica para a eficácia do algoritmo.

3.2. Greedy search

Greedy Search, o algoritmo de busca gulosa é uma estratégia simples que busca uma solução ótima local em um espaço de busca. Ele seleciona a opção mais promissora em cada etapa com base em uma função heurística, sem considerar o impacto futuro. Embora seja eficiente em termos de tempo, pode ficar facilmente preso em mínimos ou máximos locais, falhando assim em encontrar a melhor solução global. É amplamente usado quando a solução local é suficiente ou em problemas complexos onde a busca exaustiva não é viável.

3.3. Random search

Randon Search. algoritmo de busca aleatória é a abordagem mais simples e direta conhecida. servindo para explorar soluções em um espaço de busca. Ele começa gerando uma solução aleatória, em seguida, avalia essa solução com base em uma função de avaliação da problematica. O algoritmo continua gerando novas soluções de forma aleatórias e as avaliando, mantendo armazenada a melhor solução encontrada. Embora não utilize informações estruturais do problema, a busca aleatória pode ser útil quando não há informações suficientes disponíveis. No entanto, ela pode ter dificuldade em encontrar soluções ótimas em problemas complexos ou com espaços de busca grandes.

4. Experimento

O experimento consiste em realizar uma comparação de entre os algortimos de otmização de busca aleatoria (Randon Search), um algotimo Guloso (Greddy Searsh) e o Simulated Anneling para tres casos de umas espressao boleana de na sua forma normal conjuntiva 3 CNF - SAT, das tres espressoes a primeira possui uma 20 variaveis e 91 causulas, a segunda possuia 100 variaveis e 430 clausulas a terceira possui 250 variaveis e 1065 clausulas.

4.1. Linguagem de Programação

Para a implementação do projeto, a a linguagem escolhida foi o Python, isso devido à sua popularidade (caso alguem queira replicar o experimento em condições similares), facilidade de uso e ampla disponibilidade de bibliotecas. Entre elas, destaca-se o Matplotlib, que permite a criação de gráficos e análises dos dados do experimento. Essa capacidade de visualização e análise dos dados é essencial para obter insights e embasar decisões. Ao utilizar Python, os desenvolvedores optaram por uma solução eficiente e elegante, que promove uma exploração detalhada dos resultados e contribui para um melhor entendimento do experimento realizado.

4.2. Gerando a Solucao Inicial

A sulução inicial é gerada igualmente para todos todos os tres casos, para isso inicialmente sao verificados qual a quantidade de variveis contem o problema, em seguida é gerada de forma aleario o valor 0 ou 1 para cada uma das variveis, e essa é a nossa solucao inicial.

4.3. Gerando a Nova Solução

A nova solução é gerada de duas formas diferentes, dependendo do algoritmo utilizado. Em primeiro lugar, pode ser gerada de forma completamente aleatória, o que significa que é gerada como a solução inicial para o algoritmo de busca aleatória. Por outro lado, para os algoritmos de busca gulosa e Simulated Annealing, a nova solução é gerada a partir de uma solução vizinha da solução atual.

Para gerar o vizinho, é escolhido empiricamente um valor máximo para o número de variáveis que terão seus valores alterados. Durante os experimentos, foram utilizados valores aleatórios entre 1 e a raiz quadrada da quantidade total de variáveis. É importante destacar que as mudanças no vizinho podem fazer com que algoritmos como o Simulated Annealing se comportem de maneira semelhante a algoritmos de busca aleatória.

4.4. Fitness - Função de Avaliação

A função de avaliação é dada pela expressão booleana que está sendo analisada. Essa função conta a quantidade de cláusulas que são satisfeitas pela solução proposta. assim sendo essa medida pode ser transformada em um valor percentual, permitindo a análise em duas perspectivas: maximização e minimização.

4.4.1. Maximização

A maximização da função objetivo refere-se ao objetivo de encontrar uma solução que maximize o valor dessa função. No contexto da otimização, isso significa buscar a solução que satisfaça o maior número possível de cláusulas na expressão booleana. O objetivo é encontrar a solução que atinja o maior valor percentual possível, indicando que a maioria das cláusulas está sendo satisfeita.

$$Solucao = \left(\frac{Clausulasresolvidas}{QuantidadedeClausulas}\right) \times 100$$

4.4.2. Minimização

Minimização da função objetivo envolve o objetivo de encontrar uma solução que minimize o valor dessa função. Nesse caso, o objetivo é encontrar a solução que satisfaça o menor número de cláusulas na expressão booleana. Busca-se minimizar o valor percentual, indicando que poucas cláusulas estão sendo satisfeitas ou que há um grande número de cláusulas insatisfeitas.

$$Solucao = 1 - \left(\frac{Clausulasresolvidas}{QuantidadedeClausulas}\right) \times 100$$

4.5. Queda da temperatura

A queda de temperatura no algoritmo do Simulated Annealing é de longe o fator mais importante do algoritmo. Ela é responsável por conduzir o algoritmo a explorar inicialmente um espaço de busca mais amplo, fazendo uma transição gradual para uma busca mais focada nas etapas posteriores, mantendo ao mesmo tempo a estocasticidade para escapar de ótimos locais e garantir a convergência.

Para este experimento, a equação escolhida para o decaimento da temperatura é apresentada abaixo. Ela foi selecionada empiricamente e tem se mostrado eficiente.

$$fatorResfriamento = 4$$
$$NovaTemperatura = \left(1 - \frac{interaction\_number}{maxquantInteraction}\right)^{fatorResfriamento}$$

4.6. Tempo e Hardware

O experimento foi conduzido em um processador AMD Ryzen 3 de 3ª geração, com 8 núcleos e 16 GB de memória RAM disponível. Embora o código não tenha threads, foram executadas 3 instâncias em paralelo usando um script de shell. O sistema operacional utilizado foi o Ubuntu 2022 LTS. O tempo médio de execução foi semelhante para os três algoritmos, com aproximadamente 0m0.6s para o de 20 variáveis, 0m3s para o de 100 variáveis e 1m15.5s para o de 250 variáveis.

5. Resultados

A tabela apresenta uma comparação dos resultados obtidos em três casos diferentes de busca heurística aplicados a problemas específicos representados pelos casos ”uf20-91-01”, ”uf100-91-01” e ”uf250-91-01”. Os três casos são: Greedy Search, Random Solution e Simulated Annealing .

Analisando os resultados, podemos observar que o método Simulated Annealing supera os outros dois métodos em todos os casos apresentados. Ele alcança 100.00% de eficiência em encontrar uma solução ótima para o caso ”uf20-91-01” e mantém uma alta porcentagem de eficiência nos outros dois casos. Além de que tem o menor desvio padrão em todos os casos de execução.

Esses resultados sugerem que o Simulated Annealing é uma abordagem eficaz para a busca heurística nessas instâncias de problemas específicos. Ele demonstra a capacidade de encontrar soluções ótimas com alta eficiência e em um tempo relativamente curto, tornando-o uma escolha superior em comparação com o Greedy Search e a Random Solution.

Caso	Greedy search		Random Solution		Simulated Anneling	
uf20-91-01	100.00%	0.000	99.78%	0.463	100.00%	0.000
uf100-91-01	99.60%	0.311	94.07%	0.295	99.81%	0.240
uf250-91-01	99.54%	0.219	92.04%	0.246	99.56%	0.166

5.1. Gráficos

5.1.1. 20 Variáveis

Nesta imagem, podemos analisar o desempenho de cada algoritmo e compará-los no contexto de uma expressão com 20 variáveis e 91 cláusulas.

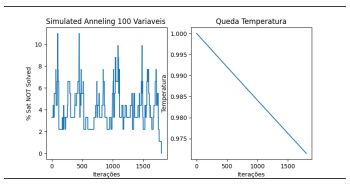

Figura (a): Simulated Annealing

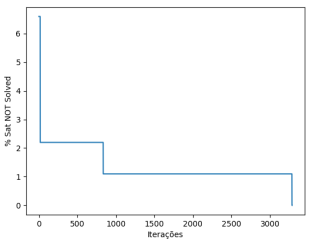

Figura (b): Greedy Search

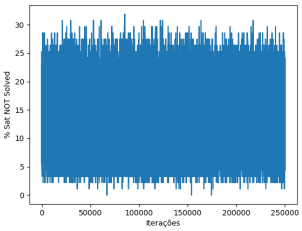

Figura (c): Random Search

5.1.2. 100 Variáveis

Nesta imagem, podemos analisar o desempenho de cada algoritmo e compará-los no contexto de uma expressão com 100 variáveis e 430 cláusulas.

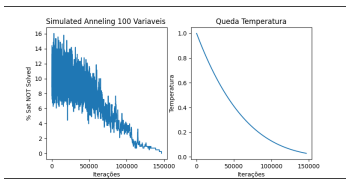

Figura (a): Simulated Annealing

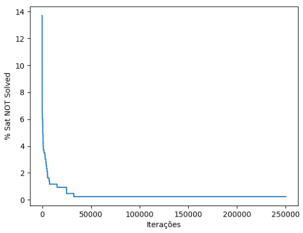

Figura (b): Greedy Search

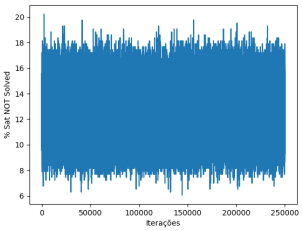

Figura (c): Random Search

5.1.3. 250 Variáveis

Nesta imagem, podemos analisar o desempenho de cada algoritmo e compará-los no contexto de uma expressão com 250 variáveis e 1065 cláusulas.

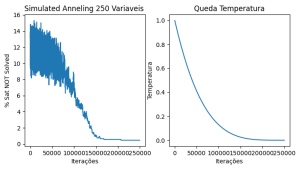

Figura (a): Simulated Annealing

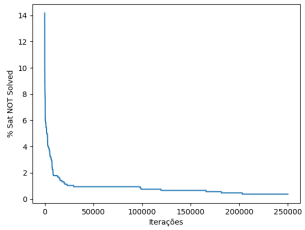

Figura (b): Greedy Search

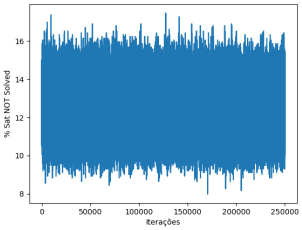

Figura (c): Random Search

6. Conclusão

Na figura apresentada abaixo, é possível observar uma comparação entre as três situações de instanciamento da problemática SAT, juntamente com as tentativas de solução por meio dos algoritmos propostos. Ao analisar o gráfico, destaca-se a eficiência do Simulated Annealing, o que demonstra resultados mais consistentes e estáveis em comparação com o Greedy Search e o Random Search. Sua capacidade de encontrar soluções satisfatórias de forma mais robusta e confiável é evidente no contexto avaliado.

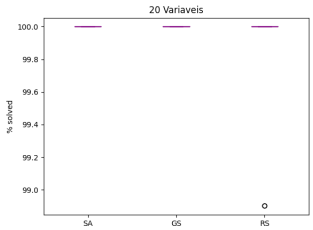

Figura a: Boxplot para 20 variáveis

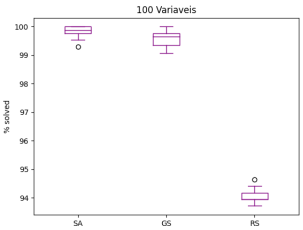

Figura b: Boxplot para 100 variáveis

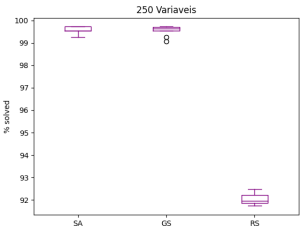

Figura c: Boxplot para 250 variáveis

7. Referências de pesquisa

The satisfiability problem. Disponível em: ¡http://www.cs.ecu.edu/karl/6420/spr16/Notes/NPcomplete/sat.h

RUTENBAR, R. A. Simulated annealing algorithms: an overview. IEEE Circuits and Devices Magazine, v. 5, n. 1, p. 19–26, jan. 1989.

Simulated Annealing Algorithm - an overview — ScienceDirect Topics. Disponível em: ¡https://www.sciencedirect.com/topics/engineering/simulated-annealing-algorithm¿.

KRISCHANSKI, A. IA. Disponível em: ¡https://github.com/adilsonkrischanski/IA¿.

CHATGPT. ChatGPT. Disponível em: ¡https://chat.openai.com/¿.