Relatório de aplicação de Simulated Annealing em problema de satisfabilidade booleana

Bruno M. Pires[1]

[1]Departamento de Ciências da Computação
Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 15.064 – 88.035-901 – Joinville – SC – Brasil

bruno.pires@edu.udesc.br

Abstract. *In this report, we conducted a thorough analysis of the practical application of the Simulated Annealing (SA) algorithm, also known as Simulated Annealing, to a Boolean Satisfiability (SAT) problem. Within the scope of this study, our aim was to compare the results obtained using SA to those obtained with a random search algorithm. This approach not only seeks to understand the operation of SA but also to demonstrate its efficiency when compared to a straightforward random search algorithm.*

Resumo. *Neste relatório, realizamos uma análise minuciosa da aplicação prática do algoritmo Simulated Annealing (SA), também conhecido como Recozimento Simulado, a um problema de Satisfatibilidade Booleana (SAT). No âmbito deste estudo, buscamos comparar os resultados obtidos com o uso do SA em relação a um algoritmo de busca aleatória. Essa abordagem visa não apenas compreender o funcionamento do SA, mas também comprovar sua eficiência quando comparado a um algoritmo simples de busca aleatória.*

1. Introdução

Dentro de Inteligência artificial a preocupação pela busca de soluções ótimas é uma constante e a otimização desempenha um papel central em muitos campos, desde o design de sistemas complexos até o treinamento de algoritmos de aprendizado de máquina. Fazendo uma analogia, é possível explicar a otimização como a busca de uma agulha em um palheiro, um problema com diversas possibilidades, que representa muito bem a complexidade de muitos problemas que precisam de fato de otimização para encontrarem soluções, e é nesses contextos que os algoritmos de otimização desempenham um papel crucial.

O Recozimento Simulado emerge como uma ferramenta poderosa e versátil inspirada no processo físico de têmpera de aço, fornecendo uma abordagem única para solução de problemas de otimização. Neste relatório, vamos explorar o algoritmo em profundidade sendo aplicado a um problema de Satisfabilidade Booleana.

Pra entender o funcionamento geral do simulated annealing, vamos fazer uma analogia com a confecção de uma espada por um ferreiro, onde o simulated annealing representa o ferreiro. O primeiro passo, chamado de inicialização ou aquecimento inicial, temos o ferreiro aquecendo a peça bruta de metal a altas temperaturas da mesma forma que o simulated annealing começa com uma soluçao inicial e a ”aquece” (Define a temperatura de início), estamos considerando aqui a peça de metal como uma solução inicial.

A partir da primeira etapa, o ferreiro começa a moldar a peça bruta de metal com marteladas de forma a ajustar a forma do metal, analogamente, o simulated annealing explora o espaço de soluções realizando pequenas modificações na solução atual de forma a criar variantes, e cada modificação é análoga a um golpe de martelo alterando e moldando a solução. A terceira etapa está relacionada á qualidade, frequentemente o ferreiro analisa a lâmina para verificar sua qualidade da mesma forma que o simulated annealing avalia sua solução com o que chamamos de Função de custo. Na quarta etapa, da mesma forma que o ferreiro resfria a lâmina em um ritmo controlado pra evitar a quebra, o simulated annealing resfria, ou seja, diminuia a taxa de aceitação de soluções piores à medida que o algoritmo progride, e isso ajuda a evitar que o algoritmo fique preso em mínimos locais e permite explorar soluções potencialmente melhores. Por fim, o ferreiro decide se a forma atual da espada (solução) está adequada ou não, decidindo assim se o ciclo irá recomeçar.

O artigo tem como objetivo relatar o que se foi observado aplicando esta técnica em um problema de satisfabilidade booleana chamado "3-SAT". O 3-SAT (Three-Satisfiability) é um problema de decisão na teoria da complexidade computacional e é uma variação do problema SAT (Satisfabilidade Booleana). De forma simples, temos fórmulas booleanas compostas por cláusulas, que por sua vez possúem três variáveis booleanas e o objetivo é encontrar uma solução para essas variáveis de forma que a fórmula inteira seja avaliada como uma Tautologia. A formula mais simples do problema é representada por uma forma normal conjuntiva, uma conjunção de cláusulas onde cada clausula é uma disjunção de três variáveis. Um bom exemplo está a seguir, onde F1, F2, F3, ..., Fn representam as variáveis.

$$\begin{aligned} &(F1 \vee \neg F2 \vee F3) \wedge (\neg F1 \vee F2 \vee F4) \wedge (F2 \vee \neg F3 \vee F5) \wedge \\ &\dots \\ &\wedge (\neg Fn-2 \vee Fn-1 \vee \neg Fn) \end{aligned} \tag{1}$$

O relatório estará estruturado em 6 seções, esta dedicada a Introdução, uma dedicada a explicar o desenvolvimento do trabalho, as próximas três apresentam os resultados obtidos, e por último a conclusão e trabalhos futuros.

2. Desenvolvimento

Assim como no desenvolvimento de software, inicialmente foi necessário realizar uma espécie de "modelagem" do problema, pra enteder melhor suas características. De forma básica, a tabela mostrada abaixo foi preenchida, utilizando os conceitos também mostrados abaixo:

- Agentes: O agente é a entidade que toma decisões e interage com o ambiente.
- Observável: Refere-se à capacidade do agente de observar o estado atual do ambiente.
- Determinístico: Indica se o comportamento do agente é completamente previsível com base no estado atual e nas ações tomadas.
- Episódico: Indica se o agente toma decisões em episódios ou etapas discretas, onde as ações tomadas em um episódio não afetam diretamente os episódios subsequentes.
- Estático: Refere-se a se o ambiente muda ao longo do tempo.

- Discreto: Indica se o espaço de ações ou estados do agente é discreto, o que significa que existem um número finito e bem-definido de opções.
- Multiagente: Indica se existem múltiplos agentes interagindo no mesmo ambiente

Agente	Observável	Determinístico	Episódico	Estático	Discreto	Multiagente
SA	Sim	Não	Sim	Sim	Sim	Não

Table 1. Modelagem das características 3-SAT

Compreendendo as características do ambiente, é possível iniciar a codificação. Para facilitar os testes, foi desenvolvida uma ferramenta em python utilizando uma biblioteca de interface gráfica que gera HTML chamada ”streamlit” juntamente com uma biblioteca para plotagem de gráficos chamada plotly. A interface em HTML é responsável pelas parametrizações do sistema (conjunto de dados, quantidade de iterações, quantidade de execuções). A figura 1 representa o menu da ferramenta.

Figure 1. Tela Inicial Ferramenta de aplicação Simulated Annealing

O trabalho está estruturado de forma a testar 3 conjuntos de clausulas, sendo os casos:

- uf20-01.cnf
- uf100-01.cnf
- uf250-01.cnf

Além disso, aplicamos Simulated Annealing e comparamos cada caso com ”Random Search” para evidenciar a eficiência da implementaçao do SA. Cada caso também contará com duas implementações diferentes em termos de temperatura sendo a primeira uma função exponencial e a segunda uma função linear.

Métrica	Random Search	Simulated Annealing Linear	Simulated Annealing Exponencial
Média	0.2	0.0	0.0
Desvio Padrão	0.4	0.0	0.0

Table 2. Resultados para 100 variáveis

3. Caso uf20-01.cnf

O primeiro caso é bastante simples e consiste em uma Forma Normal Conjuntiva de apenas 20 variáveis e com 91 cláusulas a serem atendidas. Como parametrização, temos 250.000 avaliações de função, e 10 execuções. A tabela 2 apresenta todos os resultados obtidos e o boxplot da figura 2 deixa bem claro que todas as execuções do SA convergiram, e a maioria das execuções de random search também, porém algumas execuções se tornaram outliers e não alcançaram um valor ótimo. As próximas duas sub-seções são dedicadas a apresentação e análise dos gráficos e execuções realizadas.

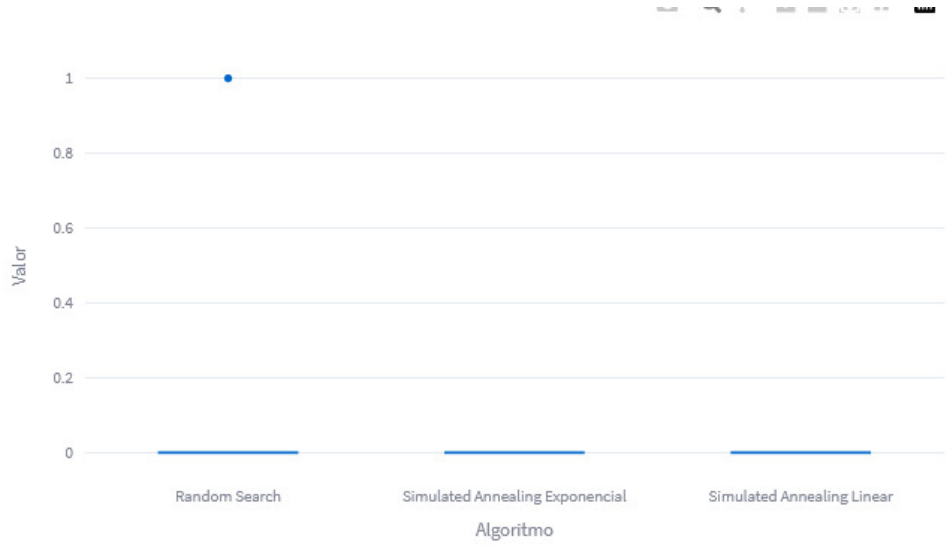

Figure 2. Boxplot de 10 execuções para 20 variáveis

3.1. Temperatura Exponencial - uf20-01.cnf

Para uma equação de energia exponencial, a figura 3 apresenta a média das 10 execuções para Simulated Annealing e Random Search, enquanto a figura 4 apresenta o gráfico com o comportamento da temperatura conforme as iterações. Conseguimos observar que por ser um caso bastante simples o algoritmo converge de forma bem rápida, e em 50.000 iterações já temos todas as cláusulas satisfeitas para todas as execuções do Simulated Annealing, o que não é verdade para random search.

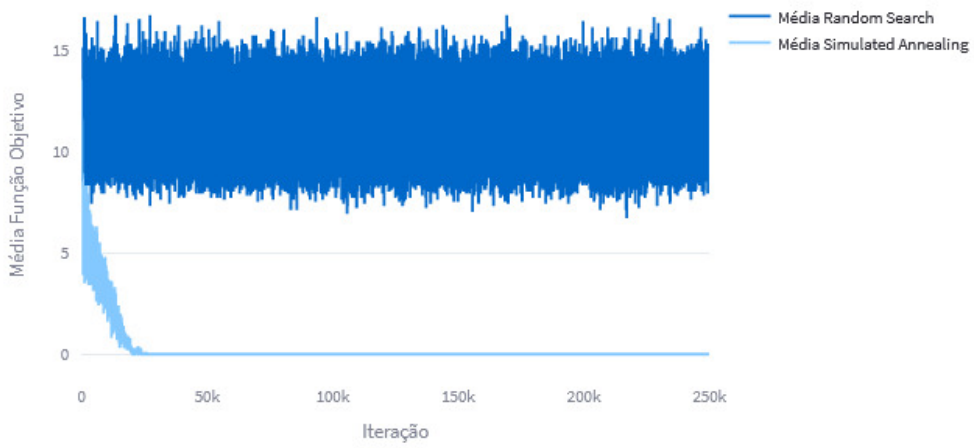

Figure 3. Convergência - 20 variáveis - Cooling Exponencial

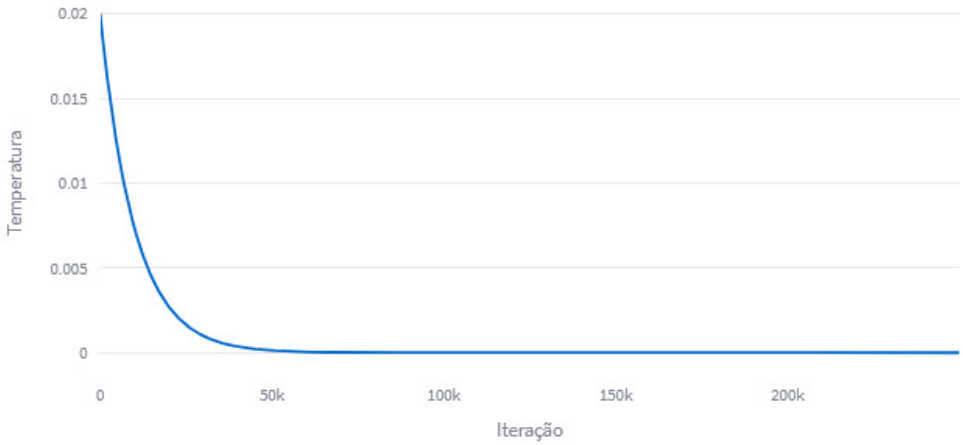

Figure 4. Temperatura utilizada (Exponencial)

3.2. Temperatura Linear - uf20-01.cnf

Além da temperatura exponencial, também foi realizado experimento utilizando temperatura linear, a média está presente na figura 5 enquanto a temperatura está na figura 6. Aqui percebemos que o algoritmo fica por um tempo maior realizando buscas globais, em comparação com o teste anterior, e converge próximo das 225.000 iterações.

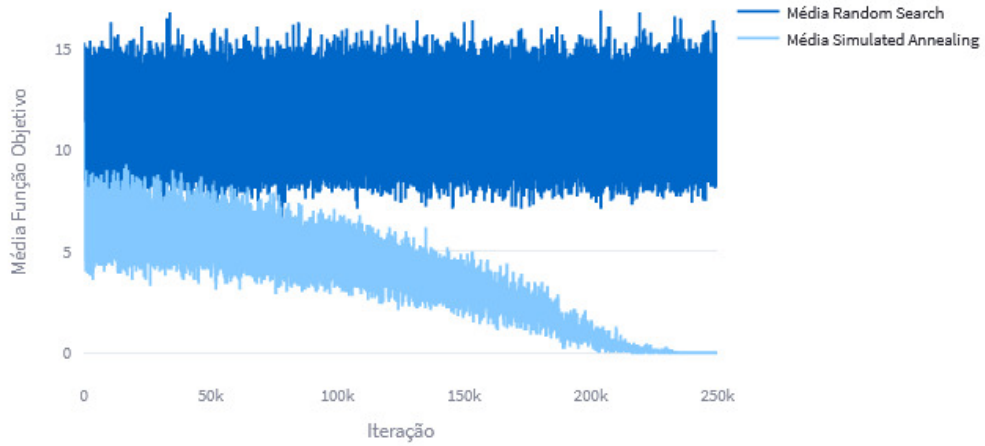

Figure 5. Convergência - 20 variáveis - Cooling Linear

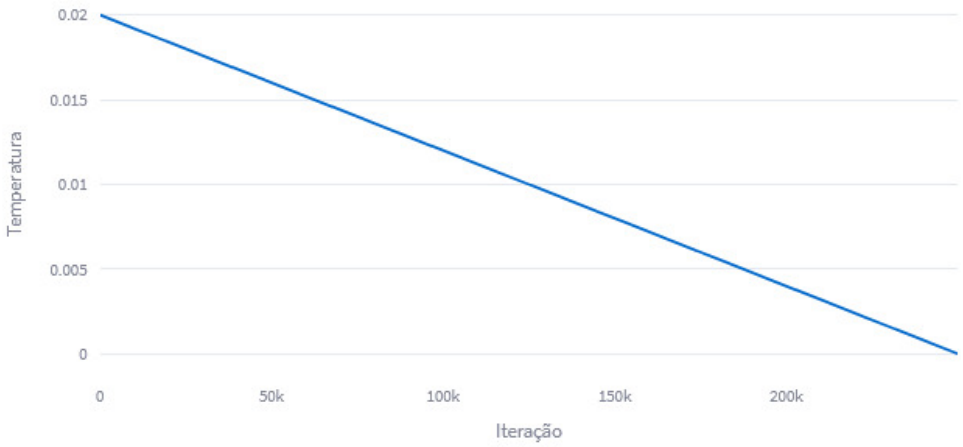

Figure 6. Temperatura utilizada (Linear)

4. Caso uf100-01.cnf

O segundo problema possui 100 variáveis e 430 cláusulas a ser atendidas. Como parametrização, temos 250.000 avaliações de função, e 10 execuções. A tabela 3 apresenta todos os resultados obtidos, evidanciando uma melhor média de cláusulas não atendidas por parte do ”SA Exponencial” com menor desvio padrão, e o boxplot da figura 7 deixa bem claro que algumas das execuções do SA convergiram para um ótimo, porém algumas execuções não alcançaram um valor ótimo, apenas uma aproximação do mesmo. E para o Random Search, finalmente neste caso nenhuma execução alcançou um valor próximo ao ótimo. As próximas duas sub-seções são dedicadas a apresentação e análise dos gráficos e execuções realizadas.

Métrica	Random Search	Simulated Annealing Linear	Simulated Annealing Exponencial
Média	25	1.5	0.8
Desvio Padrão	1.2	1.4	0.7

Table 3. Resultados para 100 variáveis

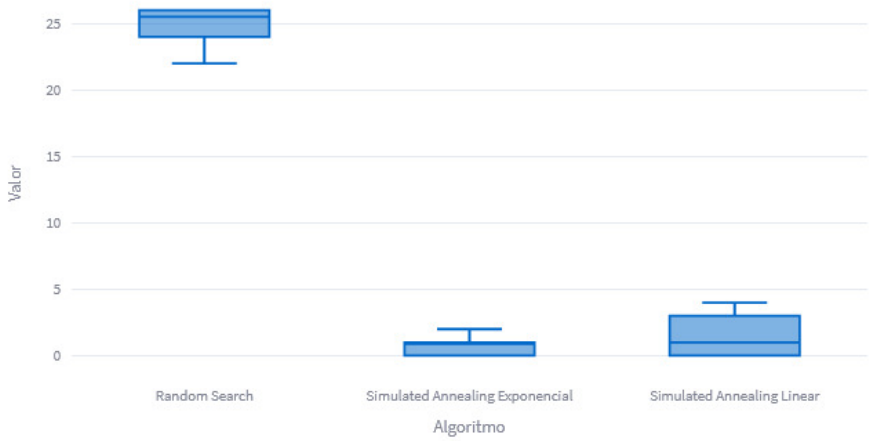

Figure 7. Boxplot das 10 execuções

4.1. Temperatura Exponencial - uf100-01.cnf

Para uma equação de energia exponencial, a figura 8 apresenta a média das 10 execuções para Simulated Annealing e Random Search, enquanto a figura 4 apresenta o gráfico com o comportamento da temperatura conforme as iterações que é o mesmo em todos os casos. Ainda que este caso seja mais complexo que o anterior, conseguimos observar que o SA converge rapidamente devido ao comportamento da temperatura, e em 50.000 iterações já temos uma aproximação do ponto ótimo em alguma execuções, e o ponto ótimo de fato em outras (evidente no boxplot da figura 7).

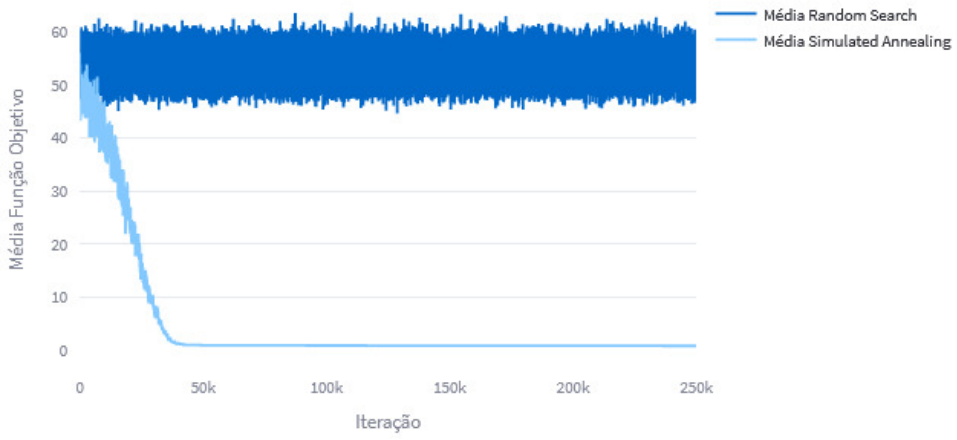

Figure 8. Convergência - 100 variáveis - Cooling Exponencial

4.2. Temperatura Linear - uf100-01.cnf

Por outro lado, para uma equação de energia com comportamento linear, temos uma convergência mais lenta e consequentemente uma média maior de cláusulas não atendidas, mas ainda sim como no caso exponencial, algumas execuções obtiveram sucesso ao encontrar uma solução ótima. A figura 9 deixa claro o comportamento de convergência.

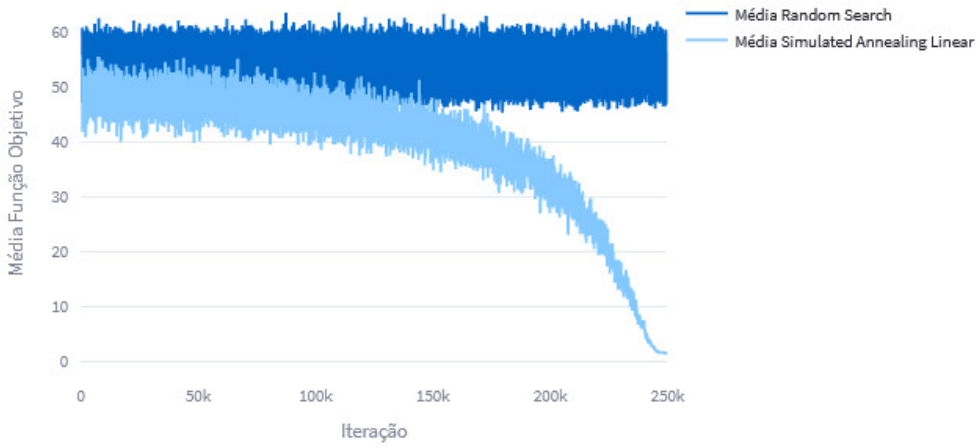

Figure 9. Convergência - 100 variáveis - Cooling Linear

5. Caso uf250-01.cnf

Este é o caso mais complexo, aqui temos um total de 250 variáveis e 1065 cláusulas a serem atendidas. Como parametrização, temos 250.000 avaliações de função, e 10 execuções. A tabela 4 apresenta todos os resultados obtidos, evidanciando uma melhor média de cláusulas não atendidas por parte do ”SA Exponencial” com menor desvio padrão, e o boxplot da figura 10 deixa bem claro que algumas das execuções do SA convergiram para um ótimo, porém algumas execuções não alcançaram um valor ótimo, apenas uma aproximação do mesmo. E para o Random Search novamente nenhuma execução alcançou um valor próximo ao ótimo. As próximas duas sub-seções são dedicadas a apresentação e análise dos gráficos e execuções realizadas.

Métrica	Random Search	Simulated Annealing Linear	Simulated Annealing Exponencial
Média	84.4	4.6	1.1
Desvio Padrão	3.6	2.0	0.5

Table 4. Resultados para 250 variáveis

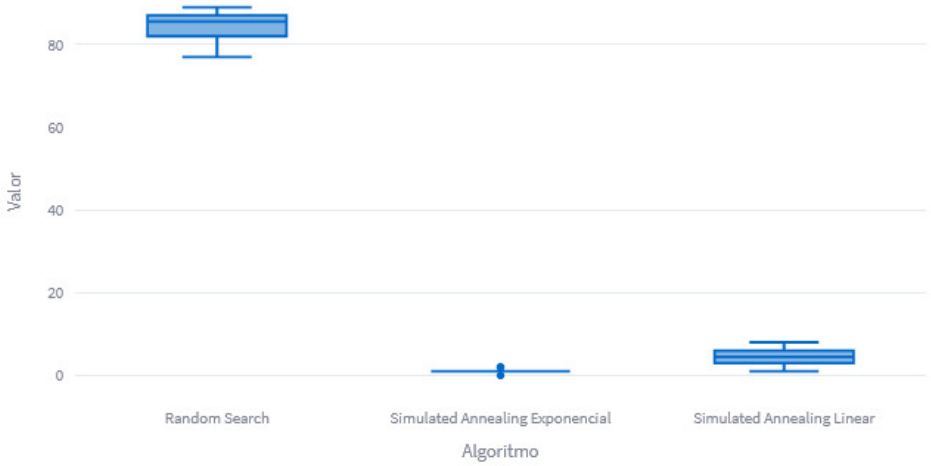

Figure 10. Boxplot para 10 execuções - 250 variáveis

5.1. Temperatura Exponencial - uf250-01.cnf

Utilizando uma equação de energia exponencial, tivemos os melhores resultados, evidente no boxplot da figura 10, conseguimos uma execução que tornou-se um outlier no gráfico que atingiu um valor ótimo com todas as cláusulas satisfeitas, as demais execuções conseguiram uma aproximação muito boa evidente na média com desvio calculada. A figura 11 mostra o gráfico de convergência para este caso, percebemos uma rápida convergência para buscas locais devido ao comportamento da curva de temperatura, que é o mesmo de todos os outros casos, mostrado na figura 4.

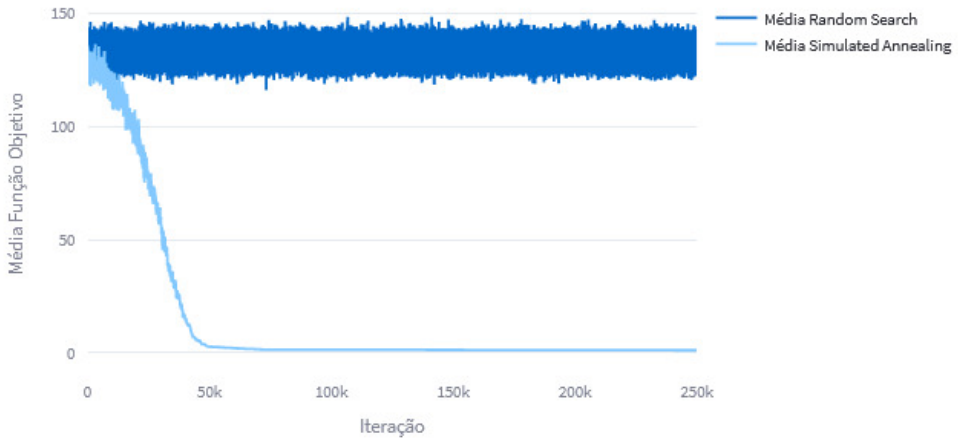

Figure 11. Convergência - 250 variáveis - Cooling Exponencial

5.2. Temperatura Linear - uf250-01.cnf

Utilizando uma equação de temperatura linear, tivemos resultados piores que o caso exponencial, mas também conseguimos boas aproximações em todas as execuções, visto que nenhuma atingiu um valor ótimo com todas as cláusulas satisfeitas. Comparando a média de quase 5 cláusulas não satisfeitas deixa bastante claro a diferença, e a figura 12 demonstra que o algoritmo ficou muito tempo realizando buscas globais e ao convergir não conseguiu mesmo resultado do experimento anterior.

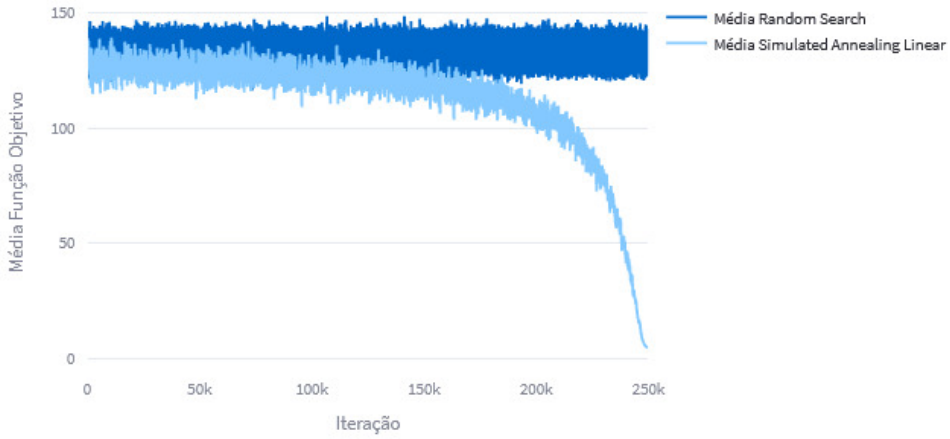

Figure 12. Convergência - 250 variáveis - Cooling Linear

6. Conclusão e Trabalhos Futuros

O problema atacado por mais simples que seja entendê-lo, é complexo em termos de solução, tratamos poucos casos simples e ainda sim em algumas execuções para 100 e 250 variáveis não foi possível encontrar um valor ótimo onde todas as cláusulas foram satisfeitas. Existe um grande limiar de testes que podem ser realizados com principalmente diferentes parâmetros, sejam em iterações ou equações de temperatura que certamente conseguiriam um resultado mais satisfatório.

Entretando, com os testes realizados, ficou evidente a eficiência de aplicação do Simulated Annealing para resolver o problema de satisfabilidade booleana, principalmente quando comparado a um algoritmo de busca aleatória, algo que ficou bastante evidente nos boxplot de 100 e 250 variáveis.

Como trabalhos futuros, o principal ponto a ser atacado é a otimização de parâmetros com finalidade de achar um bom balanço de tempo em busca global e local, e a complexidade e eficiência do código em si, que por se tratar de um problema NP-Completo acaba tendo execuções demoradas de acordo com o aumento das variáveis. Com certeza rodar experimentos de forma paralela para posterior comparação ajudaria a agilizar qualquer análise feita, a ferramenta atual roda tudo de forma sequencial consumindo muito tempo.