UNIVERSIDADE FEDERAL DE UBERLÂNDIA

PROJETO 1 _ INTELIGÊNCIA ARTIFICIAL

Prof: Gina Maria Barbosa

Integrantes do Grupo: Matheus José da Costa - 11711BCC008 /

John Vitor Cunha - 11821BCC005 / Laíz Segatto - 11211BCC033

- Introdução

  O objetivo deste trabalho é obter uma solução computacional para o problema Sudoku, utilizando diferentes algoritmos de busca, os quais solucionarão 2 modelos do problema

  - Sudoku modelo simples: N x N (para grades de tamanho 4x4, 5x5, 6x6 e 9x9)
  - Sudoku modelo complexo: N2xN2 (para grades de tamanho 4x4 e 9x9)

  - Os algoritmos propostos foram:

- Busca em profundidade iterativa: é um algoritmo de busca utilizado para explorar grafos ou árvores em busca de uma solução, permitindo uma busca eficiente e econômica em termos de memória. Realiza uma busca em profundidade, explorando cada ramo do grafo até que atinja uma profundidade máxima pré-determinada. Após atingir essa profundidade, o algoritmo volta ao nó raiz e continua a busca em profundidade com uma profundidade maior.

- Busca gulosa: é um algoritmo de busca heurística que segue uma abordagem de "tomar a melhor decisão no momento". Ele seleciona o próximo passo com base em uma heurística que avalia quais são as melhores opções disponíveis no momento, sem levar em consideração as consequências futuras. Frequentemente é usado em problemas de otimização, onde o objetivo é encontrar uma solução aceitável de forma rápida, mesmo que não seja a solução ideal.

- Algoritmo A*: é um algoritmo de busca informada que combina elementos da busca em largura e busca gulosa para encontrar o caminho mais curto entre um nó inicial e um nó objetivo em um grafo ou em um espaço de estados. Ele é amplamente utilizado em problemas de inteligência artificial, como em sistemas de planejamento, jogos e roteamento. É um algoritmo de busca informada que utiliza uma combinação de busca em largura e busca gulosa para encontrar o caminho mais curto entre um nó inicial e um nó objetivo. Ele é eficiente, completo, ótimo e depende de uma heurística admissível para orientar a busca.

- Subida de Encosta ou Hill Climbing (com Movimentos Laterais): visa superar uma das principais limitações desse último, que é a possibilidade de ficar preso em ótimos locais subótimos. Uma técnica de otimização heurística que busca encontrar o melhor resultado possível em um espaço de solução, considerando uma função objetivo.

A ideia básica é realizar movimentos iterativos a partir de uma solução inicial, selecionando vizinhos que possam melhorar o valor da função objetivo. No entanto, o Hill Climbing tradicional pode ficar preso em ótimos locais subótimos, onde não é possível encontrar soluções melhores.

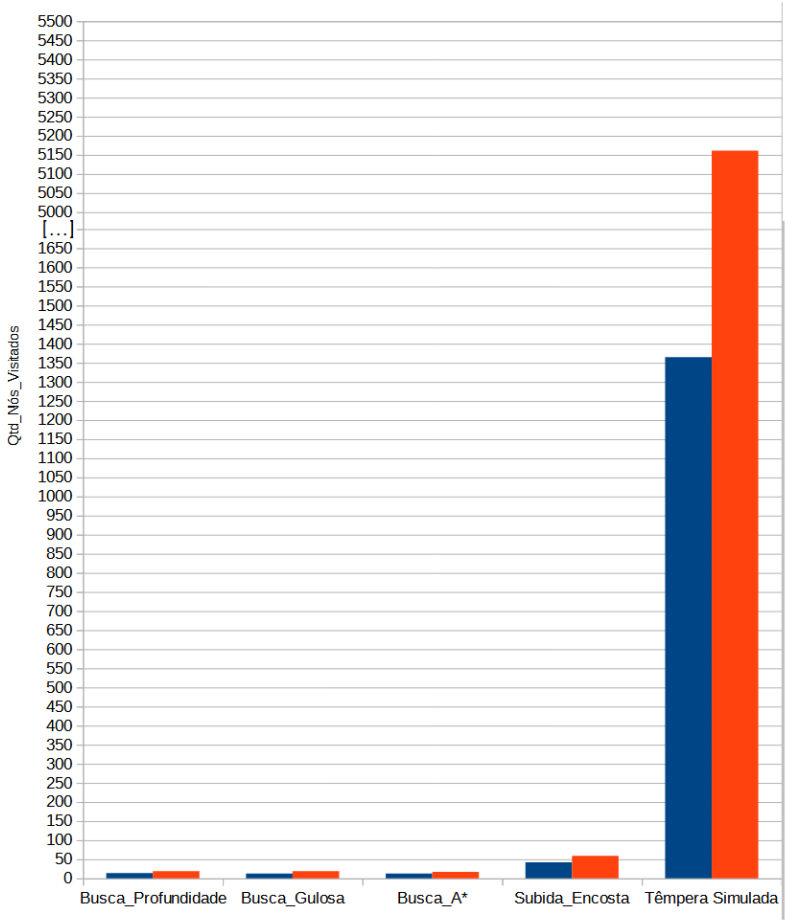
- Têmpera simulada ou Simulated Annealing: algoritmo de otimização global inspirado pelo processo físico de têmpera do metal. Usado para encontrar soluções aproximadamente ótimas em problemas de otimização combinatória, onde o objetivo é encontrar a melhor solução possível em um espaço de busca grande e complexo.Começa com uma solução inicial e, em seguida, faz uma busca no espaço de solução movendo-se para soluções vizinhas. A principal característica do algoritmo é a aceitação de movimentos que pioram a solução atual em uma certa probabilidade controlada. Essa característica é o que distingue a têmpera simulada de outros algoritmos de otimização, como o algoritmo de busca local.

- **Experimentos**

Durante o desenvolvimento do trabalho, alguns experimentos foram realizados. Com Sudoku de tamanhos variados, (por exemplo: 4x4, 5x5, 6x6, 9x9), nos modelos simples e complexo. A seguir, serão os gráficos apresentarão algumas das comparações mais relevantes:

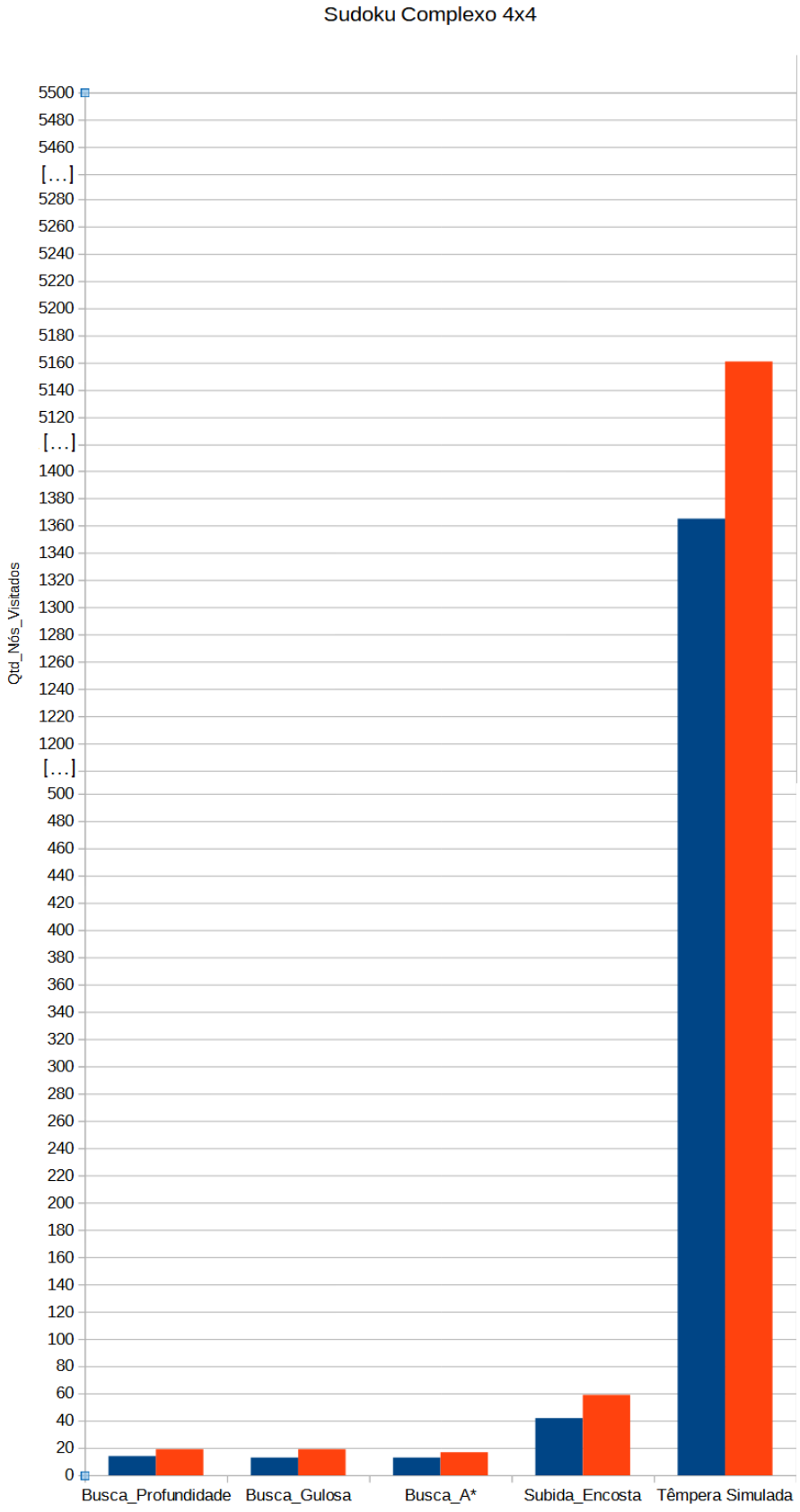
- Experimento 1 - leitura de 2 arquivos diferentes do **Sudoku Simples 4x4**

Analise- quantidade de nós visitados nos 5 algoritmos de busca



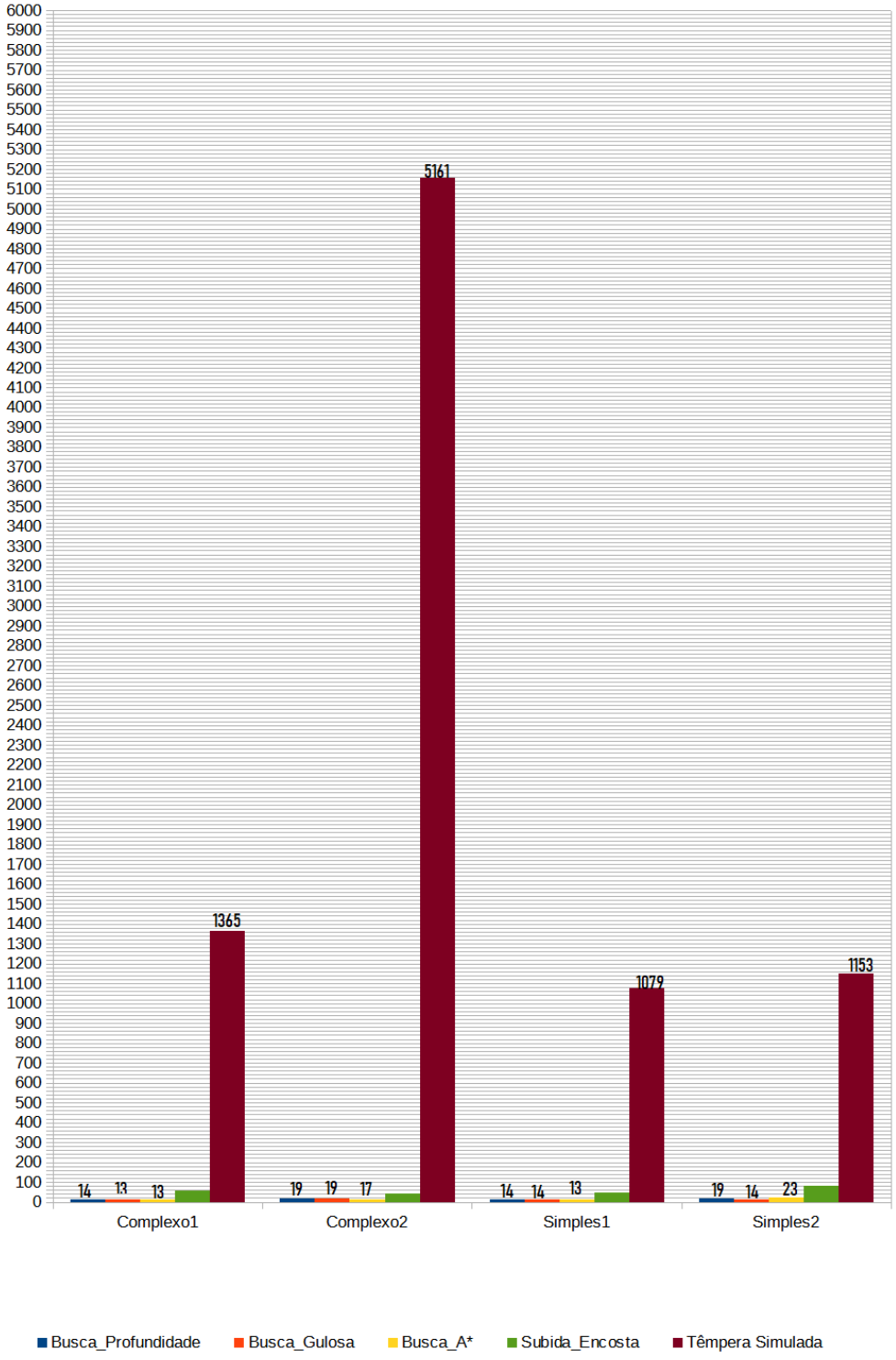
- Experimento 2- leitura de 2 arquivos diferentes do **Sudoku Complexo 4x4**

  Analise- quantidade de nós visitados nos 5 algoritmos de busca



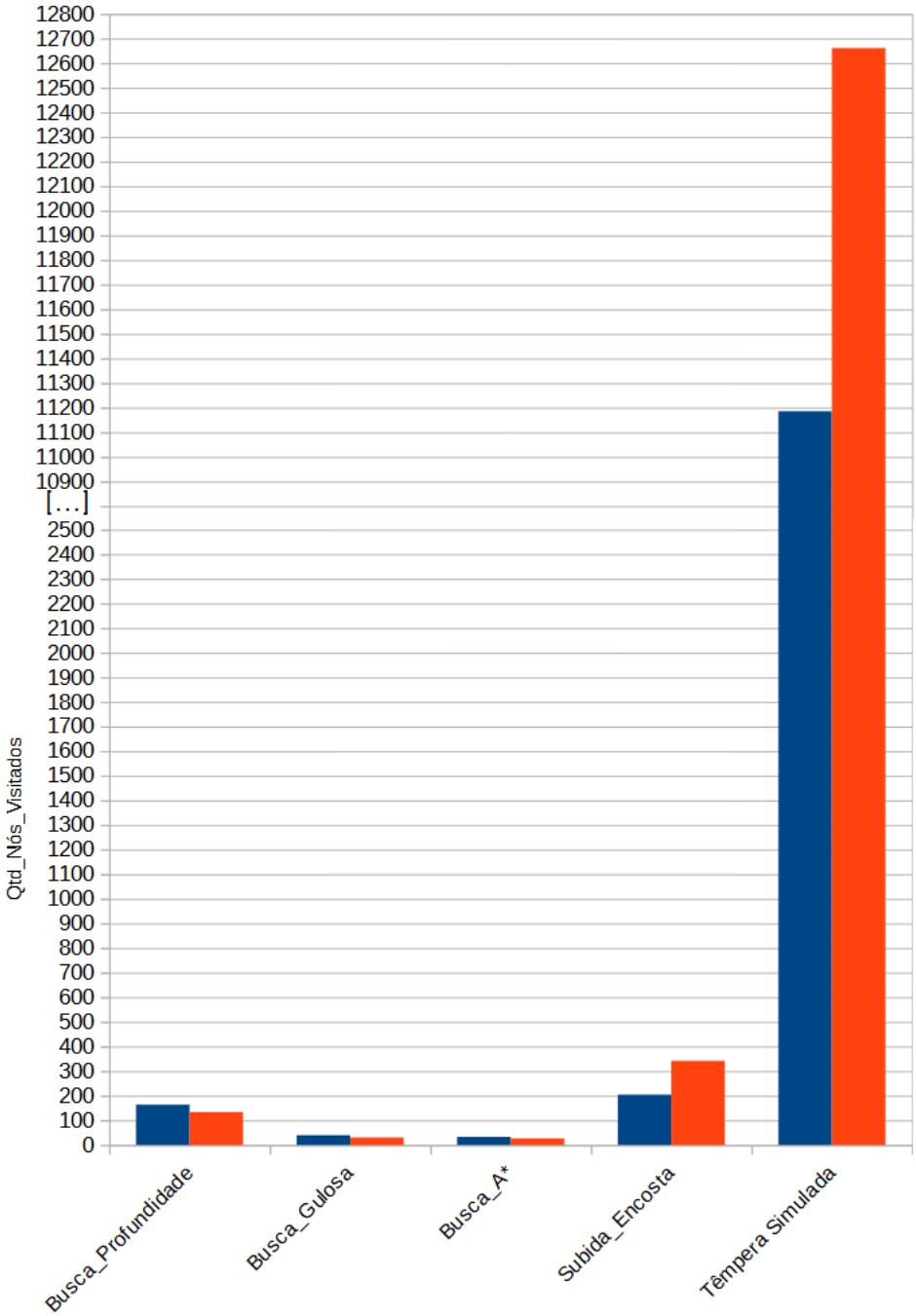
- Experimento 3- leitura de 2 arquivos diferentes do **Sudoku 4x4 Simples e Complexo.**

Analise- quantidade de nós visitados nos 5 algoritmos de busca



- Experimento 4- leitura de 2 arquivos diferentes do **Sudoku 6x6 Simples**

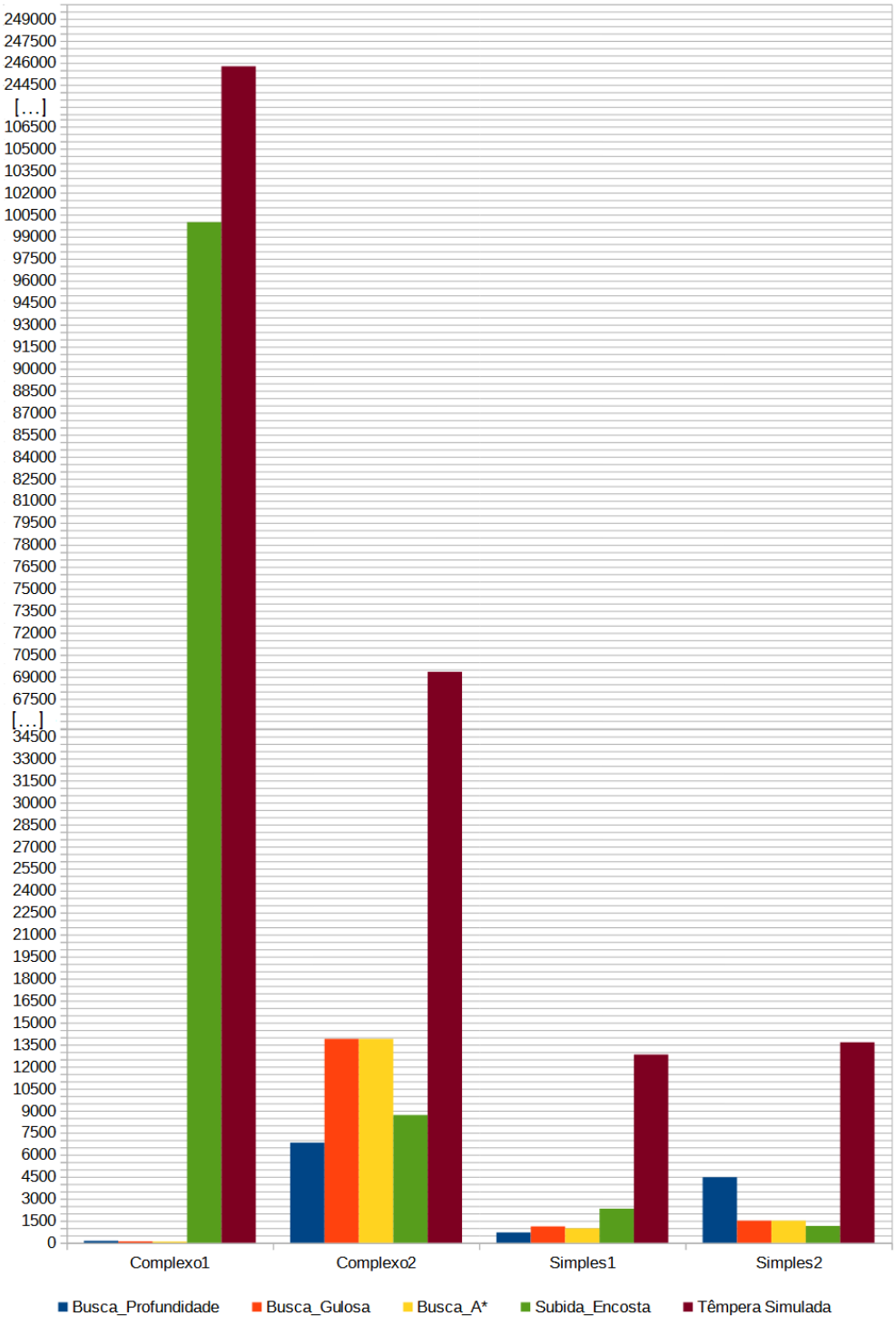
  Analise- quantidade de nós visitados nos 5 algoritmos de busca



.

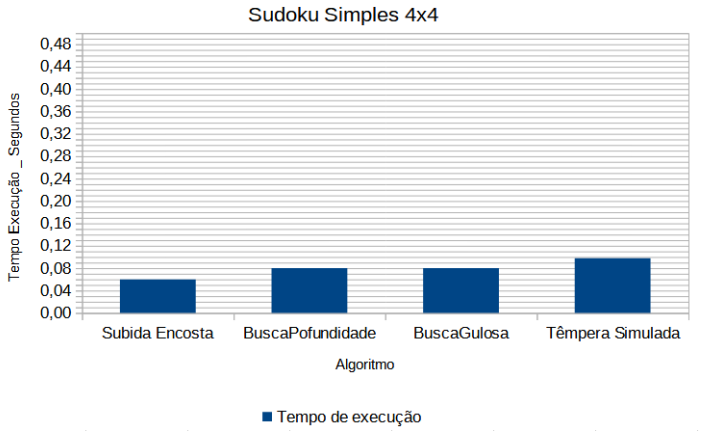
- Experimento 5- leitura de 2 arquivos diferentes do **Sudoku 9x9 Simples e Complexo.**

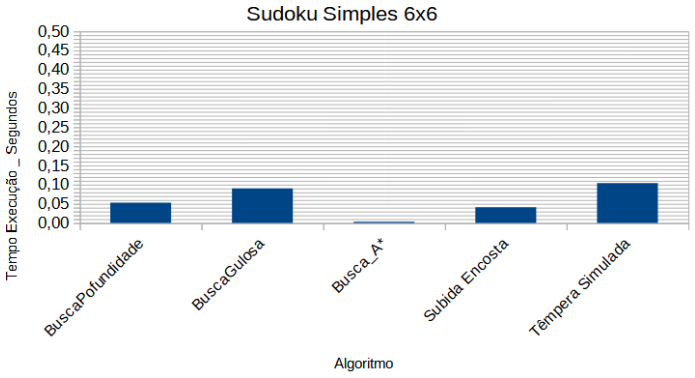
  Analise- quantidade de nós visitados nos 5 algoritmos de busca

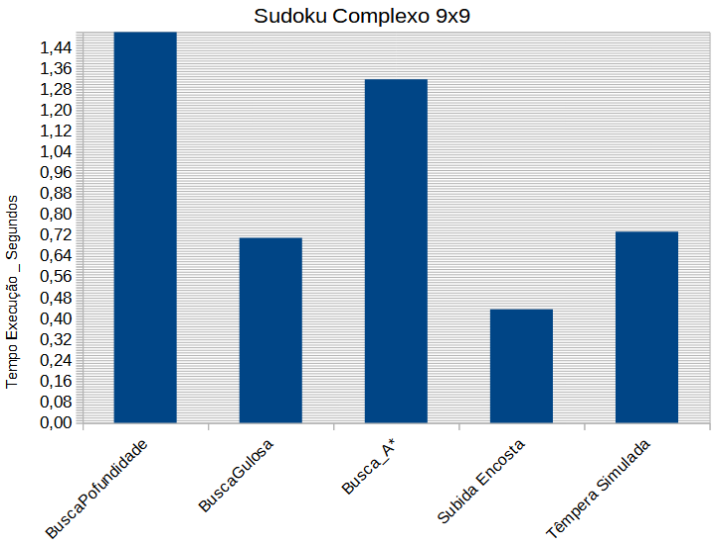


- Experimento 5- **Sudoku 4x4 Simples / Sudoku 6x6 Simples / Sudoku 9x9 Complexo.**

Analise- tempo de execução em segundos







- Considerações Finais

Sobre os algoritmos de busca desenvolvidos, para percorrimento do Sudoku, nos modelos simples e complexos, é relevante fazer as seguintes observações, sobre as dificuldades encontradas:

- Para os algoritmos de Subida de Encosta e Têmpera Simulada as dificuldades encontradas foram em relação aos movimentos aleatórios. Pois em vários casos gerava-se movimentos que não chegava a lugar algum, daí o algoritmo ficava praticamente em looping. Então tais parâmetros foram ajustados.

  No algoritmo de Subida de Encosta, limitou-se os movimentos laterais em uma quantidade máxima de movimentos possíveis, e também chegou a uma melhor solução para a troca aleatória dos pares de números dentro do board para esse algoritmo. Com isso o algoritmo foi ficando mais eficiente.

  No algoritmo Têmpera, a dificuldade foi em embaralhar bem a solução para que gerasse bons resultados. Para isso em pesquisas foi descoberto uma distribuição matemática, a distribuição de Boltzmann, na qual é feito um cálculo probabilístico com um número randômico, vê se um número random é menor que a exponencial da negativa do novo score menos o antigo score dividido pela temperatura.

- Outra dificuldade foi em ajustar os parâmetros da solução de Têmpera Simulada, saber qual seria uma melhor temperatura inicial, parâmetro de resfriamento e até mesmo a quantidade de movimentos que tentar-se-ia fazer antes de fazer um resfriamento.

- Na buscas Iterativa, Gulosa e A*, a dificuldade foi modelar o Sudoku em estados e ações. Implementar a função que calcula os valores válidos para células vazias também foi um desafio. A expansão dos nós de um estado também foi difícil de implementar. Além disso a busca Iterativa estava demorando muito para executar alguns tabuleiros mais complexos, mas isso por causa de sua característica de sempre reiniciar toda a busca incrementando em 1 a profundidade.

- Para a busca Gulosa as maiores dificuldades foram achar uma heurística admissível e controlar o array que armazena os estados estendidos, levando em consideração o valor da heurística dos estados. Foi necessário desenvolver uma espécie de "fila de prioridade", após implementar o A* notou-se que o Java já possui uma estrutura de dados de fila de prioridade.

- Para a busca A* as maiores dificuldades foram achar uma boa função custo e também controlar a fila de prioridade que armazena os estados estendidos, levando em consideração a soma do custo e da heurística dos estados.

Estas foram as principais observações, relacionadas as dificuldades encontradas durante o desenvolvimento dos algoritmos de buscas propostos.