UNIVERSIDADE FEDERAL DE MINAS GERAIS

DCC | SISTEMAS DE INFORMAÇÃO | ALGORITMOS I

DOCUMENTAÇÃO
TRABALHO PRÁTICO 3 | SUDOKU
PROBLEMA NP-COMPLETO

Rômulo Rafael da Silva, 2012055308

Novembro / 2019

Sumário

1. DESCRIÇÃO DO PROBLEMA

Esta documentação é referente ao trabalho prático 3 da disciplina de Algoritmos 1 do curso de Sistemas de Informação da UFMG. A motivação do trabalho é o puzzle de lógica: Sudoku, jogo composto por um "quadrante numérico" de linhas e colunas com algumas células pré-preenchidas. A resolução consiste em preencher cada célula de modo a não haver números repetidos em cada linha, coluna ou subquadrante.

Sudoku é um arranjo de células dispostos em uma grade n^2 x n^2 particionadas em regiões de células dispostas em uma subgrade n x n, que possuem como objetivo o preenchimento de todas as células com elementos de um conjunto de símbolos, de tal forma que em cada linha, coluna e região contenham somente um elemento do conjunto.

A proposta do trabalho apresentava o jogo Sudoku como um caso especial de Quadrados Latinos e que pode ser transformado em um problema de Coloração de Grafos. Este é um problema NP-completo. Para resolver uma instância Sudoku como uma Coloração de Grafos, faz-se sua representação como um grafo: cada célula Sudoku é um vértice no grafo; para cada vértice de célula, adicionamos as arestas entre tal vértice todos os outros vértices que a célula possui uma relação de conflito: linha, coluna e quadrante. As instâncias de problemas no trabalho compreendem Sudoku clássico e Sudoku com versões que não são quadrado perfeito como, por exemplo, 8 x 8 onde cada quadrante é da forma 4 x 2.

O objetivo do trabalho é desenvolver, a partir de uma instância de tamanho n x n, uma solução algorítmica que informe se o Sudoku possui ou não solução e apresentar a solução. As seções seguintes estão assim estruturadas:

- descrições técnicas sobre hardwares, softwares e linguagem de programação utilizados ao longo do desenvolvimento do trabalho;
- estruturas de dados e dos algoritmos implementados como parte da solução do problema;
- apresentação das análises de complexidade, espaço e tempo, da solução;
- avaliação experimental de testes utilizando os datasets disponibilizados pelos monitores da disciplina e outros criados.

2. DESCRIÇÕES TÉCNICAS

2.1. HARDWARE

O desenvolvimento e testes do programa foram em um dispositivo cujas configurações são: (1) Intel® Core™ i5-7200U CPU 2.50GHz 2.71GHz, (2) 8GB de memória RAM, (3) Sistema Operacional Windows Pro 10 x64 com Linux Ubuntu rodando como subsistema.

2.2. SOFTWARE

Os programas utilizados foram: (1) Visual Studio Code, editor de código-fonte desenvolvido pela Microsoft. Para criar a estrutura do projeto do programa, utilizou-se a extensão do VS (2) Easy C++ projects.

2.3. LINGUAGEM DE PROGRAMAÇÃO

Programa desenvolvido em linguagem C++, com utilização somente das bibliotecas padrões disponíveis até a versão C++17 (2014). Não são necessárias bibliotecas adicionais, de terceiros e/ou que exijam a instalação para compilação e execução do programa.

3. ESTRUTURAS DE PROJETO E ALGORITMOS

3.1. DIRETÓRIO DO PROJETO

A estrutura dos projetos provê:

- dataset: estão os arquivos de entrada (subpasta input) disponibilizados no ambiente Moodle da disciplina e arquivos de saída (subpasta ouput);
- docs: constam os arquivos do tipo documento (.pdf, .doc etc.) disponibilizado no ambiente Moodle da disciplina e produzidos ao longo do trabalho;
- src: constam todos os arquivos de header (.h) e de implementação/corpo (.cpp), que possuem o código dos métodos assinados em .h, bem como o arquivo main.cpp e um makefile;
- pasta raiz: contém os outros arquivos como, por exemplo, arquivos do GitHub (.gitattributes, gitignore e README.md)

3.2. MODELOS TEÓRICOS

Como parte da solução deste problema, recorreu-se ao entendimento dos seguintes modelos teóricos:

MODELO	DESCRIÇÃO (ALTO NÍVEL)
Quadrado Latino	Um quadrado latino de ordem n é uma quádrupla (R, C, S, L) onde R, C, S são conjuntos de cardinalidade n e L é uma aplicação L : R x C → S tal que para cada i de R e j de C, a equação L(i,j) = x tem uma única solução, isto é, fixando quaisquer duas coordenadas de (i, j, x) encontramos a terceira de forma única. Em outras palavras, um quadrado latino é um arranjo n x n onde em uma determinada linha i e coluna j, o elemento i * j não se repete nesta mesma linha e mesma coluna.
Grafo Simples	Para modelar o Sudoku em um grafo, podemos pensar em cada quadrado da grade como um vértice, onde dois vértices u e v são adjacentes se (1) u e v pertencem a mesma linha, (2) u e v pertencem a mesma coluna ou (3) u e v pertencem ao mesmo bloco. Portanto, como os vértices são adjacentes, existe uma aresta ligando cada vértice aos demais contidos no bloco, na coluna e na linha.
Coloração de grafos	Coloração de grafo é um caso de rotulagem de grafo, onde há atribuição de rótulos "cores" a elementos de um grafo sujeito a certas restrições. De forma simples, é uma forma de colorir os vértices de um grafo tal que não haja dois vértices adjacentes que compartilhem a mesma cor. Seja G(V, E) um grafo, onde V é o conjunto de vértices e E o conjunto de arestas. Uma coloração para o grafo G é uma atribuição de cores para cada vértice de forma que vértices adjacentes tenham diferentes cores. Formalmente, uma coloração consiste em função c : V(G) → N tal que c(U) != c(v), se u e v são vértices adjacentes. O problema é NP-Completo.

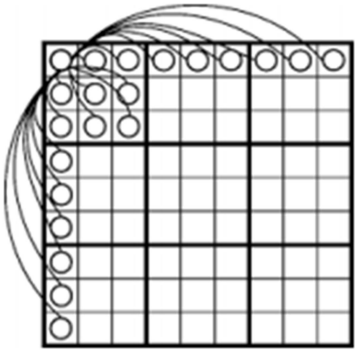


Figura: Modelagem de um Sudoku 9 x 9 em um grafo simples

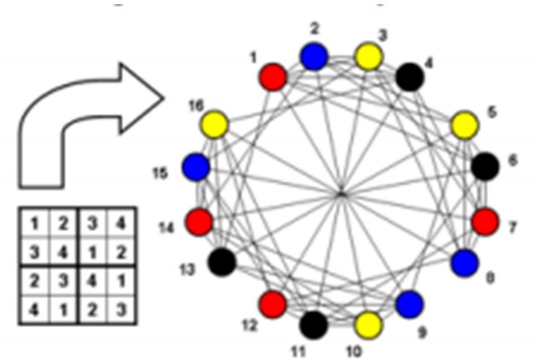


Figura: Modelagem de um Sudoku 4 x 4 em um grafo.

3.3. SOLUÇÕES ALGORÍTMICAS

A literatura possui algumas soluções algorítmicas para a resolução do Sudoku. Dentre essas soluções, o algoritmo implementado neste trabalho considerou:

ALGORITMO	DESCRIÇÃO (ALTO NÍVEL)
Backtracking	Algoritmo recursivo que gera uma árvore com todas as soluções possíveis da instância de entrada, de maneira que nós folha representam soluções ótimas e nós internos representam soluções parciais. A árvore de soluções é percorrida por busca em profundidade – passo de ida (forward) – e quando uma solução completa é encontrada ou há falha na referida solução parcial, retorna-se a um ponto da árvore – passo de volta (backward) – onde seja possível encontrar soluções alternativas a partir dele.
Branck-and-Bound	Algoritmo que explorar o espaço de soluções de maneira inteligente, ou seja, evitando que todas as possíveis soluções sejam percorridas. De maneira geral, o algoritmo também gera uma árvore de busca onde cada nó é uma solução parcial da instância de entrada. Isto é, partindo de uma instância inicial, esta é sucessivamente particionada em instância menores. Esta é a etapa de ramificação (branching). Em cada nó, tem-se a etapa de bounding, onde calcula-se um limitante inferior, que é menor ou igual ao valor de uma solução ótima. Se este valor for maior ou igual do que o limitante superior do valor da solução ótima global, então o referido nó pode ser descartado. Na etapa de branching pode ser

	aplicado um algoritmo de backtracking.

3.4. OVERVIEW DA SOLUÇÃO IMPLEMENTADA

A solução é um algoritmo de coloração de grafo usando backtracking. Este algoritmo busca essencialmente por uma k-coloração, onde k neste caso será a ordem do Sudoku ao qual o algoritmo está sendo aplicado. A cada iteração, o algoritmo verifica se é possível colorir aquele vértice com uma cor existente; caso contrário ele cria uma nova cor, e assim sucessivamente até que o número de cores seja igual a ordem do Sudoku o qual o algoritmo foi aplicado.

A implementação considerou dois algoritmos que utilizam as soluções algorítmicas para o problema de Coloração de Grafo: Algoritmo de Brown que propõe um algoritmo que encontra uma coloração exata em um grafo por meio do backtracking; e o Algoritmo DSATUR que é uma heurística gulosa para coloração de vértices, introduzindo o conceito de grau de saturação de um vértice como a quantidade de cores distintas atribuídas aos vizinhos coloridos de um vértice de um grafo G. Por questão de espaço, orienta-se uma leitura do trabalho "Algoritmos Exatos para o Problema da Coloração de Grafos" disponível aqui.

4. ANÁLISE DE COMPLEXIDADE

4.1. NP-Completo

Apresenta-se uma prova NP-completo do Sudoku. Considera-se a versão de um problema de decisão, ou seja, significa verificar se existe ou não uma solução válida para uma dada instância do Sudoku.

Um algoritmo de reconhecimento de uma solução do Sudoku está em tempo $O(n^2)$ – isto é, linear no número de células do tabuleiro. Para a prova da NP-dificuldade, é realizada uma redução polinomial do Quadrado Latino (NP-Completo) para o Sudoku. Outros problemas NP-Completos também podem ser reduzidos polinomialmente ao Sudoku. Dentre eles: Unique SAT, Pré-Coloração Estendida e Cobertura Exata.

Dado um Quadrado Latino k x k parcialmente preenchido, obtemos uma instância do Sudoku n x n, para $n = k^2$, da seguinte maneira. Seja s(i, j) o valor atribuído à célula do Sudoku da linha i e coluna j, e seja l(p, q) o valor contido na célula do Quadrado Latino da linha p e coluna q.

$$s(i,j) = \begin{cases} \ell(i, j/m) \cdot n & ((i,j) \in B, \ell(i, j/m) \neq \varnothing) \\ \varnothing & ((i,j) \in B, \ell(i, j/m) = \varnothing) \\ ((i \bmod n)n + \lfloor i/m \rfloor + j) \bmod n & \text{caso contrário} \end{cases}$$

onde

$$B = \{(i,j) | \lfloor i/m \rfloor = 0 \text{ e } (j \bmod n) = 0\}.$$

Figura: equação de recorrência do Sudoku

O conjunto B contém todas as células do Sudoku que serão correlacionadas com o Quadrado Latino. O símbolo ∅ representa uma célula não preenchida. Abaixo, o resultado de uma transformação e a relação de validade entre as instâncias do Sudoku e do Quadrado Latino: se o Sudoku tiver solução válida, então o Quadrado Latino também terá e vice-versa.

Figura: Algoritmo de Mapeamento de uma instância do Quadrado Latino para uma instância do Sudoku

4.1. STRUCT E CLASSES

A solução implementa uma struct e uma classe. A seguir, um resumo do "tamanho" destes objetos:

Classe	Nome	Tipo	Atributos	Métodos		
				public	protected	private
Grafo.h	Vértice	struct	4	-	-	-
Grafo.h	Grafo	class	-	17	8	1

4.2. TEMPO

Para o pior caso, o algoritmo apresenta **$O(V^{v-k})$, com k sendo o número de vértices já coloridos**. Percebe-se, portanto, que dependendo da quantidade de vértices coloridos a priori, ou a quantidade de células fixas que já possuem números preenchidos, o algoritmo é sobre polinomial.

4.3. ESPAÇO

Quanto ao uso de memória, a heurística salva o resultado a um custo espacial de **O(V-k)** na memória. Ao observar o espaço gasto pelo grafo para salvar as relações de adjacência, que são realizadas por meio de uma lista de adjacência, têm-se que a memória total ocupada é de **O(V+E)**.

5. AVALIAÇÃO EXPERIMENTAL

Os testes compreenderam a execução de **21 arquivos**. Os testes foram feitos num **total de 10 execuções simultâneas** com **entradas com diferentes tamanhos** com objetivo de verificar o desempenho da solução implementada.

5.1. MÉDIA E DESVIO PADRÃO

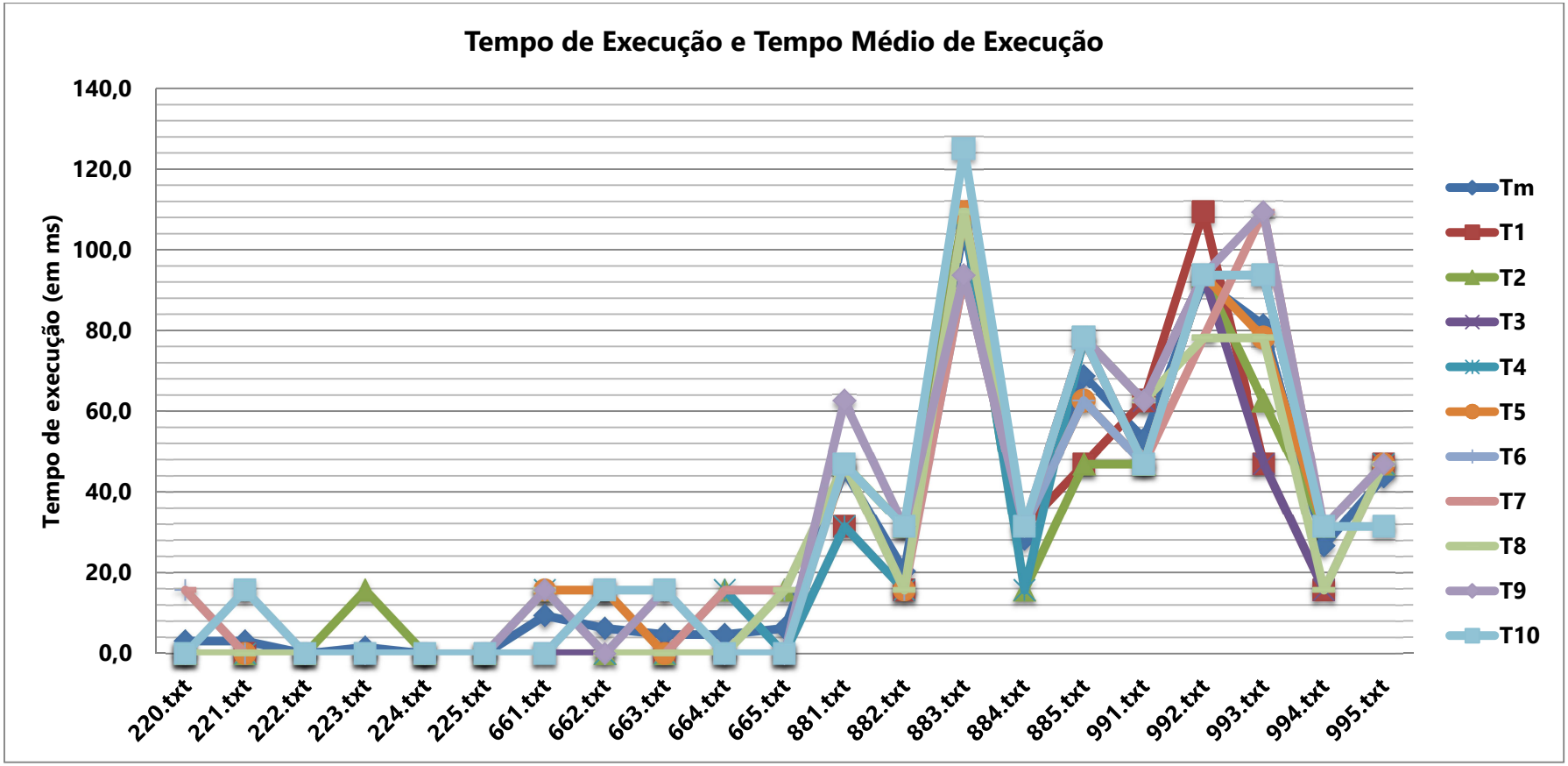
5.1.1. TABELA DE MÉDIAS

A tabela abaixo apresenta a média (em ms) de 10 execuções consecutivas de cada arquivo de entrada:

220.txt	**221.txt**	**222.txt**	**223.txt**	**224.txt**	**225.txt**	**661.txt**	**662.txt**	**663.txt**	**664.txt**
3,125	3,125	0,000	1,563	0	0	9,375	6,25	4,688	4,688
665.txt	**881.txt**	**882.txt**	**883.txt**	**884.txt**	**885.txt**	**991.txt**	**992.txt**	**993.txt**	**994.txt**
6,25	45,313	20,313	106,25	28,125	68,750	53,125	92,188	81,250	26,563
995.txt									
43,75									
							Média Global	**28,795**	

5.1.2. GRÁFICOS DE MÉDIAS

O valor "Tm" representa o **tempo médio** de todas as **10 execuções simultâneas realizadas para um mesmo arquivo de entrada.** Graficamente, é possível identificar quais arquivos consumiram maior tempo médio de execução. Destacam-se os arquivos: **881.txt, 882.txt, 883.txt, 884.txt, 885.txt, 991.txt, 992.txt, 993.txt, 994.txt e 995.txt.**

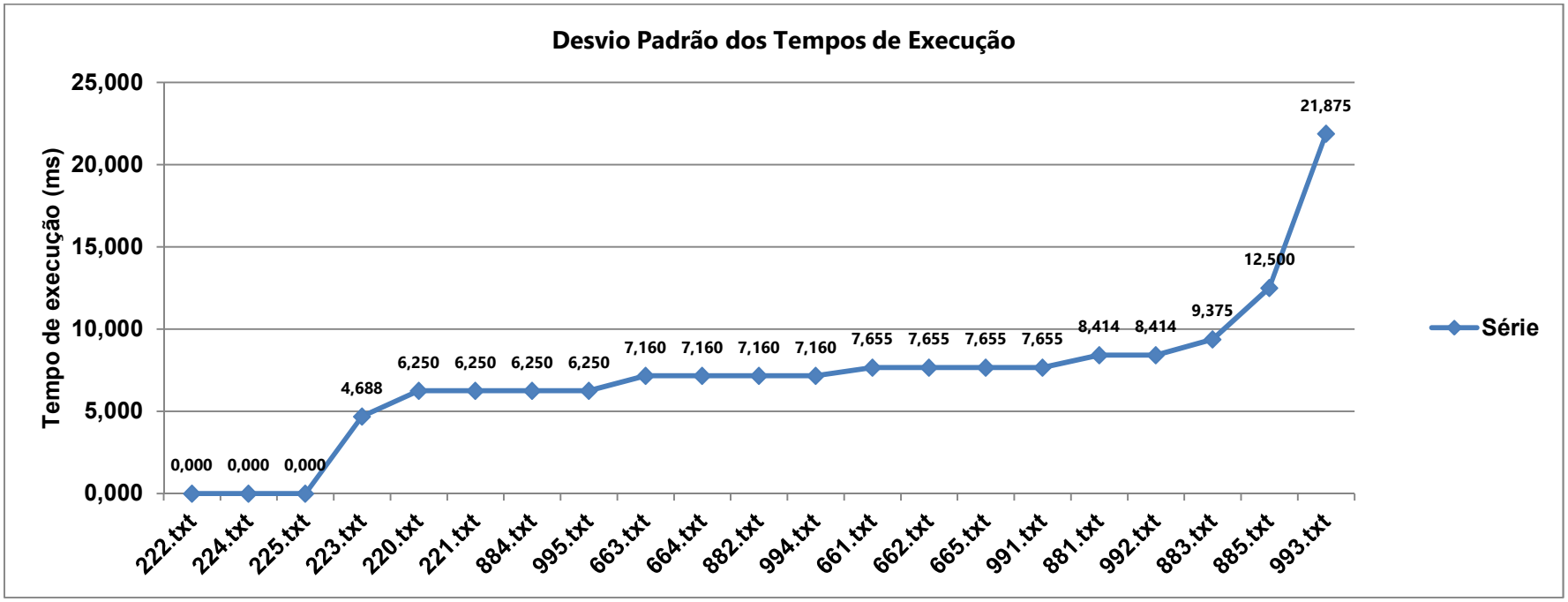

Tempo de Execução e Tempo Médio de Execução
Tempo de execução (em ms)
140,0
120,0
100,0
80,0
60,0
40,0
20,0
0,0
220.txt
221.txt
222.txt
223.txt
224.txt
225.txt
661.txt
662.txt
663.txt
664.txt
665.txt
881.txt
882.txt
883.txt
884.txt
885.txt
991.txt
992.txt
993.txt
994.txt
995.txt
Tm
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10

5.1.3. TABELA DE DESVIOS PADRÕES

A tabela abaixo apresenta os desvios padrões das 10 execuções de cada arquivo de entrada:

220.txt	**221.txt**	**222.txt**	**223.txt**	**224.txt**	**225.txt**	**661.txt**	**662.txt**	**663.txt**	**664.txt**
6,250	6,250	0,000	4,688	0,000	0,000	7,655	7,655	7,160	7,160
665.txt	**881.txt**	**882.txt**	**883.txt**	**884.txt**	**885.txt**	**991.txt**	**992.txt**	**993.txt**	**994.txt**
7,655	8,414	7,160	9,375	6,250	12,500	7,655	8,414	21,875	7,160
995.txt									
6,250									

5.1.4. GRÁFICOS DE DESVIOS PADRÕES



5.1. BREVE DISCUSSÃO DOS RESULTADOS

A solução implementada, em geral, apresentou bom desempenho para os diferentes tamanhos de entradas do Sudoku. Pelo último gráfico, podemos ver pelo Desvio Padrão dos tempos de execução das entradas que a maioria dos arquivos apresentam valores relativamente próximos, excetuando-se os 3 primeiros arquivos (222.txt, 224.txt e 225.txt) e os 2 últimos (885.txt e 993.txt).

6. CONSIDERAÇÕES FINAIS

Pode se concluir que o Sudoku é um problema NP-Completo, e desta forma ele possui algoritmos em força bruta que o solucionam em tempo polinomial. De forma a desenvolver algoritmos com uma melhor performance, a modelagem em forma de grafos é de extrema ajuda e permite que o problema seja resolvido com coloração em grafos.

Neste trabalho foi utilizado um algoritmo cuja solução é baseada em backtracking de coloração de grafo. O algoritmo possui uma boa performance, entretanto a literatura apresenta outras heurísticas que podem propiciar melhorias sobre a solução aqui desenvolvida. Algumas heurísticas se concentram na escolha da ordem dos vértices a serem coloridos, pois uma alteração na ordem pode resultar numa redução na quantidade de cores erroneamente atribuídas pelo algoritmo, o que acelera o processamento.

O padrão de tabuleiros do Sudoku é 9 x 9, considerando o limite superior das entradas do problema. Porém, é conhecido outras configurações como 16 x 16, 25 x 25 etc. O algoritmo implementado pode ser adaptado para outros tamanhos de tabuleiro com baixa dificuldade, bem como adaptar outras formas de ordenação de coloração de vértices ou utilizar outras técnicas de coloração de grafos, a fim de trazer melhor desempenho temporal.

7. BIBLIOGRAFIA

1. Quadrados Latinos. Disponível em: <https://www.ime.unicamp.br/~ftorres/ENSINO/MONOGRAFIAS/EA_2011_MONOI.pdf> Acesso em: 23 nov 2019;
2. Coloração de grafos. Disponível em: <https://pt.wikipedia.org/wiki/Colora%C3%A7%C3%A3o_de_grafos> Acesso em: 23 nov 2019;
3. Algoritmos Exatos para o Problema da Coloração de Grafos. Disponível em: <https://pt.wikipedia.org/wiki/Colora%C3%A7%C3%A3o_de_grafos> Acesso em: 23 nov 2019;
4. Coloração de Grafos Aplicado na resolução do Sudoku. Disponivel em: <http://ceur-ws.org/Vol-1754/EPoGames_2016_AC_paper_2.pdf> Acesso em: 23 nov 2019;
5. Sudoku as a Constraint Problem. Disponível em: <https://www.inf.tu-dresden.de/content/institutes/ki/cl/study/winter06/fcp/fcp/sudoku.pdf> Aceso em: 23 nov 2019;
6. O jogo de lógica Sudoku: modelagem teórica, NP-completude, algoritmos e heurísticas. Disponível em: <https://vigusmao.github.io/manuscripts/sudoku.pdf> Acesso em: 15 nov 2019;
7. Exemplo solução sudoku 4x4. Disponível em: ambiente Moodle da disciplina de Algoritmos 1 de Sistemas de Informação, semestre 2019/2. Acesso em: 15 nov 2019;