Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SCC218 – Algoritmos Avançados e Aplicações
Professor Gustavo Batista

Relatório do projeto 1 - Coloração de mapas com *backtracking* e heurísticas de poda

Matheus de França Cabrini - NºUSP: 8937375

I) Implementação

Dado um mapa com *n* regiões, como podemos pintar cada uma delas, utilizando-se 4 cores, de modo que nenhuma região tenha cor igual aos seus vizinhos (regiões adjacentes)? A solução implementada para tal problema foi a técnica de força bruta com *backtracking* e possibilidade para heurísticas de poda, na linguagem Java.

Em termos práticos, utilizamos a representação em um grafo de restrições, com matriz de adjacências, onde os nós são as regiões e arestas indicam regiões vizinhas. O algoritmo, sem as heurísticas, resume-se a pintar uma dada região com alguma cor, e pintar a próxima região de modo que não conflite com as cores já atribuídas aos seus vizinhos. Assim, podemos definir o procedimento recursivamente. A recursão irá terminar quando todas as regiões forem coloridas. Caso seja atingida alguma região a qual não aceita mais nenhuma cor, ocorre o chamado *backtracking*: voltamos à região anterior na árvore de recursão e tentamos pintá-la de outra cor.

Com as heurísticas, o funcionamento torna-se diferente em certos aspectos. Primeiramente, temos a heurística de verificação adiante, a qual consiste em manter e atualizar, ao longo do algoritmo, o domínio de cores ainda válidas para cada região. A cada atribuição de cor ocorrida durante o algoritmo, elimina-se tal cor do domínio das regiões adjacentes. Logo, podemos ir verificando se alguma região ficou sem cor válida antes mesmo de tentar colorí-la. Isso causa uma poda nos casos de coloração na árvore recursiva.

A próxima heurística implementada foi a de mínimos valores remanescentes (MVR). Ela afeta a escolha da próxima região a ser colorida, que antes era simplesmente a região de índice subsequente no grafo. Com o MVR, utilizamos os domínios estabelecidos devido à verificação adiante para obter a região com menor quantidade de cores ainda possíveis, sendo essa a região escolhida para se tentar a próxima coloração durante o algoritmo. Para se obter tais resultados, mantemos as regiões em uma fila de prioridades, cuja função comparadora prioriza regiões com menor quantidade de cores ainda válidos em seu domínio. É válido notar que, quando o domínio de alguma região é atualizado, devemos reinserir tal região na fila a fim de atualizá-la também.

A última heurística empregada foi a de grau, ou variável mais restritiva, que serve como método de desempate caso haja mais de uma possível região com mínimas cores remanescentes. No caso, a região escolhida será a de maior grau, ou seja, a com maior número de regiões vizinhas a ela.

III) Modo de execução

Para executar o programa, deve-se clicar no arquivo JAR ou entrar na linha de comando: "*java -jar backtracking.jar*". O programa foi desenvolvido utilizando-se a versão 8.60 do Java, logo, recomenda-se a instalação dessa versão ou superior. O programa não recebe parâmetros e o formato de entrada das regiões e modo de algoritmo devem ser conforme especificado no projeto.

III) Resultados e análise

Mapa de entrada	**Total de atribuições de cor**				**Tempo de execução médio (ms)**			
	a	**b**	**c**	**d**	**a**	**b**	**c**	**d**
Brasil (27 regiões)	27	27	27	27	0.110	0.626	2.253	2.277
EUA (51 regiões)	3616	560158	51	51	6.613	371.938	3.248	3.868
Europa (41 regiões)	43	43	41	41	0.186	0.982	3.057	3.779

Tabela 1: Relação entre os mapas de entrada com o total de atribuições de cor e o tempo de execução médio referentes à coloração.

Na tabela 1, podemos ver a performance do algoritmo em três mapas diferentes, os quais foram coloridos em um de quatro modos: "a", para o algoritmo normal; "b", com verificação adiante; "c", com verificação e MVR; e "d", com verificação, MVR e grau. Para cada dupla de mapa e modo, o algoritmo foi executado no mìnimo cinco vezes. Assim, o tempo de execução tabelado é proveniente da média entre os tempos de cada uma dessas execuções. O número de atribuições não variou entre as execuções.

Vê-se claramente que, quanto maior o número de regiões no mapa, maior é, em geral, o número de atribuições de cor e também o tempo de execução médio..

O número de regiões também influencia na eficácia ou não das heurísticas. Por exemplo, o modo "a" e "b" não renderam bons resultados para o mapa dos EUA, porém foram os melhores modos para os mapas de menor tamanho. Isso ocorre, em parte, devido ao *overhead* das heurísticas em "c" e "d", que acabam não compensando para mapas de menor tamanho. Portanto, o uso das heurísticas vale mais a pena quanto maior for o número de regiões do mapa.