Universidade de São Paulo
SCC - 230 Inteligência Artificial
Profa. Solange Oliveira Rezende

Métodos de Busca:
Coloração de Mapas

8598861 - Bernardo Simões Lage G. Duarte
8122585 - Eder Rosati Ribeiro
8936993 - Gabriel Luiz Ferraz Souto
8936648 - Giovani Ortolani Barbosa
8531887 - Giovanni Robira
8937271 - Rafael Bueno da Silva

São Carlos - SP
Outubro de 2017

1) Introdução

Em Inteligência Artificial, muitos problemas podem ser modelados em um espaço de estados, e resolvidos como um problema de busca. Técnicas diferentes de busca podem resolver um problema variando em eficiência, e isso depende das características do problema, ou seja, uma técnica pode ser melhor para um dado problema, porém ruim para outro problema.

O objetivo do trabalho é, usando uma abordagem de Inteligência Artificial, mostrar soluções para o problema de coloração de mapas usando metodologias de Busca Cega e Busca Informada, fazendo uma escolha de metodologia de busca que seja melhor para o problema em questão.

2) Descrição do Problema

O problema de coloração de mapas é uma abstração do problema de coloração de grafos, onde cada região do mapa representa um nó e suas fronteiras representam arestas. O problema consiste em, dado um grafo que representa um mapa, atribuir uma cor à todos os nós sem que os vizinhos tenham a mesma cor.

A quantidade de cores usadas no problema deve ser a mínima suficiente para que exista uma solução. A descoberta dessa quantidade poderia ser assunto de outro problema de busca, e nos exemplos desse trabalho partiremos do princípio que esse valor já é conhecido. No nosso caso, utilizaremos quatro cores distintas, pois com tal número é possível colorir qualquer mapa planar (Appel e Haken, 1976).

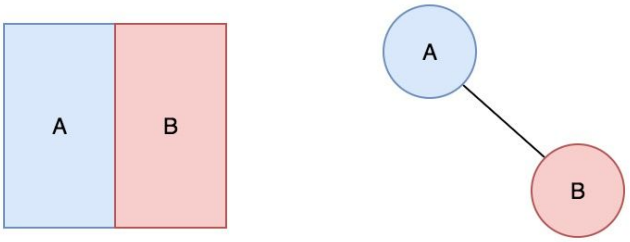

Imagem 1 - Exemplo de região, sua representação em grafo, e uma possível solução do problema

Uma particularidade do problema de coloração de mapas em relação à outros problemas visto em sala é o fato de o problema possuir várias soluções e nenhuma solução é melhor do que a outra. Portanto, a modelagem de estados não possui um estado final específico, e sim vários estados finais que obedecem às restrições do problema.

3) Modelagem de estados

A modelagem de estados do problema é feita de forma que cada estado representa a coloração de uma região com uma cor. O estado inicial é o mapa com as regiões definidas,

sem nenhuma coloração; um estado final é algum estado onde todas as regiões estejam coloridas e as restrições do problema sejam respeitadas.

A Imagem 2 é um exemplo da coloração de um mapa simples com apenas 3 regiões (A, B e C) e a utilização de 2 cores para fazer a coloração. O exemplo é simples mas suas propriedades podem ser generalizadas em exemplos maiores.

A partir da modelagem da Imagem 2 podemos observar, completando a propriedade citada anteriormente, que independente da primeira escolha de coloração que é feita, poderemos chegar a pelo menos uma solução para o problema.

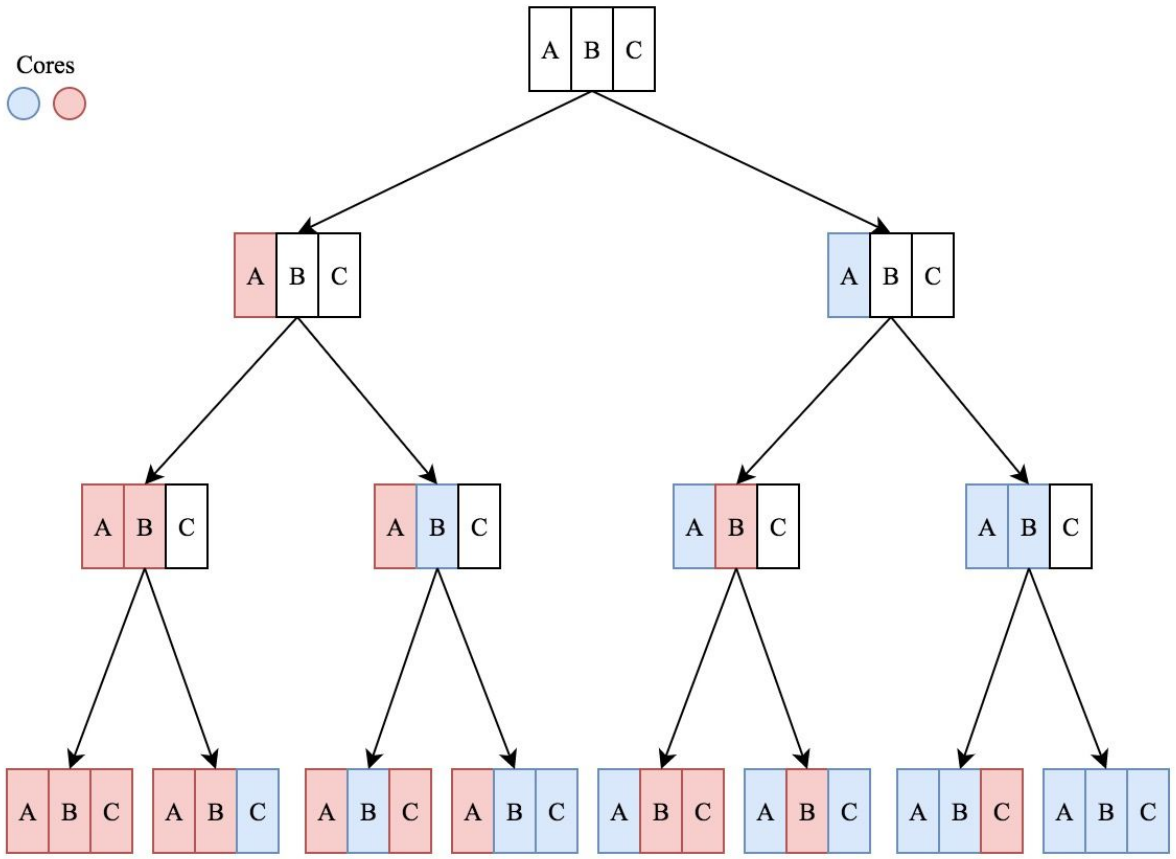

Imagem 2 - Exemplo modelagem de estados

4) Busca Cega

A definição do problema e a modelagem dos estados são capazes de mostrar que todos os estados finais (ou soluções) estão no maior nível da árvore. Além disso, sendo *n* o número de cores, cada estado possui *n* filhos (i.e. a coloração de um próximo nó com cada cor possível) fazendo com que a árvore possa crescer muito rápido horizontalmente. Como descrito na seção anterior, não seria necessário visitar a árvore inteira para alcançar uma solução, apenas uma de suas subárvores é suficiente.

A metodologia de busca cega utilizada para resolver o problema de coloração de mapas é, portanto, busca em profundidade com backtracking. A busca em largura tornaria

necessário visitar todos os nós em largura dos níveis acima do último, e já vimos que isso não é necessário; enquanto o Backtracking encontraria o resultado antes de voltar no estado inicial, ou em outras palavras, visitando no máximo metade da árvore de estados.

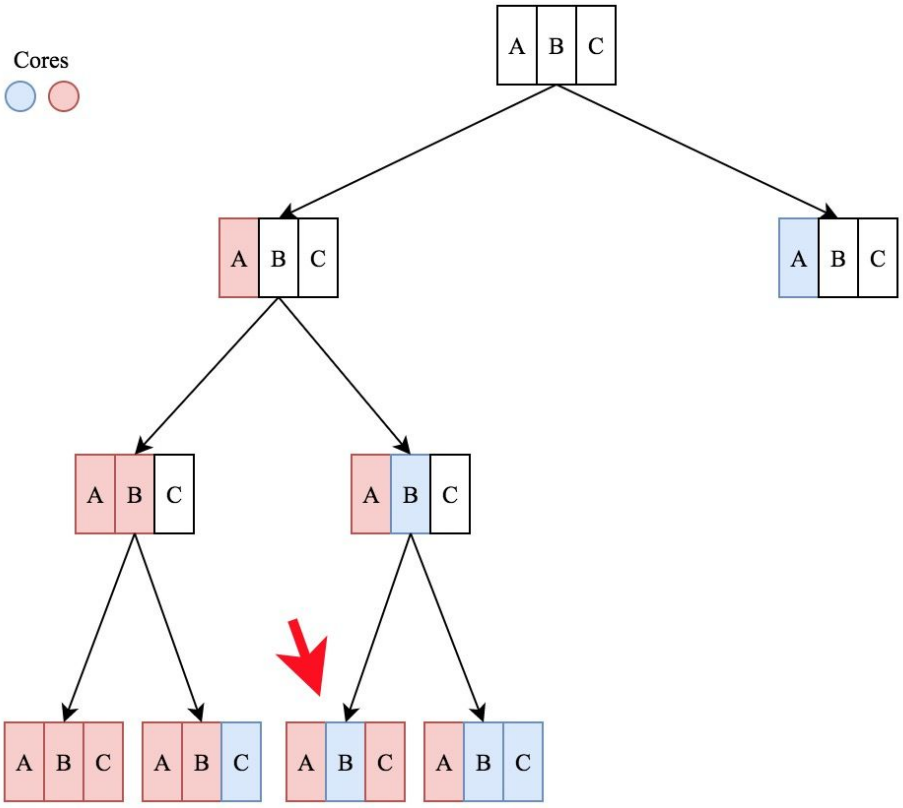

Imagem 3 - Backtracking sobre o mapa do exemplo anterior

5) Busca Informada

Tendo como base a Busca Cega citada no tópico anterior, demos um passo adiante e implementamos outras heurísticas (Verificação Adiante, Mínimos Valores Remanescentes (MVR) e variável mais restritiva para o desempate do MVR). Com isso, ao modelar o problema percebemos que o algoritmo tem desempenho melhor em alguns casos (seção 7), pois percorre menos estados da árvore.

Foi possível perceber que a implementação de MVR, se assemelha muito com a função de avaliação do algoritmo A*, porém ao invés de procurar o menor caminho, procura os nós mais críticos, ou seja, com menos opções de cores. Junto com a Verificação Adiante, onde é possível dizer com antecedência se os nós da folha serão solução ou não, o *backtracking* se torna uma Busca Informada com satisfação de restrições perfeito para o problema de coloração de mapas.

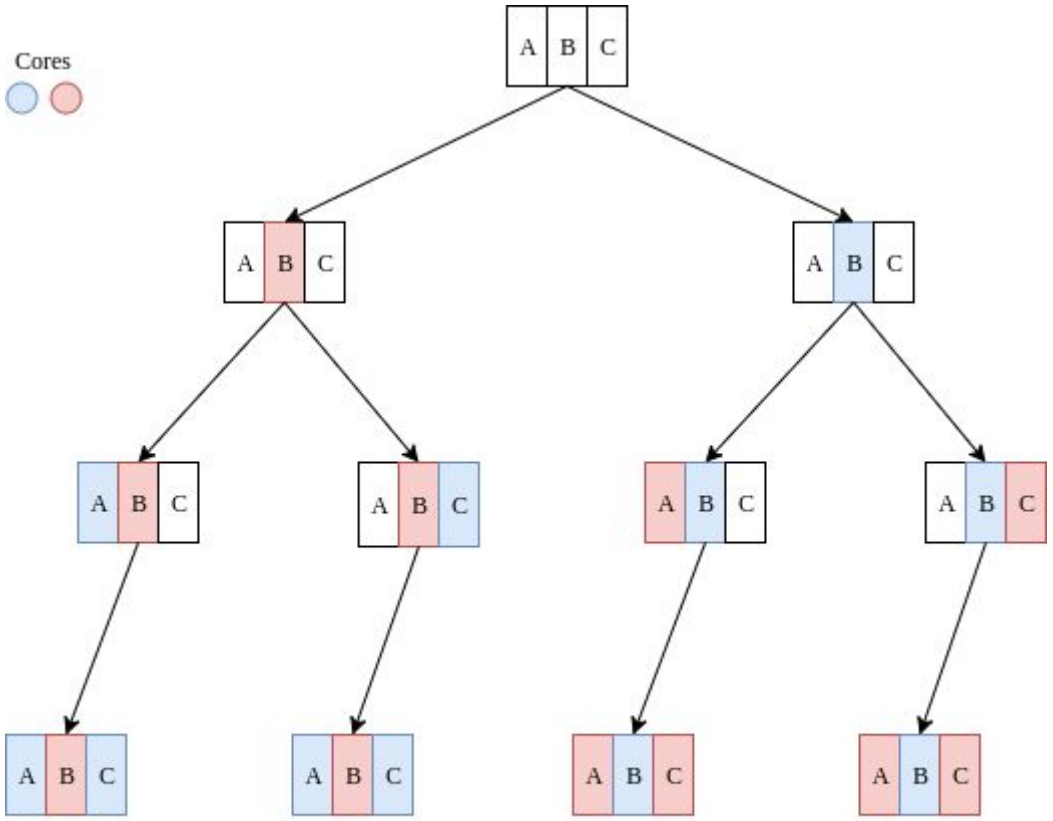

Imagem 4 - Backtracking com MVR e valor mais restritivo

Na imagem acima é possível ver os cortes realizados pela heurística MVR com verificação adiante, onde são escolhidos as regiões com o menor número de cores possíveis para colorir, usando como critério de desempate as regiões com maior número de vizinhos. Tais heurísticas reduzem muito o número de nós raízes que não são solução, na imagem todos os nós raízes são solução.

6) Implementação

O código foi desenvolvido totalmente em linguagem C em ambiente *Ubuntu 14.04* e compilado com o *gcc 4.8.4*. Para a representação do grafo foram utilizadas estruturas de dados do tipo lista ligada para acomodar as regiões dos mapas e suas regiões adjacentes.

O projeto está dividido em três arquivos principais, a saber:

- *main.c*: realiza o processamento dos dados de entrada e exibe a saída do programa.
- *busca.c*: é responsável pelo mecanismo da busca no espaço de estados, ou seja, é o *core* do projeto. Possui as implementações das Buscas Cega e Informada.
- *tad.c*: engloba as estruturas de dados utilizadas para representar o problema em forma de grafo.

Também cabe ressaltar a existência dos arquivos *busca.h* e *tad.h* que são arquivos contendo o cabeçalho e declaração dos protótipos das funções.

Compilação e Execução

Dentro da pasta contendo o código-fonte do projeto, executar o comando **`$ make all`** para compilar e **`$ make run`** para executar.

Entrada de dados

A entrada de dados é realizada pela entrada padrão na linha de comando e pode ser feita logo após a realização do comando de Execução. Para maior facilidade na pasta do projeto há um diretório chamado *Entradas* cujo conteúdo são as regiões juntamente dos seus estados adjacentes. São três arquivos que representam o Brasil, E.U.A. e Europa. O conteúdo dos arquivos pode ser copiado e colado no terminal, com o cuidado de alterar previamente a heurística de busca desejada que será explicada em seguida.

A entrada de dados segue o padrão:

```
<NumRegioes> <Heuristica>
<Regiao1: RegiaoAdjacente1, RegiaoAdjacente2, ... , RegiaoAdjacenteM.>
<Regiao2:  RegiaoAdjacente1, RegiaoAdjacente2, ... , RegiaoAdjacenteM.>
    ...
```

Sendo:

- `<NumRegioes>` - a quantidade de estados existentes na região de análise.
- `<Heuristica>` - a heurística utilizada para coloração do mapa, esse atributo permite quatro valores:
    - *a* = backtracking normal (Busca Cega);
    - *b* = backtracking com verificação adiante (Busca Informada);
    - *c* = backtracking com verificação adiante e MVR (Busca Informada);
    - *d* = backtracking com verificação adiante, MVR e variável mais restritiva (escolhe-se a variável com mais restrições para o desempate do MVR, ou seja, que possui mais regiões adjacentes) (Busca Informada).

A Imagem 4 apresenta uma exemplificação reduzida da entrada:

```
27 d
Acre: Amazonas, Rondonia.
Alagoas: Pernambuco, Sergipe, Bahia.
Amapa: Para.
Amazonas: Acre, Rondonia, Mato Grosso, Para, Roraima.
Bahia: Alagoas, Sergipe, Pernambuco, Piaui, Tocantins, Goias, Minas Gerais, Espirito Santo.
Ceara: Piaui, Pernambuco, Paraiba, Rio Grande do Norte.
Distrito Federal: Goias.
Goias: Tocantins, Mato Grosso, Mato Grosso do Sul, Minas Gerais, Bahia, Distrito Federal.
Espirito Santo: Bahia, Minas Gerais, Rio de Janeiro.
Maranhao: Para, Tocantins, Piaui.
```

Imagem 4 - entrada de dados utilizando os 27 estados do Brasil

Saída de dados

O resultado do algoritmo é apresentado na saída padrão na linha de comando, contendo o tipo de heurística escolhida, quantas atribuições de cores (entre atribuições corretas e trocas de cores devido ao backtracking ou heurísticas) foram necessárias para alcançar a solução e o tempo decorrido em milissegundos. Por fim, para cada região é mostrada a cor com a qual ela foi pintada. Caso não seja encontrada solução, o algoritmo irá informar o usuário.

A saída de dados segue o padrão:

```
                Uso da heurística: <Heurística>
Mapa colorido com sucesso em <N> atribuicoes de cor e em <S> ms.
<Regiao1: Cor>
<Regiao2: Cor>
     ...
```

Sendo:

- `<Heuristica>` - a mesma heurística apresentada na **Entrada de dados**.
- `<N>` - número de atribuições/trocas de cores realizadas até atingir o estado final.
- `<S>` - tempo decorrido para atingir o estado final.

A Imagem 5 apresenta uma exemplificação reduzida da saída:

```
                      Uso da heurística: d
Mapa colorido com sucesso em 27 atribuicoes de cor e em 0.243000 ms.


Acre:  Vermelho
Alagoas:  Amarelo
Amapa:  Vermelho
Amazonas:  Verde
Bahia:  Vermelho
Ceara:  Vermelho
```

Imagem 5 - saída de dados com a heurística *d* para a atribuição de cor aos 27 estados do Brasil

7) Experimentos

Para os testes foram utilizados os arquivos referentes aos mapas do Brasil, E.U.A e Europa presentes no diretório *Entradas*. Os formatos de entrada e saída já foram especificados na seção Implementação.

Nas Tabelas 1, 2 e 3 abaixo são apresentados os valores do tempo de execução e do números de atribuições de cores realizadas para cada um dos arquivos do diretório *Entradas*.

Tabela 1 - atribuições de cor e tempo de execução para o arquivo brasil.in

Região	**Heurística**	**Atribuições de cor**	**Tempo (em ms - média de 5 execuções)**
Brasil	a	395	0,2074
	b	29	0,0996
	c	27	0,1502
	d	27	0,2328

Tabela 2 - atribuições de cor e tempo de execução para o arquivo europe.in

Região	**Heurística**	**Atribuições de cor**	**Tempo (em ms - média de 5 execuções)**
Europa	a	43	0,0236
	b	43	0,1725
	c	41	0,3240
	d	41	0,3672

Tabela 3 - atribuições de cor e tempo de execução para o arquivo usa.in

Região	**Heurística**	**Atribuições de cor**	**Tempo (em ms - média de 5 execuções)**
E.U.A.	a	4088658	1721,467
	b	151368	576,337
	c	51	0,547
	d	51	0,584

Através da tabelas podemos notar que houve uma diferença significativa no número de atribuições ocorridas (quanto mais o algoritmo realizou o *backtracking*, maior é o número de atribuições ocorridas) e no tempo de execução apenas no mapa dos E.U.A.. Isso decorre diretamente do formato do grafo dessa região, visto que os estados possuem geralmente de 3 a 6 estados vizinhos (o que reflete em um número de restrições muito alto) e desse modo é mais custoso chegar na solução final. Os mapas do Brasil e da Europa possuem alguns estados com muitas restrições, mas a diferença está na presença de alguns estados que possuem 1 ou 2 restrições, tal fato facilita o trabalho do algoritmo de coloração.

Também podemos explicar o número de atribuições com uma perspectiva da busca em árvore. Quanto maior este número, mais fundo o algoritmo busca na árvore, ao utilizar as heurísticas de busca informada tem-se uma diminuição dessas atribuições, pois o algoritmo não precisa descer na árvore de busca para identificar que um estado não é válido. Em alguns casos como o mapa da Europa o *overhead* de tempo não compensa, isso decorre da própria organização das restrições como mencionado no parágrafo anterior e também da aplicação das heurísticas de busca. Por exemplo: utilizando a heurística *d* o tempo de execução é extrememente maior do que a heurística *a*, pois o algoritmo deve realizar a aplicação de 3 heurísticas e só faz 2 atribuições a menos que o algoritmo de Busca Cega.

8) Conclusão

No projeto implementamos a coloração de mapas utilizando 4 cores distintas seguindo as abordagens de Busca Cega e Busca Informada (utilizando heurísticas de Verificação Adiante, Mínimos Valores Remanescentes (MVR) e Variável mais restritiva). Em relação à Busca Informada e os métodos aprendidos em sala de aula (A*, Hill-Climbing entre outros) não conseguimos aplicá-los no problema escolhido, pois ao desenvolver a ideia do algoritmo acabávamos caindo nas heurísticas já conhecidas as quais foram aplicadas em nosso projeto.

É fascinante observar como pode-se extrair um problema do mundo real e simplificar através de uma busca em espaço de estados realizada por um algoritmo relativamente simples, que utiliza o *backtracking* como sua principal ferramenta. Ademais, a utilização de heurísticas em certos casos traz benefícios de mais de 2000 vezes em tempo de execução, como no caso dos E.U.A., e em certos casos não há diferenças significativas ou até há piora no tempo de execução. Nota-se que a maneira como está disposto o mapa bem como o número de restrições médio de cada estado influencia diretamente o desempenho do algoritmo.

Por fim, conseguimos adquirir um conhecimento mais profundo em relação ao que aprendemos na disciplina de Algoritmos Avançados, já que aqui entendemos o que é uma busca, como um problema real pode ser modelado em um problema de Inteligência Artificial e as diferentes abordagens que são possíveis de aplicação.