Meta-heurística GRASP para o Problema de Dominação Romana

Ieremies Romero

3 de junho de 2022

Resumo

A Dominação Romana é um problema proposto por Stewart 1999 em que desejamos defender o império romano dispondo de um certo número de legiões. Para isso, cada cidade deve ser assegurada de forma que ou ela possua uma legião alocada, ou seja, vizinha de outra que possua duas legiões. Assim, deseja-se minimizar a quantidade de legiões distribuídas sem abdicar da segurança do império.

Esse projeto visa estudar o Problema de Dominação Romana utilizando a meta-heurística do GRASP (*greedy randomized adaptive search procedure*) apresentado por Resende e Ribeiro 2019. Desejamos entender como os parâmetros e variações da meta-heurística influenciam o tempo de execução e o valor das soluções produzidas e obter boas soluções em tempos computacionais razoáveis.

Para isso, utilizaremos uma larga base de instâncias já apresentadas na literatura das quais coletaremos dados empíricos para realizar comparações relativas entre as diferentes configurações testadas.

1 Introdução

1.1 Motivação histórica

Proposto inicialmente por Stewart 1999, durante a Segunda Guerra Mundial, General Douglas MacArthur propôs uma estratégia de movimentação que consistia em avançar suas tropas de uma ilha para outra apenas quando ele poderia deixar para trás um número suficiente de tropas. Ele não foi o primeiro a utilizar dessa estratégia: segundo Stewart 1999, referências históricas apontam que o Imperador Constantino, no quarto século A.C., aplicou estratégia similar para defender o Império Romano de invasões dos povos ditos "bárbaros".

Para exemplificar o uso de tal estratégia, considere o mapa do Império Romano simplificado na Figura 1. Nesse exemplo, o imperador possui 4 legiões para serem distribuídas pelo território e ele deseja fazê-lo de forma que todas as cidades sejam consideradas **seguras**. Uma região é dita segura, ou **coberta**, se há uma legião em seu território ou se está conectada a outra região com duas legiões.

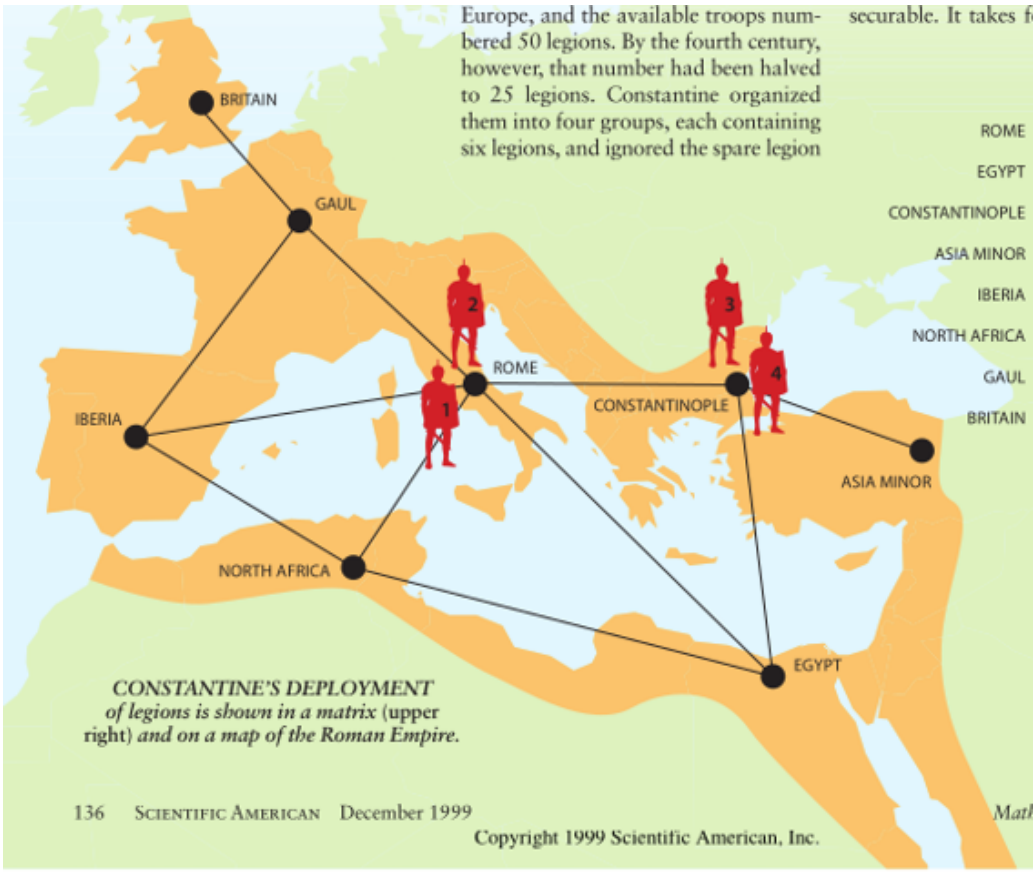

Figura 1: Representação do mapa do Império Romano usada como ilustração do problema retirada de Stewart 1999.

1.2 Modelo matemático

Para um grafo $G = (V, E)$, dizemos que a **vizinhança aberta** $N(v)$ de um vértice v é definida como o conjunto de vértices adjacentes a v em G, ou seja, $N(v) = \{u|(u, v) \in E\}$. Similarmente, dizemos que a **vizinhança fechada** $N[v]$ de um vértice v é a vizinhança aberta incluindo o próprio v, ou seja, $N[v] = N(v) \cup \{v\}$. Para um conjunto de vértices s, a vizinhança aberta desse conjunto é a união das vizinhanças abertas de cada um dos seus vértices (o respectivo pode ser dito para a vizinhança fechada).

Um **conjunto dominante** de um grafo G é um subconjunto de vértices D tal que a vizinhança fechada de D é o próprio conjunto V. Por sua vez, o **número de dominação** de um grafo G, dito $\gamma(G)$, é a cardinalidade do menor conjunto dominante do grafo G.

Para o problema de dominação romana, podemos defini-lo num grafo $G = (V, E)$ simples, finito e não-direcionado, onde cada vértice representa uma cidade ou região do império e as arestas são as conexões entre elas. Diremos que uma **função de dominação romana** é uma função $f : V \to \{0, 1, 2\}$ na qual $f(v)$ indica a quantidade de legiões naquela região, de forma que, para qualquer v tal que $f(v) = 0$, deve existir u vizinho a v cujo $f(u) = 2$. Definimos o **número de dominação romana total** de um grafo G como o menor valor $\sum_{v \in V} f(v)$ tal que f é uma função de dominação romana do grafo G.

1.3 Revisão bibliográfica

Após a descrição do problema por Stewart 1999, ReVelle e Rosing 2000 apresentou o desenvolvimento teórico inicial. Além disso, Cockayne et al. 2004 apresentou alguns resultados importantíssimo de teoria de grafos sobre o problema, com limitantes e propriedades da função de dominação romana, os quais foram estendidos e aprimorados por Xing, Chen, e Chen 2006, Favaron et al. 2009, Mobaraky e Sheikholeslami 2008. Klobučar e Puljić 2014 demonstraram que algumas classes especiais de grafos podem ser resolvidas em tempo linear, mas, no caso geral, o problema é NP-difícil (Dreyer Jr 2000; Klobučar e Puljić 2014; Shang e Hu 2007).

Ivanović e Urošević 2019 utilizaram **Variable Neightborhood Search** (VNS) no mesmo problema, obtendo resultados interessantes para as mesmas instâncias propostas por Currò 2014 que usaremos nesse projeto. Essa meta-heurística parte da ideia de que soluções ótimas são encontradas "próximas"de boas soluções, assim utilizando busca local e algumas técnicas para escapar de mínimos locais e intensificar a procura.

Já Khandelwal, Srivastava, e Saran 2021 utilizaram **algoritmos genéticos** no problema de dominação romana, uma ideia que toma de inspiração da evolução das espécies observadas na natureza. Partindo de um conjunto de soluções, realizamos os chamados "cruzamentos"das melhores para produzir novas gerações. A cada uma, induzimos "mutações"aleatórias que alteram certos pontos das soluções, espelhando a realidade.

Além disso, Filipović, Matić, e Kartelj 2022 também utilizaram a VNS e **programação por restrição** no problema, mas obtiveram melhores resultados com duas novas formula-

ções inteiras que eles mesmo propuseram. Neste, também foram utilizadas as instâncias de Currò 2014.

2 Metodologia

Para esse problema utilizaremos a meta-heurística **GRASP** (*greedy randomized adaptive search procedure*), proposta por Resende e Ribeiro 2019 para problemas de minimização. Esta consiste em alternamos, a cada iteração, entre: geração de novos candidatos e busca local para busca de novos mínimos. Na primeira, utilizamos uma heurística gulosa aleatorizada para construir uma solução viável. Na segunda, realizamos uma busca local partindo desta solução. Por fim, repetimos esse processo um certo número de vezes, guardando a melhor solução encontrada.

```
procedure Greedy_Randomized_Construction(Seed)
1     Solution ← ∅;
2     Initialize the set of candidate elements;
3     Evaluate the incremental costs of the candidate elements;
4     while there exists at least one candidate element do
5           Build the restricted candidate list (RCL);
6           Select an element s from the RCL at random;
7           Solution ← Solution∪{s};
8           Update the set of candidate elements;
9           Reevaluate the incremental costs;
10    end;
11    return Solution;
end Greedy_Randomized_Construction.
```

Figura 2: Algoritmo apresentado por Resende e Ribeiro 2019 para a fase construtiva do GRASP.

É importante salientar que, como demonstrado em Resende e Ribeiro 2019, não é necessário fazer a etapa de "reparo"da solução se só incluirmos nela aqueles que não torna-la-ão inviável. Tais variáveis são chamadas **candidatas** e, a cada iteração da etapa de construção, montamos (ou atualizamos) a chamada lista de candidatas **CL**.

Nesta etapa, a cada iteração analisamos cada elemento da lista CL e qual custo que sua inserção na solução irá causar. Em posse do maior e menor custos nesta lista, selecionamos aleatoriamente elementos que estão suficientemente próximos do menor custo, tais elementos compõem a chamada **lista restrita de candidatas** (RCL). A definição de suficientemente fica a cargo do parâmetro α e é relativa ao intervalo de valores que obtivemos na análise da lista. Podemos repetir esse processo até que a lista de candidatos seja esgotada.

Já na etapa de busca local, analisamos as vizinhanças da nossa solução procurando por melhorias locais até não ser mais possível. Para cada problema, podem existir diversas

```
procedure Local_Search(Solution)
1     while Solution is not locally optimal do
2           Find s' ∈ N(Solution) with f(s') < f(Solution);
3           Solution ← s';
4     end;
5     return Solution;
end Local_Search.
```

Figura 3: Algoritmo apresentado por Resende e Riberio 2019 para a busca local do GRASP.

definições de vizinhança e mais de uma pode ser utilizada nessa fase.

2.1 Construtiva (heurística)

É importante nesse momento perceber a similaridade do nosso problema com o **problema de cobertura mínima de vértices** de um grafo. Neste, almeja-se encontrar um conjunto de vértices V' no grafo $G = (V, E)$ tal que a vizinhança fechada de V' seja o próprio V.

Assim, se para o nosso problema, atribuirmos a todos os elementos de V' acima construído duas legiões e aos vértices $V \setminus V'$ nenhuma, temos uma solução viável. Além disso, Parekh 1991 demonstra uma simples heurística para resolver o problema de cobertura cuja cardinalidade do conjunto resultante é menor que $n+1-\sqrt{2m+1}$, sendo $n = |V|$ e $m = |E|$, portanto, o número de dominação romana total é até duas vezes esse valor.

A cada iteração, adicionamos o vértice de menor índice que possui a maior quantidade de vizinhos descobertos.

Teorema 2.1. A heurística acima produz soluções de custo menor que $2(n+1-\sqrt{2m+1})$.

2.2 Busca local

Na etapa de busca local, como descrito anteriormente, partimos de uma solução viável e, analisando a(s) vizinhança(s) desta solução, tomamos "passos"em direção a melhorar nossa função objetivo. O desafio então jaz em decidir quem serão nossas vizinhanças já que qualidade da busca depende diretamente nelas.

Algumas possibilidades de vizinhanças a serem estudadas são:

- inserção de um novo elemento da CL na solução.
- remoção de um elemento já presente na solução.
- substituição de um elemento na solução por outro na CL.

Além disso, estudaremos o uso de vizinhanças maiores que envolvem a remoção, inserção ou substituição de mais de um elemento simultaneamente.

Além disso, estudaremos abordagens de como decidir qual "passo"a ser tomado: **best-improving** e **first-improving**. Na primeira, percorremos todos os vizinhos (soluções que podem ser obtidas a partir de uma das operações acima) e tomamos o passo na direção do vizinho que melhor afeta nossa função objetivo (no nosso caso, o de maior contribuição). Em contrapartida, a segunda nos propõe a tomar o primeiro "bom vizinho", ou seja, o primeiro vizinho encontrado que melhora a nossa solução.

2.3 Técnicas alternativas

Além da implementação padrão da meta-heurística, almejamos implementar algumas variações na sua abordagem, como apresentadas por Resende e Ribeiro 2019:

construção gulosa por amostra modificamos a ordem dos passos na heurística construtiva: da lista de candidatos, montamos a lista restrita com no máximo p elementos aleatórios e deles escolhemos o melhor. Perceba que o parâmetro p, nesse caso, é determina o balanço entre as abordagens gulosa e aleatória.

POP *Proximate Optimality Principle* propõem que, em alguns momentos da heurística construtiva, realizemos alguns passos de busca local, o que corrigiria algumas "imperfeições"criadas pela heurística.

Bias na versão padrão, usamos uma função de distribuição de probabilidade constante, onde cada elemento possui a mesma chance de ser escolhido de RCL. Nessa nova versão, alteramos essa função para candidatas como exponencial, logarítmica e linear. É importante ressaltar que todas essas funções citadas acima ainda mantém uma "preferência"pelos menores valores.

3 Avaliação dos resultados

Como nosso objetivo é a construção de boas soluções usando a meta-heurística GRASP para o problema de dominação romana, experimentaremos em um largo conjunto de instâncias os métodos apresentados aqui (e possivelmente outros). Para tal, experimentaremos com diferentes parâmetros, como valores de α, critério do passo de busca local (de "best"para "first") dentre outros parâmetros do algoritmo e suas variações.

Além disso, cada instância terá um tempo limite de 10 minutos.

Para as instâncias desse projeto, usaremos aquelas originalmente apresentadas por (Currò 2014) divididas em 6 diferentes classes de grafos simétricos:

planar cada uma das 17 instâncias mapeia cada vértice a uma coordenada no plano. Os vértices são ligados aos seus vizinhos baseado numa probabilidade que é maior quanto mais próximos no plano eles se encontram.

grade composta por grafos que, imerso no plano $\mathbb{R}^2$, formam azulejos regulares (*regular tiling*). É composta de 171 instâncias que variam de 3x3 a 30x20, com 600 vértices.

rede os 4 grafos que a compõe são formados adicionando arestas que conectam os vértices "mas próximos"na diagonal dos grafos de grade.

bipartido com um total de 81 instâncias, dimensões variando de 50 a 400 vértices, a densidade é aleatória controlada por um parâmetro p.

randômico contendo 72 instâncias e dimensões de 50 a 200, cada par de vértice tem uma probabilidade de estarem ligados entre si por uma aresta.

recursivo as 7 instâncias possuem de 7 a 3283 vértices.

Com os resultados em mãos, ou seja, valor da função objetivo e tempo de execução para diferentes configurações, podemos comparar os valores das soluções encontradas com os valores ótimos sabidos para cada instância bem como os resultados obtidos por Filipović, Matić, e Kartelj 2022 e Ivanović 2016 sob as mesmas instâncias.

4 Referências bibliográficas

Cockayne, Ernie J, Paul A Dreyer Jr, Sandra M Hedetniemi, and Stephen T Hedetniemi. 2004. "Roman Domination in Graphs." *Discrete Mathematics* 278 (1-3). Elsevier: 11–22.

Currò, Vincenzo. 2014. "The Roman Domination Problem on Grid Graphs." Università di Catania.

Dreyer Jr, Paul Andrew. 2000. *Applications and Variations of Domination in Graphs*. Rutgers The State University of New Jersey-New Brunswick.

Favaron, Odile, Hossein Karami, R Khoeilar, and Seyed Mahmoud Sheikholeslami. 2009. "On the Roman Domination Number of a Graph." *Discrete Mathematics* 309 (10). Elsevier: 3447–51.

Filipović, Vladimir, Dragan Matić, and Aleksandar Kartelj. 2022. "Solving the Signed Roman Domination and Signed Total Roman Domination Problems with Exact and Heuristic Methods." *Arxiv Preprint Arxiv:2201.00394*.

Ivanović, Marija. 2016. "Improved Mixed Integer Linear Programing Formulations for Roman Domination Problem." *Publications de L'institut Mathematique* 99 (113): 51–58.

Ivanović, Marija, and Dragan Urošević. 2019. "Variable Neighborhood Search Approach for Solving Roman and Weak Roman Domination Problems on Graphs." *Computing and Informatics* 38 (1): 57–84.

Khandelwal, Aditi, Kamal Srivastava, and Gur Saran. 2021. "On Roman Domination of Graphs Using a Genetic Algorithm." In *Computational Methods and Data Engineering*, 133–47. Springer.

Klobučar, Aneta, and Ivona Puljić. 2014. "Some Results for Roman Domination Number on Cardinal Product of Paths and Cycles." *Kragujevac Journal of Mathematics* 38 (1): 83–94.

Mobaraky, BP, and SM Sheikholeslami. 2008. "Bounds on Roman Domination Numbers of Graphs." *Matematički Vesnik* 60 (234). Društvo matematičara Srbije: 247–53.

Parekh, Abhay K. 1991. "Analysis of a Greedy Heuristic for Finding Small Dominating Sets in Graphs." *Information Processing Letters* 39 (5): 237–40. doi:`https://doi.org/https://doi.org/10.1016/0020-0190(91)90021-9`.

Resende, Mauricio GC, and Celso C Ribeiro. 2019. "Greedy Randomized Adaptive Search Procedures: Advances and Extensions." In *Handbook of Metaheuristics*, 169–220. Springer.

ReVelle, Charles S, and Kenneth E Rosing. 2000. "Defendens Imperium Romanum: A Classical Problem in Military Strategy." *The American Mathematical Monthly* 107 (7). Taylor & Francis: 585–94.

Shang, Weiping, and Xiaodong Hu. 2007. "The Roman Domination Problem in Unit Disk Graphs." In *International Conference on Computational Science*, 305–12. Springer.

Stewart, Ian. 1999. "Defend the Roman Empire!" *Scientific American* 281 (6). JSTOR: 136–38.

Xing, Hua-Ming, Xin Chen, and Xue-Gang Chen. 2006. "A Note on Roman Domination in Graphs." *Discrete Mathematics* 306 (24). Elsevier: 3338–40.