

Seleção de Maior Distância Mínima Total Aplicação da Metaheurística GRASP

Andrei Pochmann Koenich

Pedro Company Beck



Trabalho Final

Professor Marcus Ritt

Otimização Combinatória

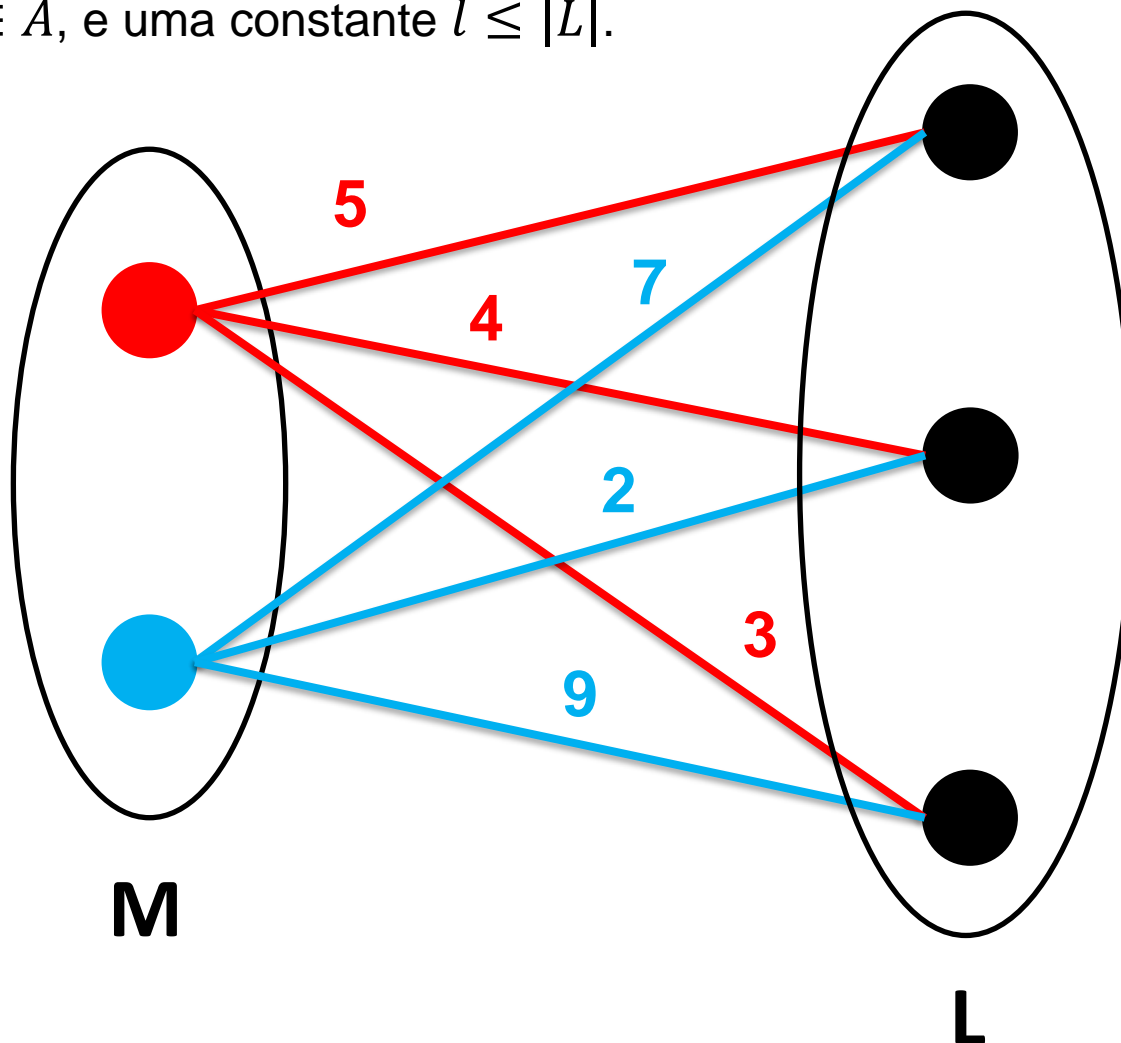
2022/02



Definição do Problema

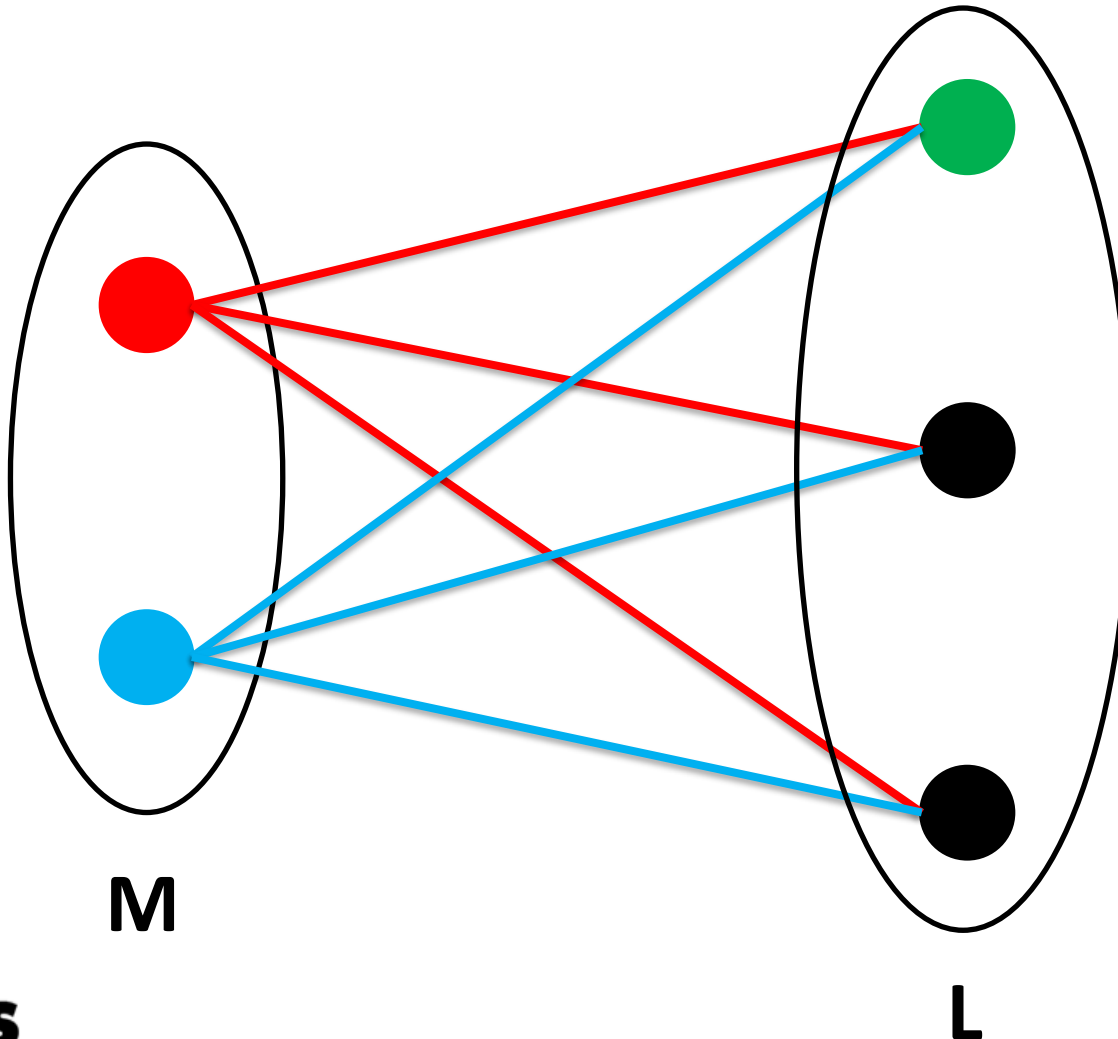
Definição do Problema

Instância: um grafo bipartido completo $G = (M \cup L, A)$ com distâncias d_a nas arestas $a \in A$, e uma constante $l \leq |L|$.



Definição do Problema

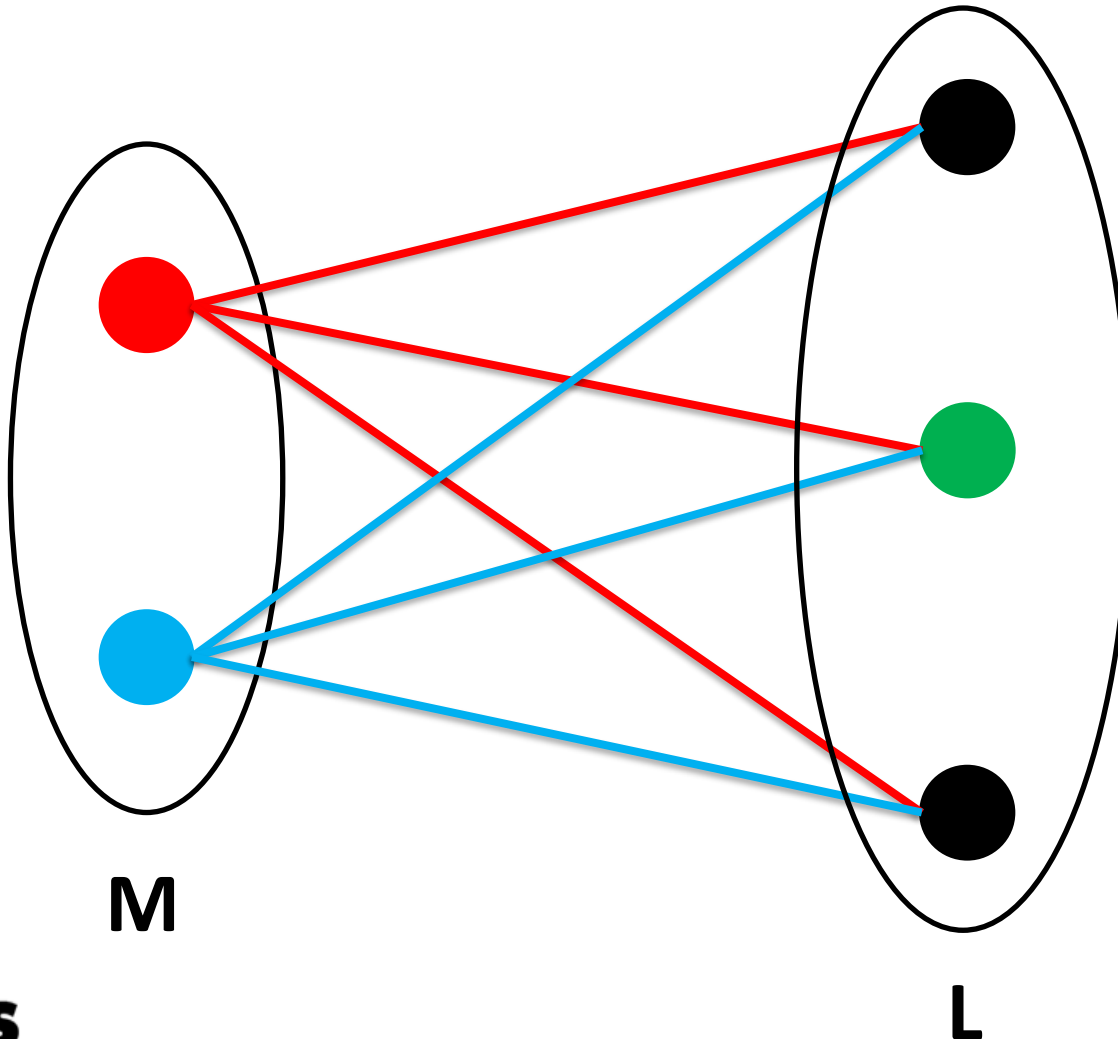
Solução: uma seleção $S \subseteq L$ com l elementos.



$l = 1$

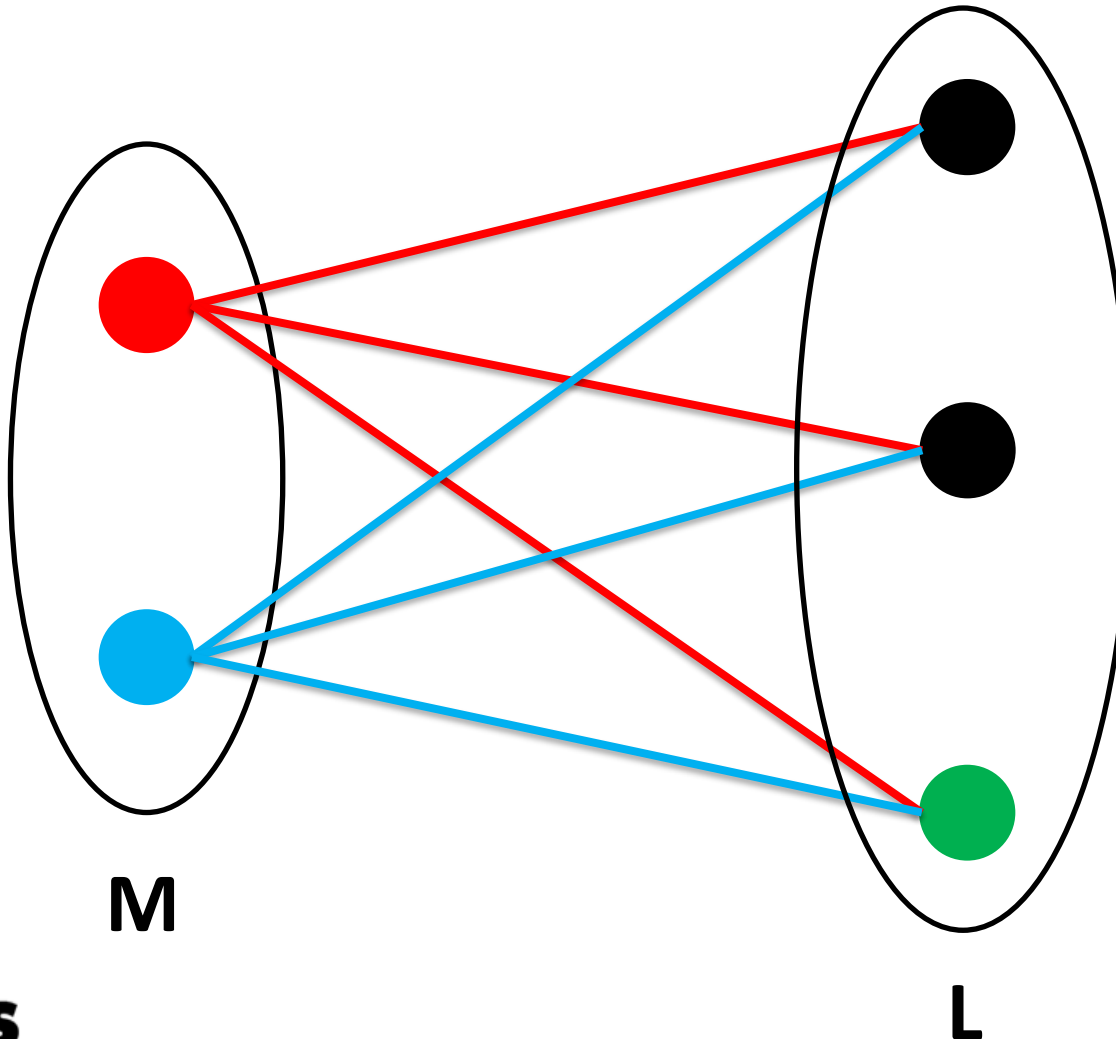
Definição do Problema

Solução: uma seleção $S \subseteq L$ com l elementos.



Definição do Problema

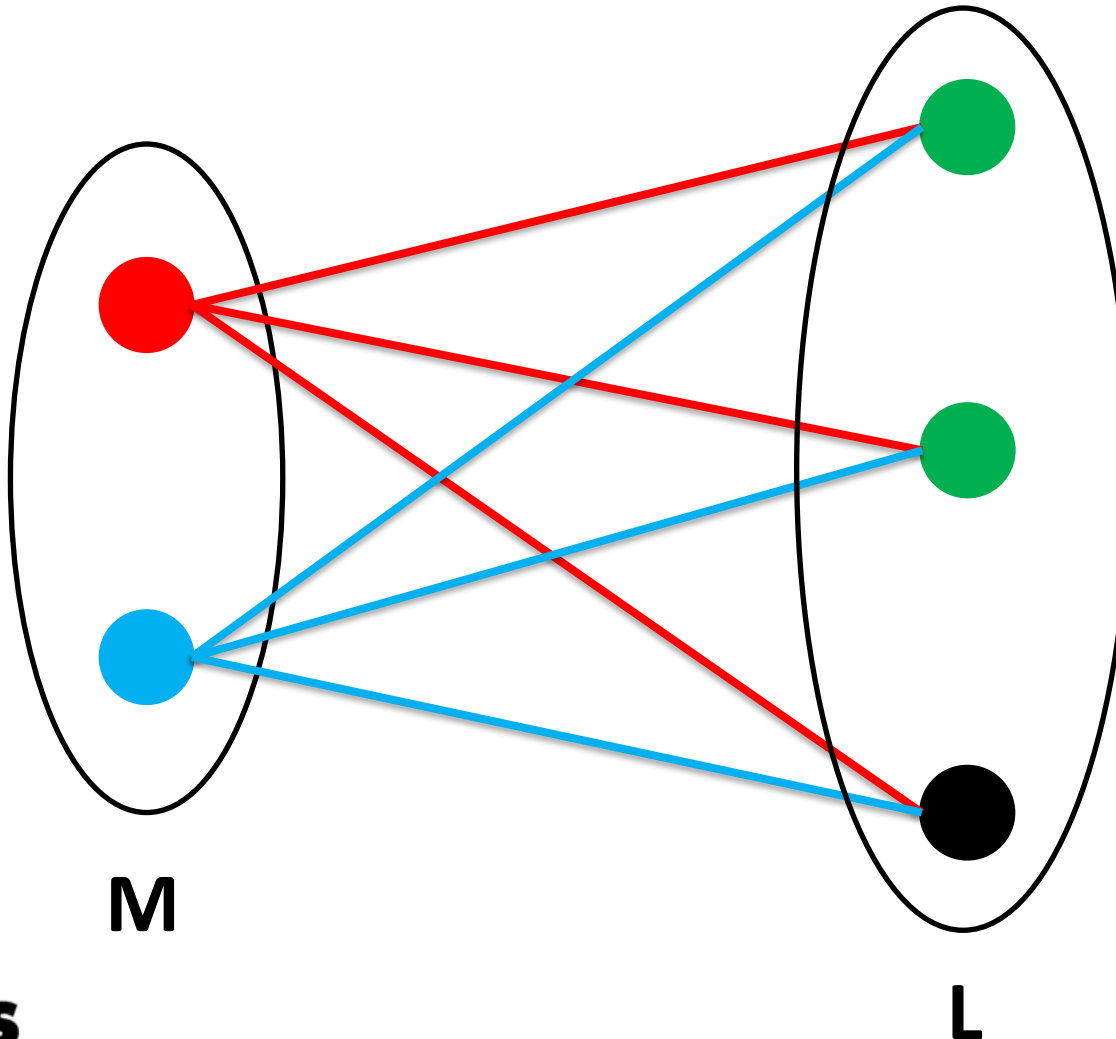
Solução: uma seleção $S \subseteq L$ com l elementos.



$l = 1$

Definição do Problema

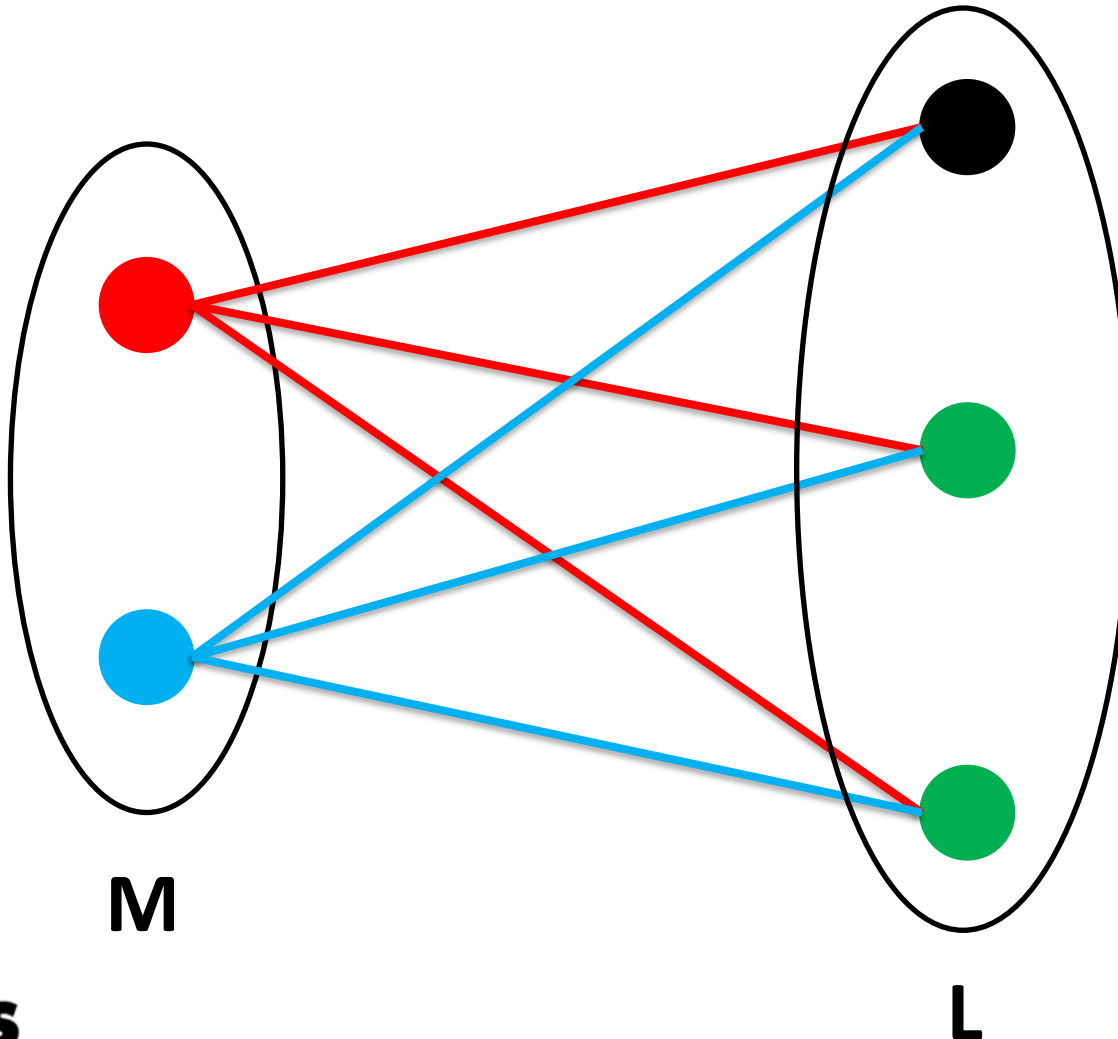
Solução: uma seleção $S \subseteq L$ com l elementos.



$l = 2$

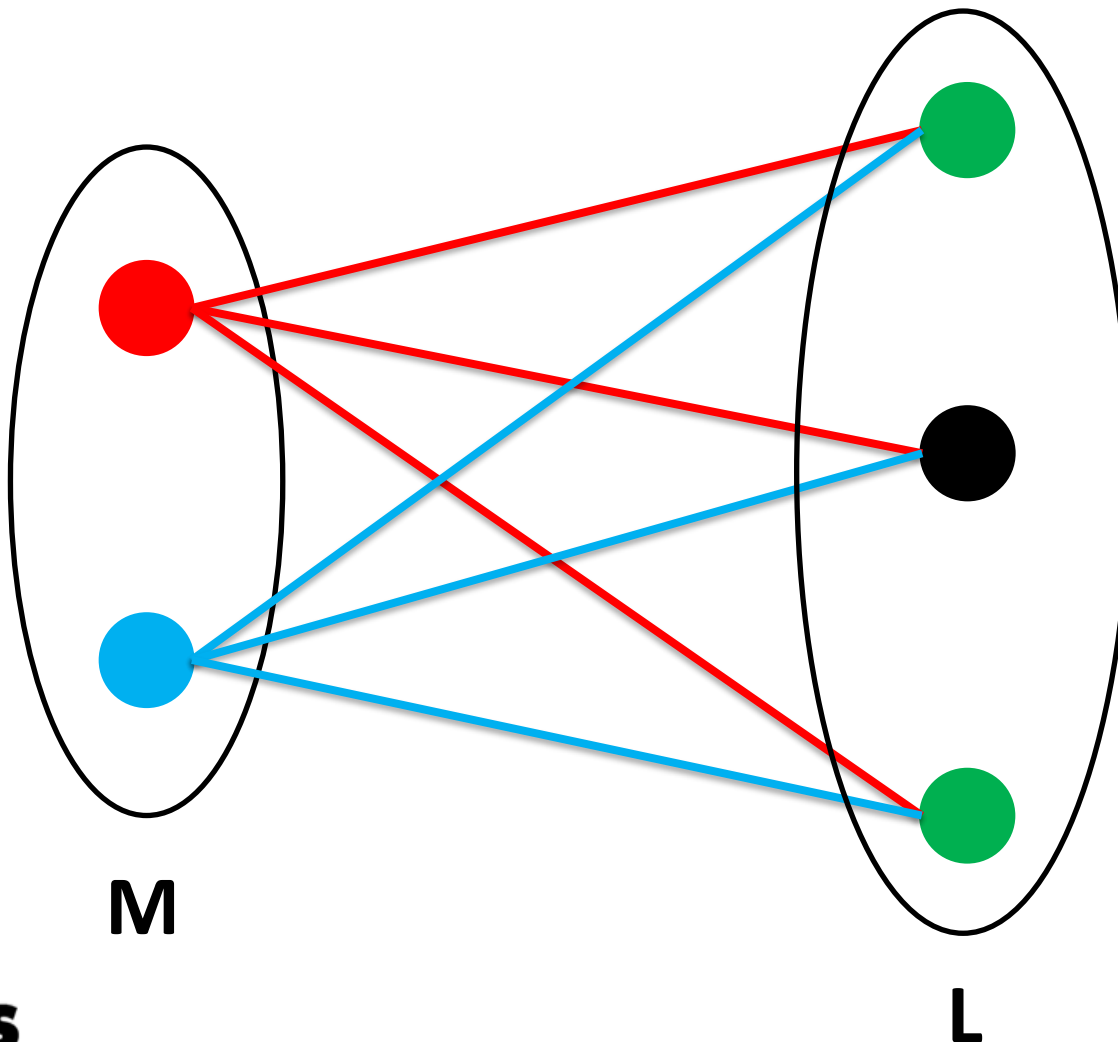
Definição do Problema

Solução: uma seleção $S \subseteq L$ com l elementos.



Definição do Problema

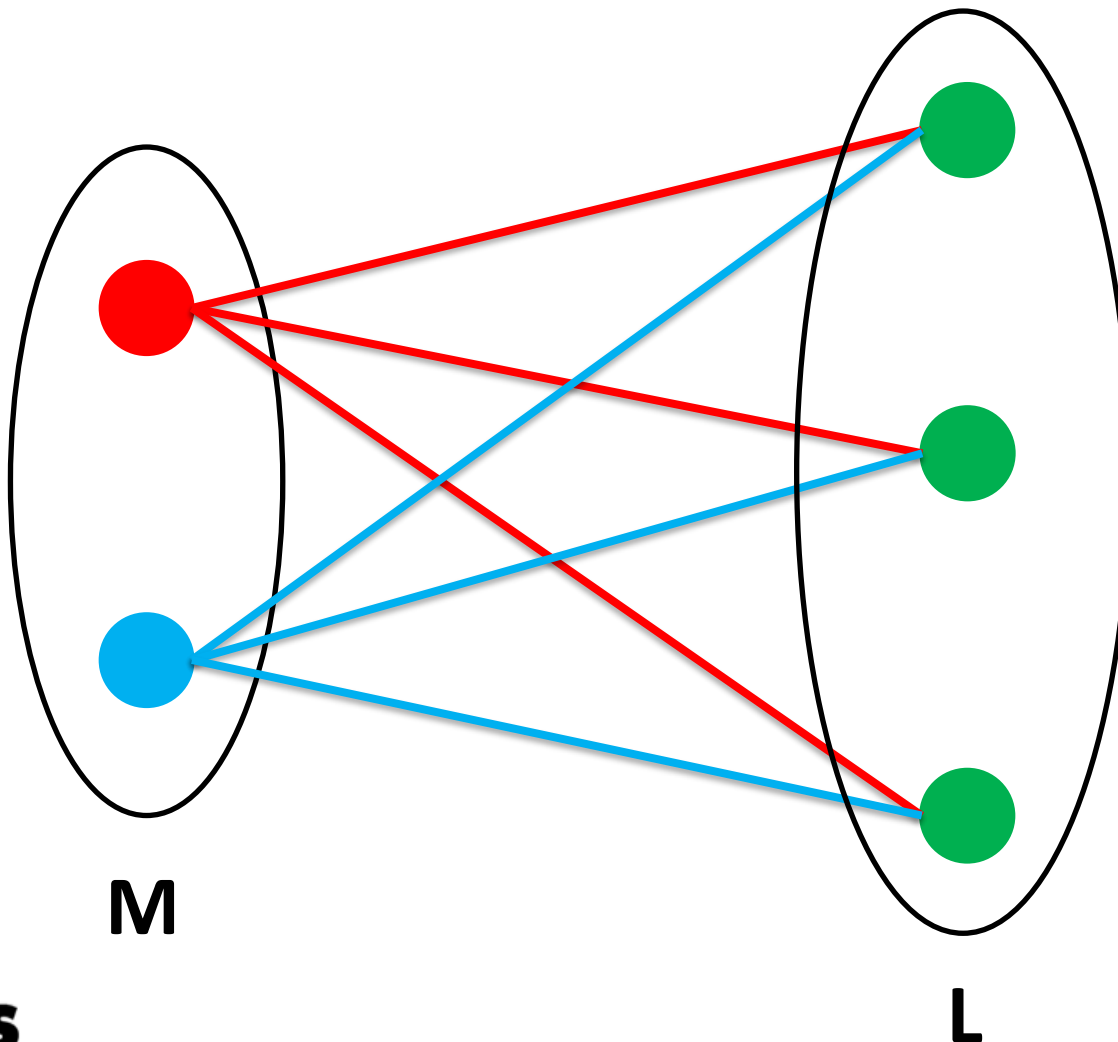
Solução: uma seleção $S \subseteq L$ com l elementos.



$l = 2$

Definição do Problema

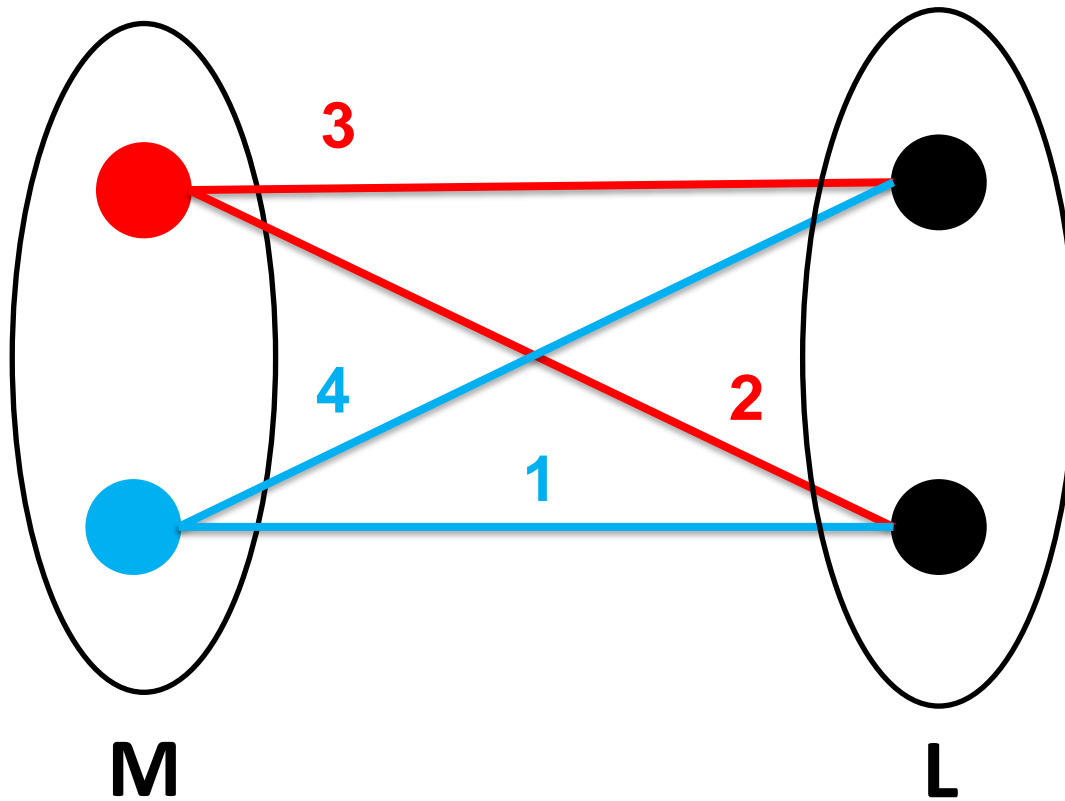
Solução: uma seleção $S \subseteq L$ com l elementos.



$l = 3$

Definição do Problema

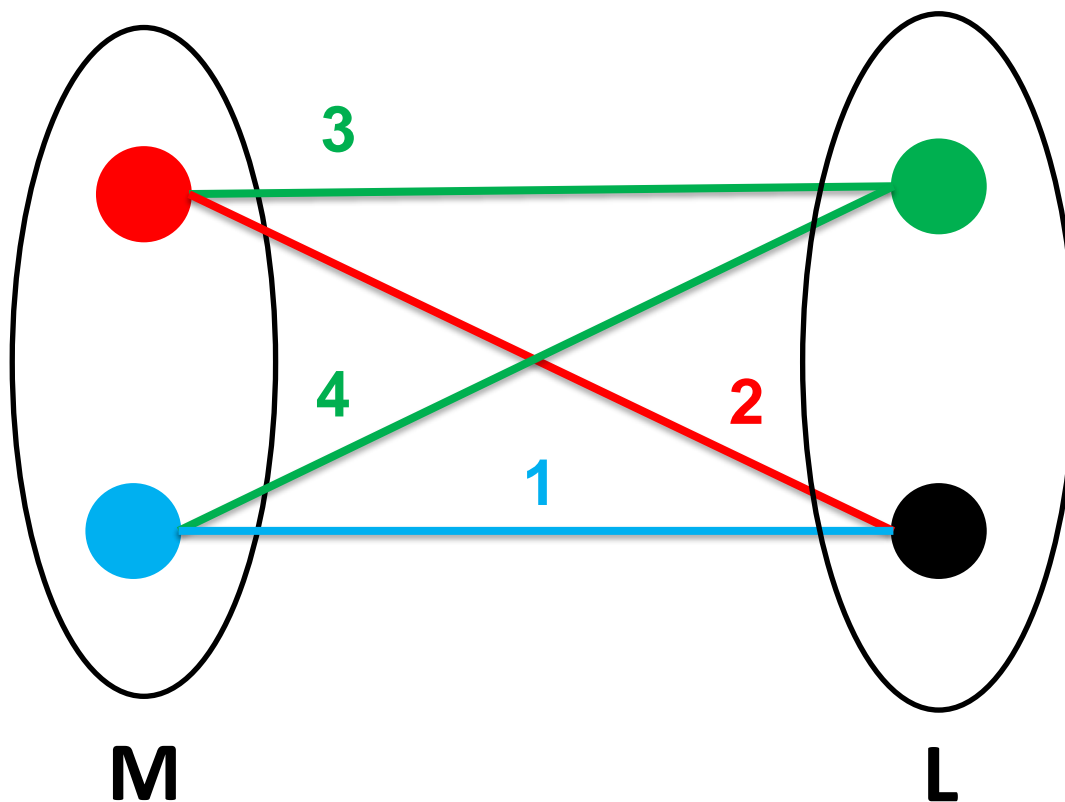
Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.



$$l = 1$$

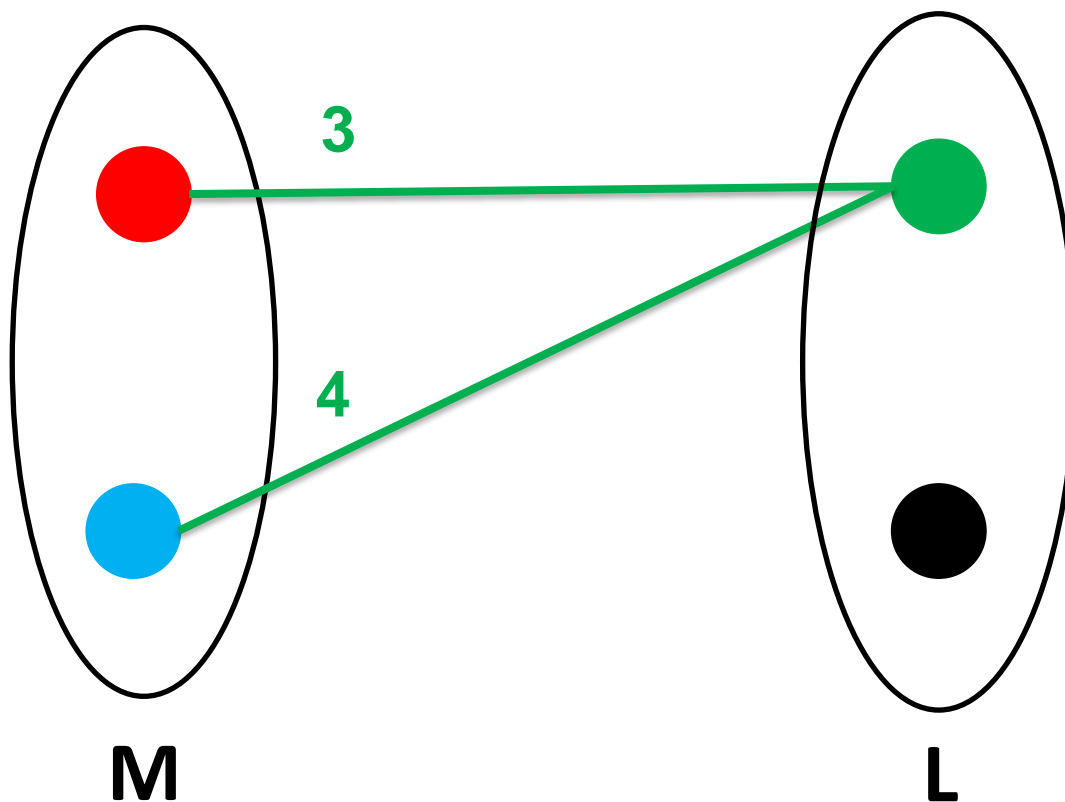
Definição do Problema

Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.



Definição do Problema

Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.

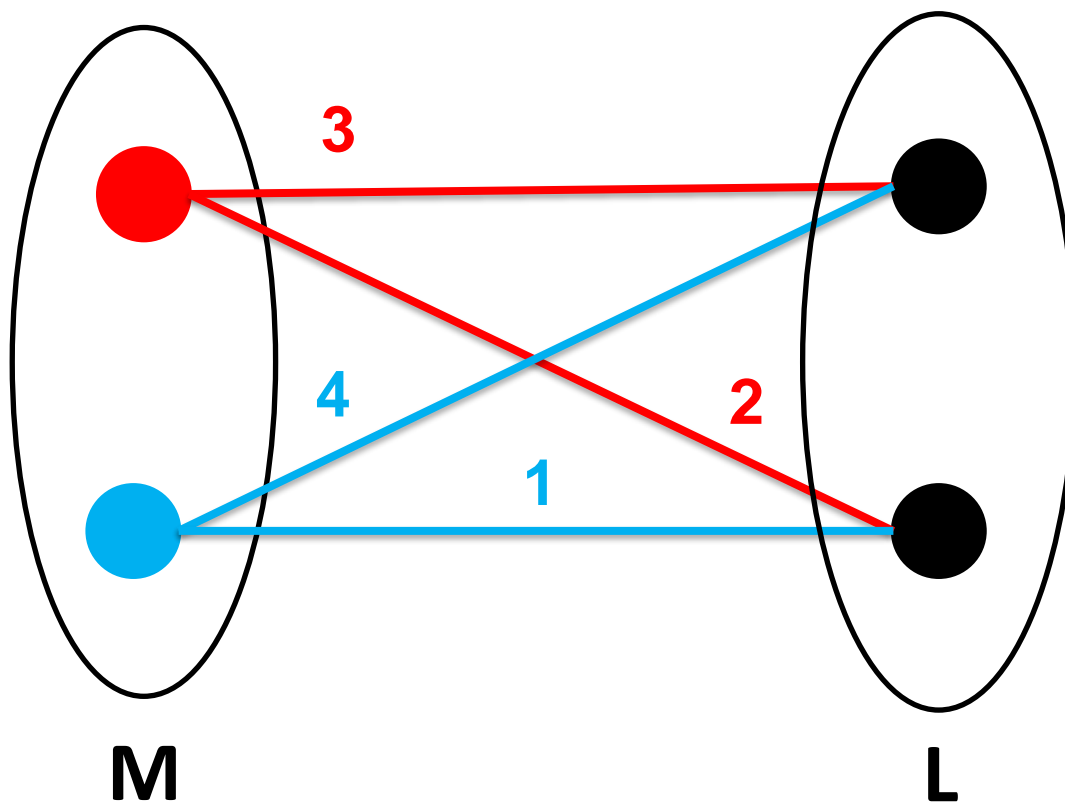


$$l = 1$$

Valor da função objetivo é $3 + 4 = 7$

Definição do Problema

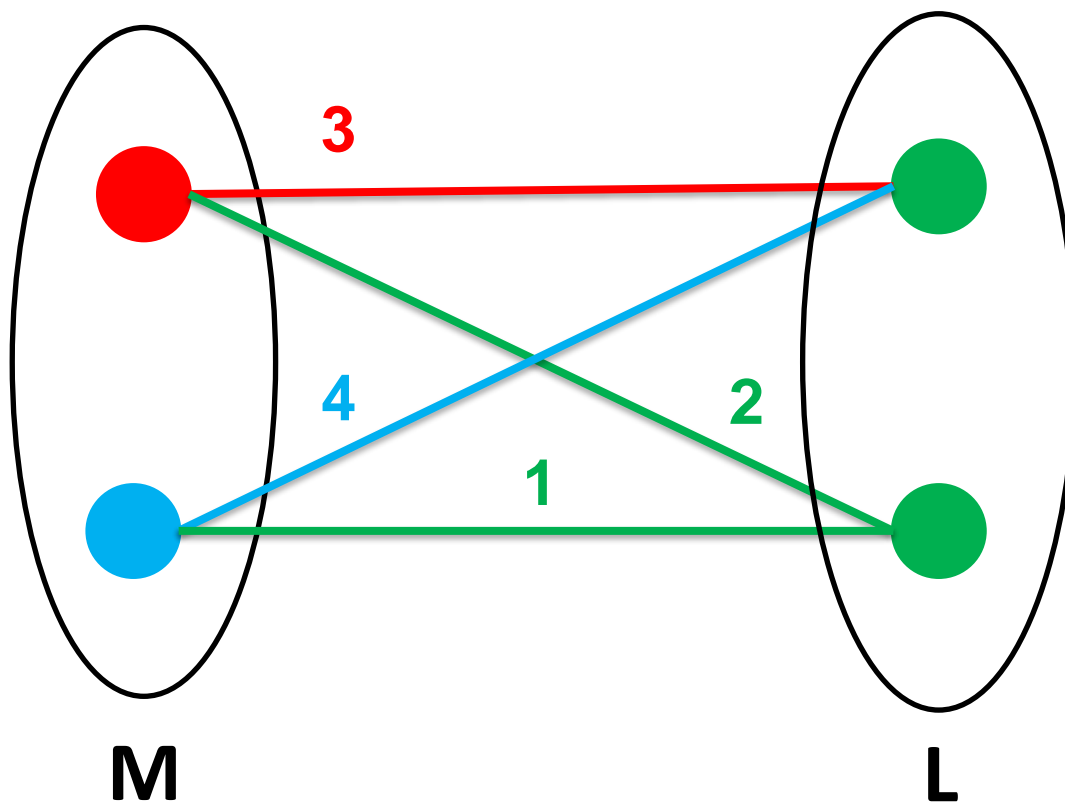
Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.



$$l = 2$$

Definição do Problema

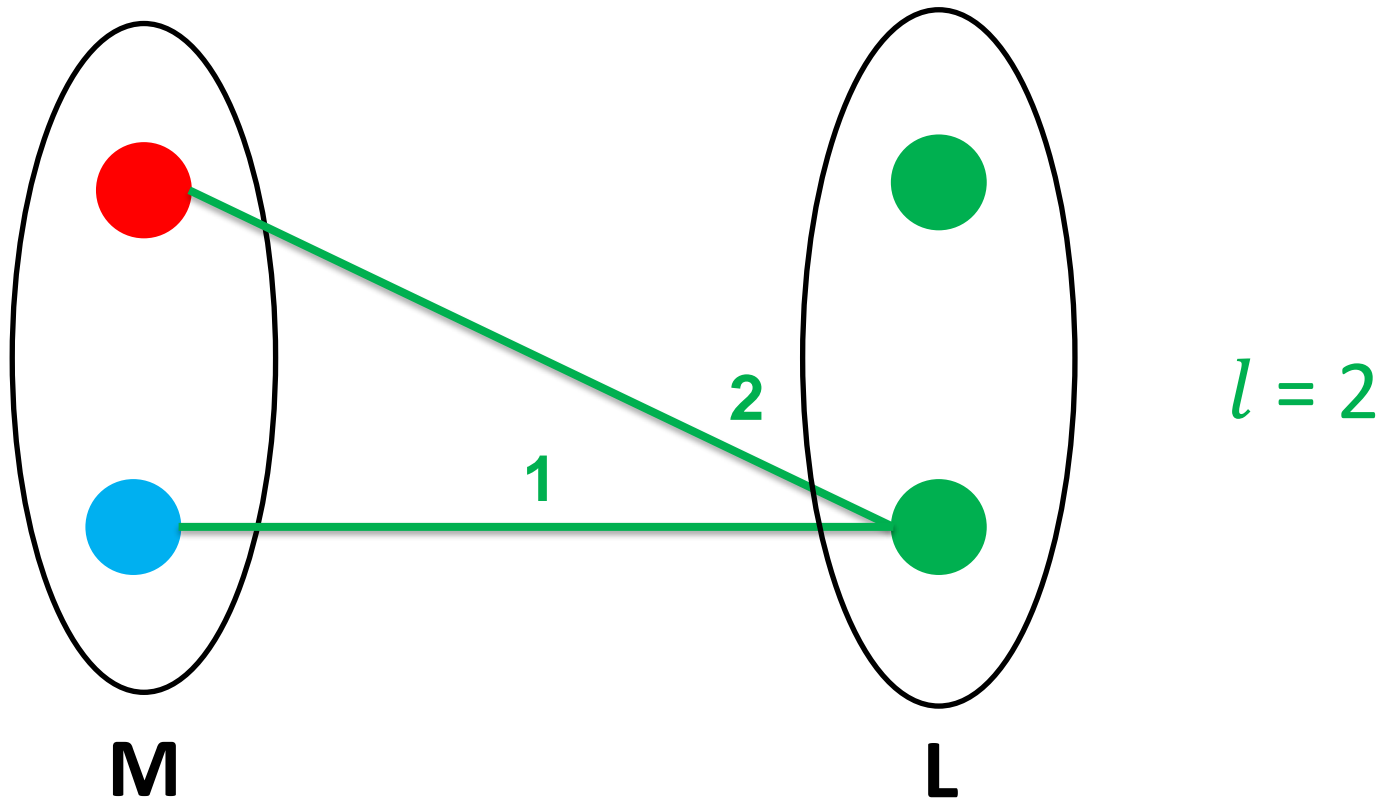
Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.




$$l = 2$$

Definição do Problema

Objetivo: para uma seleção S , definimos a distância mínima $d(m) = \min_{s \in S} d_{ms}$ de um elemento $m \in M$ para um elemento do conjunto S , que foi selecionado. O objetivo é maximizar a soma das distâncias mínimas $\sum_{m \in M} d(m)$.



Valor da função objetivo é $2 + 1 = 3$



Formulação Matemática do Problema



Descrições das Variáveis

- $x_j \rightarrow$ variável com valor igual a um caso o vértice j seja selecionado para integrar o conjunto de vértices S , com j correspondendo a um vértice de L , ou com valor igual a zero, caso contrário.
- $y_{ij} \rightarrow$ variável com valor igual a um caso o par de vértices (i, j) seja selecionado (com o vértice i correspondendo a um vértice de M , e o vértice j correspondendo a um vértice de L selecionado para integrar o conjunto de vértices S), ou com valor igual a zero, caso contrário.

Descrição dos Valores Recebidos como Entrada

- $d_{ij} \rightarrow$ distância existente entre os pares de vértices (i, j) , representada por meio dos pesos das arestas (com o vértice i correspondendo a um vértice de M , e o vértice j correspondendo a um vértice de L).

Função Objetivo

$$\sum_{i \in M} \sum_{j \in L} d_{ij} y_{ij}$$



Restrição #1

Devemos selecionar exatamente l vértices.

$$\sum_{j \in L} x_j = l$$



Restrição #2

Para cada vértice de M , devemos contabilizar exatamente uma das suas arestas que realiza uma conexão com um vértice de L , para obter o valor da função objetivo.

$$\forall i \in M, \sum_{j \in L} y_{ij} = 1$$

Restrição #3

Devemos estabelecer um vínculo entre as variáveis;

Não deve ser possível selecionar uma aresta ligando um vértice de M a um vértice de L, sem que o respectivo vértice de L tenha sido selecionado.

$$\forall i \in M, \forall j \in L, x_j - y_{ij} \geq 0$$



Restrição #4

Devemos escolher a aresta com a menor distância, que realiza uma conexão com um vértice de L , que pertence ao grupo dos vértices selecionados.

$$\forall i \in M, \forall j \in L, \forall k \in L, d_{ij}(x_k + y_{ij} - 1) \leq d_{ik}x_k$$

Restrições #5 e #6

As variáveis devem ser binárias.

$$\forall j \in L, x_j \in \{0,1\}$$

$$\forall i \in M, \forall j \in L, y_{ij} \in \{0,1\}$$

Programa Inteiro Final

$$\text{maximiza} \quad \sum_{i \in M} \sum_{j \in L} d_{ij} y_{ij}$$

$$\text{sujeito a} \quad \sum_{j \in L} x_j = l \quad (1)$$

$$\forall i \in M, \sum_{j \in L} y_{ij} = 1 \quad (2)$$

$$\forall i \in M, \forall j \in L, x_j - y_{ij} \geq 0 \quad (3)$$

$$\forall i \in M, \forall j \in L, \forall k \in L, d_{ij}(x_k + y_{ij} - 1) \leq d_{ik}x_k \quad (4)$$

$$\forall j \in L, x_j \in \{0,1\} \quad (5)$$

$$\forall i \in M, \forall j \in L, y_{ij} \in \{0,1\} \quad (6)$$



Metaheurística

GRASP



Metaheurística GRASP

Composta por duas fases:

- a) Construção Gulosa-Randomizada
- b) Busca Local

do

`s := GulosoRandomizado(α)`

`s := BuscaLocal(s)`

`atualiza s* caso $f(s) > f(s^*)$`

until critério de parada satisfeito

return s^*



Implementação da Metaheurística



Parâmetros de Entrada

`seed`: representa o valor utilizado para geração de números aleatórios, durante a execução da implementação. O usuário pode escolher qualquer valor inteiro do intervalo $[0,9999]$.

`inícios`: representa a quantidade de inícios aleatórios (segundo a construção gulosa-randomizada) que ocorrerão, para que a partir desses inícios ocorra a busca local.



Parâmetros de Entrada

`construcoes`: representa a quantidade de soluções aleatórias que será gerada pela implementação, na etapa da construção gulosa-randomizada.

`candidatos`: representa, dentre todas as soluções iniciais aleatórias geradas na construção gulosa-randomizada, quantas das melhores soluções obtidas participarão do “torneio” de seleção.



Representação de uma Solução

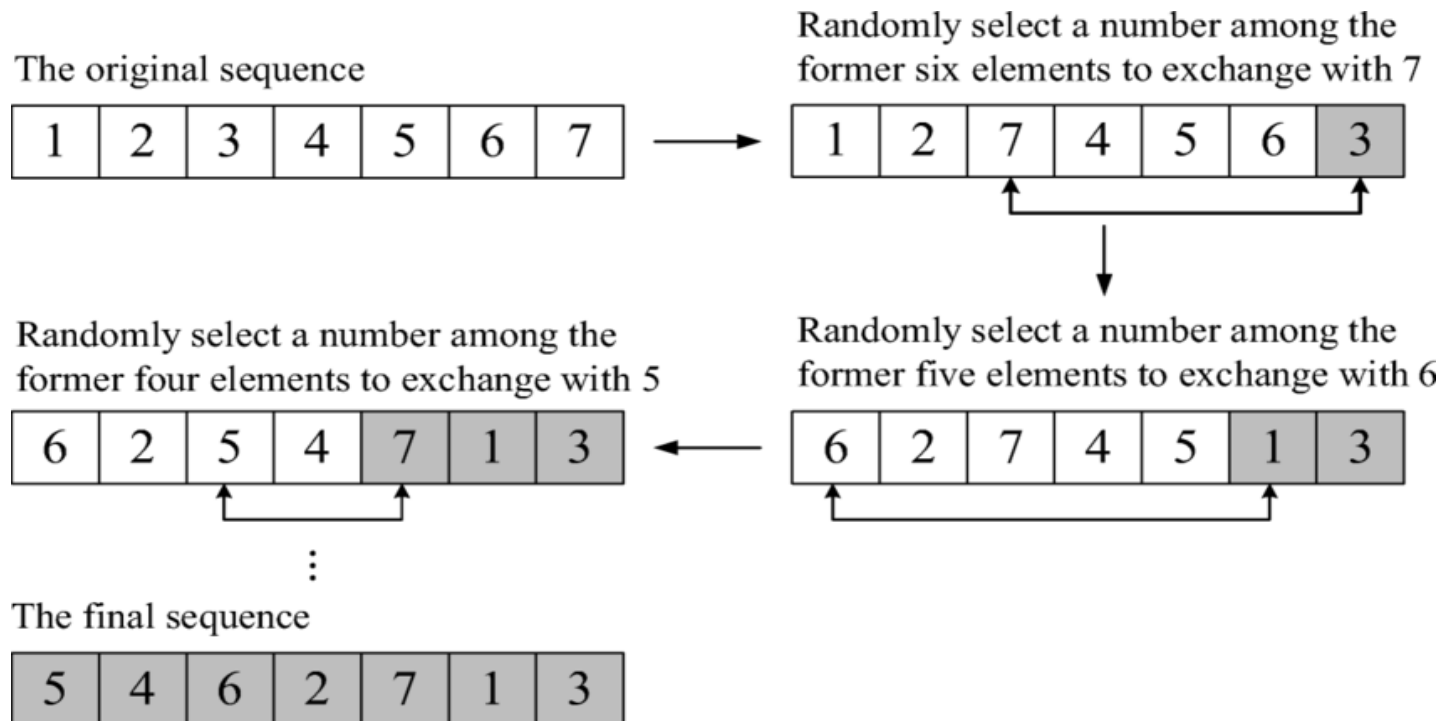
Uma possível solução para o problema é representada por meio de um array de tamanho $|M|$. Cada posição i do array corresponde a um vértice de M . Assim, cada posição contém a informação sobre qual vértice de L está conectado com o respectivo vértice $i \in M$, além de um valor indicando a distância da aresta que realiza a ligação. Exemplo:

```
solucao[25].vertice_L = 47
```

```
solucao[25].distancia = 180.00
```


Construção Gulosa-Randomizada


Inicialização de uma solução é feita de forma aleatória, utilizando o algoritmo de Fisher-Yates.





Construção Gulosa-Randomizada

Após a geração de todas as possíveis soluções iniciais (segundo o parâmetro `construcoes`), ocorre um ordenamento das soluções geradas com base no valor da sua função objetivo. Dentre algumas das melhores soluções iniciais geradas (de acordo com o parâmetro `candidatos`), uma é escolhida de forma aleatória para iniciar a busca local.



Geração de Vizinhança para Busca Local

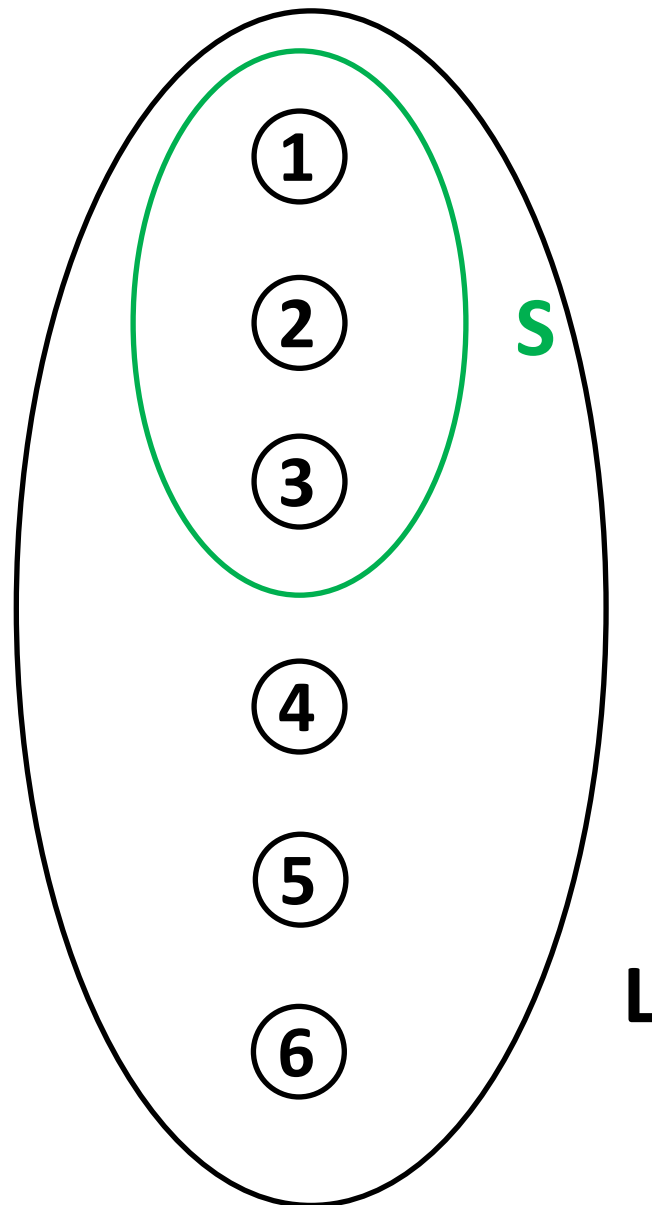


Vizinhança Determinística

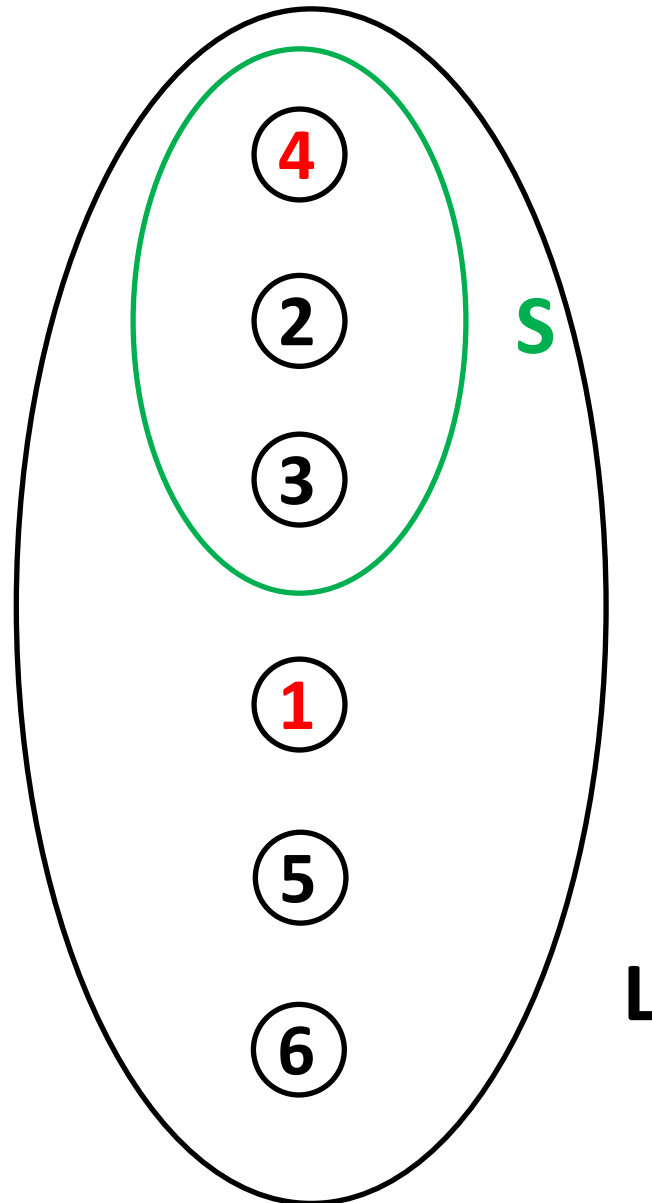
- Em um determinado ponto da busca, a solução corrente irá conter uma determinada seleção de vértices.
- Essa forma de geração de vizinhanças é feita permutando, de forma individual, **cada vértice selecionado por cada vértice não selecionado**.
- Logo, em cada passo da busca local temos um número de vizinhos igual a

$$(|L| - l) * l$$

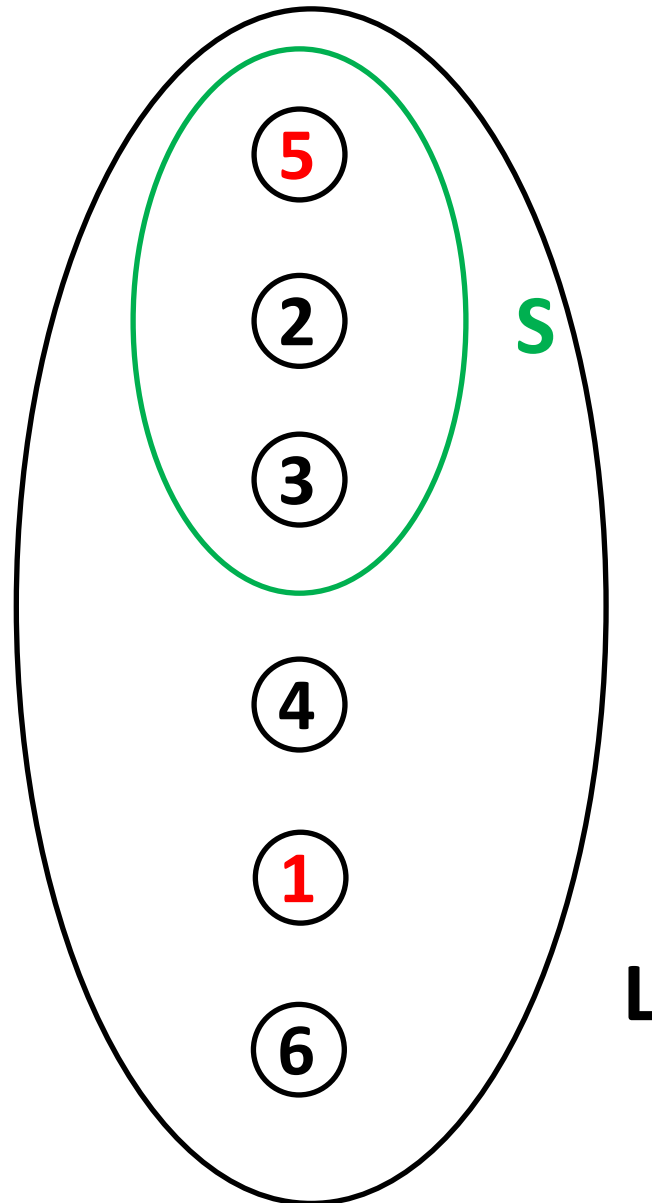
Vizinhança Determinística



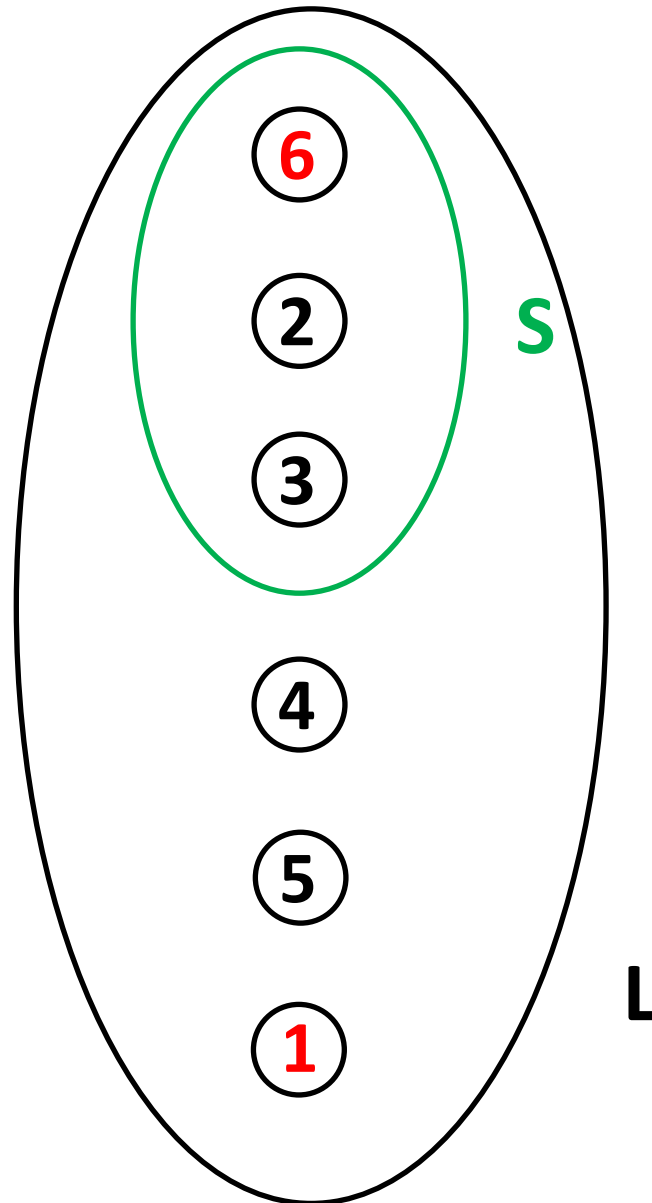
Vizinhança Determinística



Vizinhança Determinística



Vizinhança Determinística



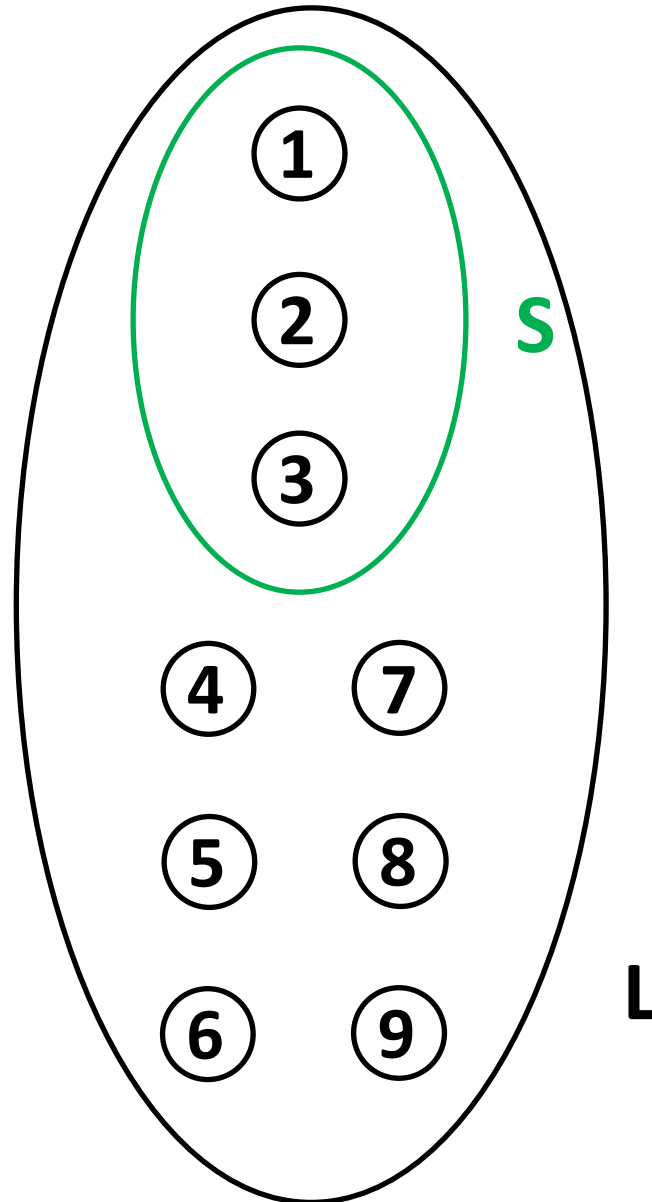


Vizinhança Estocástica

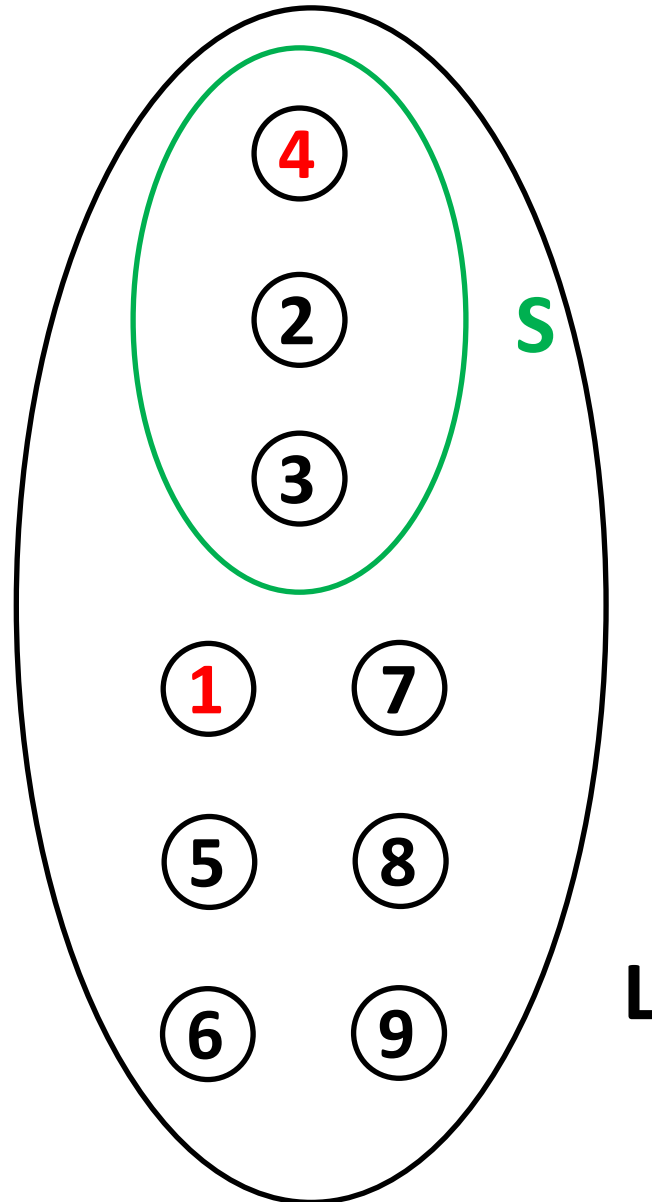
- Essa forma de geração de vizinhanças é feita inserindo, **cada vez mais vértices não-selecionados no conjunto de vértices selecionados.**
- Ao final do processo, todos os vértices que não foram selecionados são “embaralhados” (novamente, com algoritmo de Fisher-Yates)
- Logo, em cada passo da busca local temos um número de vizinhos igual a

$$\min((|L| - l), l)$$

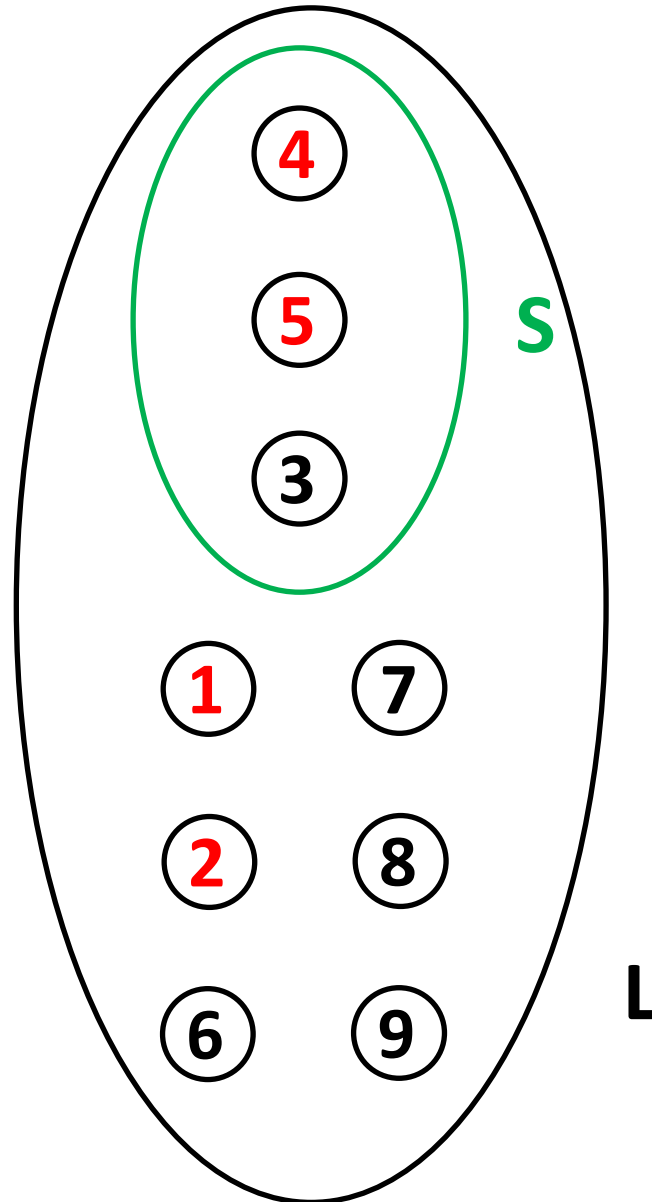
Vizinhança Estocástica



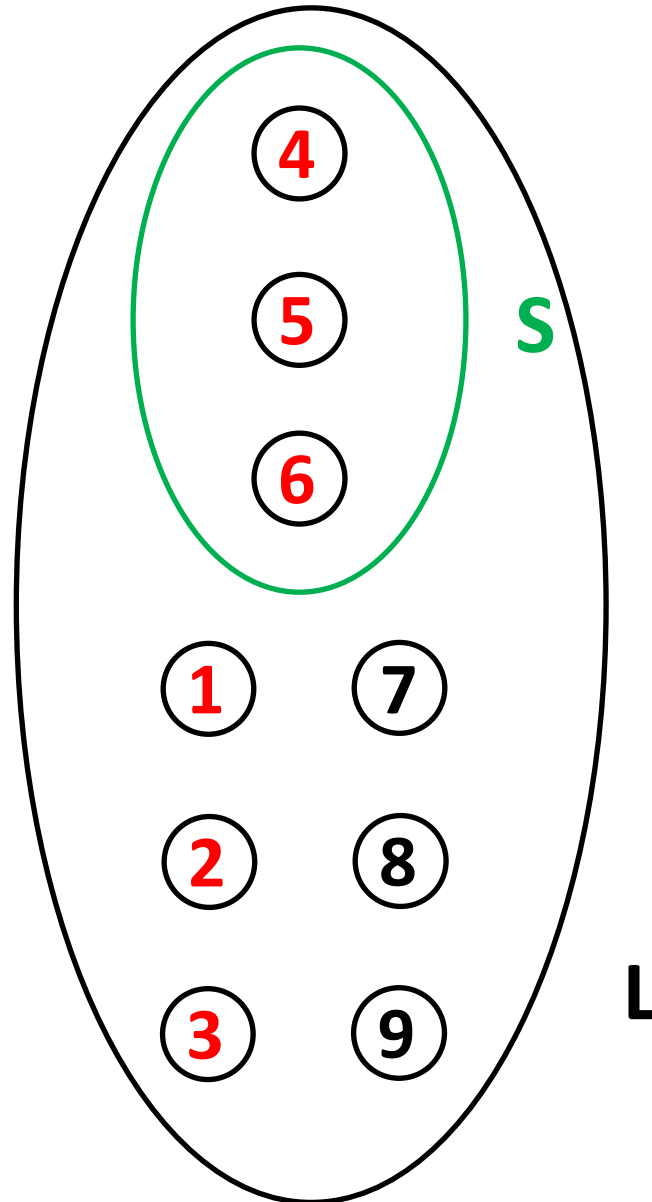
Vizinhança Estocástica



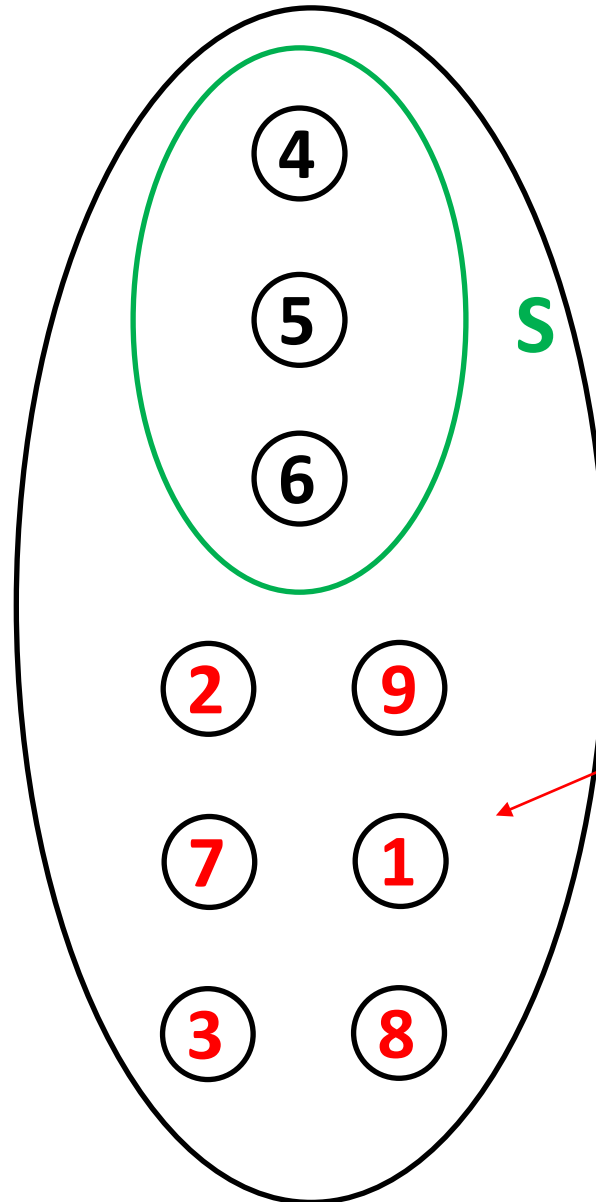
Vizinhança Estocástica



Vizinhança Estocástica



Vizinhança Estocástica



Embaralhamento
de vértices




Resultados Obtidos via Solvers CPLEX e GLPK



Resultados obtidos com CPLEX e GLPK

- O programa inteiro foi implementado nos solvers CPLEX e GLPK. Para instâncias menores foi possível obter os resultados corretos, mas para as instâncias disponibilizadas não foi encontrado nenhum resultado no tempo limite de 1 hora.



Resultados Obtidos com a Implementação

Resultados com Vizinhança Determinística

- Com a utilização de vizinhança determinística, foram realizados dez experimentos com cada uma das dez instâncias.
- `inicios = 1`
- `construcoes = 200`
- `candidatos = 10`

Resultados com Vizinhança Determinística

Tabela 1 - Comparações com Melhores Valores Conhecidos			
Instância	Melhor Valor com Metaheurística (VM)	Melhor Valor Conhecido (BKV)	Desvio Percentual (VM e BKV)
mdmt39.112.A	6123	5935	3,07
mdmt39.112.B	5805	6198	-6,77
mdmt39.225.A	4196	4654	-10,92
mdmt39.225.B	4326	4260	1,53
mdmt40.56.A	7979	8211	-2,91
mdmt40.56.B	8165	8022	1,75
mdmt40.112.A	6102	6271	-2,77
mdmt40.112.B	6210	6198	0,19
mdmt40.225.A	4441	4550	-2,45
mdmt40.225.B	4488	4492	-0,09

Resultados com Vizinhança Determinística

Tabela 2 – Testes com a instância mdmt39.112.A				
Solução Inicial	Solução Final	Desvio Percentual (SI e SF)	Tempo Computacional Metaheurística	Seed
3620	6049	-67,10	6m1s	6021
3608	6051	-67,71	6m34s	7200
3611	6102	-68,98	6m9s	8487
3570	6049	-69,44	6m14s	9692
3612	6055	-67,64	6m45s	916
3596	6123	-70,27	8m58s	5051
3639	6105	-67,77	9m22s	6821
3617	6085	-68,23	8m41s	8656
3605	6117	-69,68	8m37s	357
3569	5950	-66,71	8m43s	2046
Média aritmética das dez soluções finais: 6068.60				

Resultados com Vizinhança Determinística

Tabela 3 – Testes com a instância mdmt40.56.B				
Solução Inicial	Solução Final	Desvio Percentual (SI e SF)	Tempo Computacional Metaheurística	Seed
5040	8045	-59,62	1m23s	5566
4937	8121	-64,49	1m29s	5837
5001	8165	-63,27	1m19s	6128
5065	7960	-57,16	1m17s	6386
4968	8149	-64,03	1m25s	6640
4954	8021	-61,91	1m47s	3406
4969	8108	-63,17	1m44s	3755
5086	8129	-59,83	1m41s	4095
5077	8102	-59,58	1m49s	4425
4975	8149	-63,80	1m52s	4781
Média aritmética das dez soluções finais: 8094.90				



Resultados com Vizinhança Estocástica

- Com a utilização de vizinhança determinística, foram realizados cinco experimentos com cada uma das dez instâncias.
- `inicios = 10.000`
- `construcoes = 200`
- `candidatos = 10`

Resultados com Vizinhança Estocástica

Tabela 4 – Testes com a instância mdmt39.112.A				
Solução Inicial	Solução Final	Desvio Percentual (SI e SF)	Tempo Computacional Metaheurística	Seed
3712	4078	-9,86	17m46s	5626
3572	4113	-15,15	15m09s	8545
3589	4107	-14,43	15m01s	1513
3589	4068	-13,35	14m34s	1688
3687	4090	-10,93	14m34s	4542
Média aritmética das cinco soluções finais: 4091.20				

Resultados com Vizinhança Estocástica

Tabela 5 – Testes com a instância mdmt39.112.B				
Solução Inicial	Solução Final	Desvio Percentual (SI e SF)	Tempo Computacional Metaheurística	Seed
3709	4174	-12,54	14m32s	7396
3751	4143	-10,45	14m39s	244
3700	4109	-11,05	14m33s	3114
3770	4114	-9,12	14m36s	5965
3718	4165	-12,02	14m35s	8825
Média aritmética das cinco soluções finais: 4141.00				



Comparação entre Vizinhanças

Instância	Melhor Valor com Vizinhança Determinística	Melhor Valor com Vizinhança Estocástica	Desvio Percentual (VD e VE)
mdmt39.112.A	6123	4113	32,83
mdmt39.112.B	5805	4174	28,10
mdmt39.225.A	4196	3141	25,14
mdmt39.225.B	4326	3175	26,61
mdmt40.56.A	7979	5735	28,12
mdmt40.56.B	8165	5791	29,08
mdmt40.112.A	6102	4457	26,96
mdmt40.112.B	6210	4444	28,44
mdmt40.225.A	4441	3362	24,30
mdmt40.225.B	4488	3431	23,55



Conclusões



Conclusões

- Mesmo nos casos em que a busca local foi iniciada a partir de soluções iniciais que não correspondem às melhores soluções geradas no processo guloso-randomizado, foi possível obter resultados muito interessantes.
- Fica evidenciada, assim, a importância do processo de inicialização que é proposto pela metaheurística GRASP, que **propõe a exploração de espaços que não seriam originalmente investigados**, caso fossem utilizadas heurísticas convencionais.



Conclusões

- A superioridade da busca local com utilização de vizinhança determinística, em relação à vizinhança estocástica, ficou bastante evidente.
- Dessa forma, foi possível verificar experimentalmente a importância da **geração de uma vizinhança de forma logicamente coerente**, com um número vasto de possíveis transições, para a obtenção de máximos locais mais interessantes durante a busca local referente à metaheurística.



Conclusões

- Mesmo nos seis casos em que não foi possível ultrapassar os melhores valores conhecidos, o desvio percentual corresponde a valores próximos de zero, evidenciando a eficácia dos experimentos realizados.
- Assim, considera-se que o trabalho foi bem-sucedido na obtenção de resultados promissores.



Referências

Blum C.; Roli A. (2003) **Metaheuristics in combinatorial optimization: Overview and conceptual comparison.** ACM Comput. Surv., Vol 35, N. 3, September , pp. 268-308.

Resende M.; Ribeiro C. (2003) **Greedy Randomized Adaptive Search Procedures: Advances and Extensions.** pp 1-2.