Ciclo Hamiltoniano
e
Problema do Caixeiro Viajante

Problema do Caixeiro Viajante

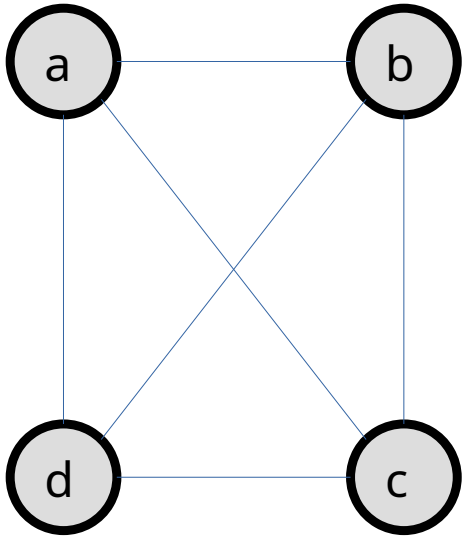
Problema do Caixeiro Viajante

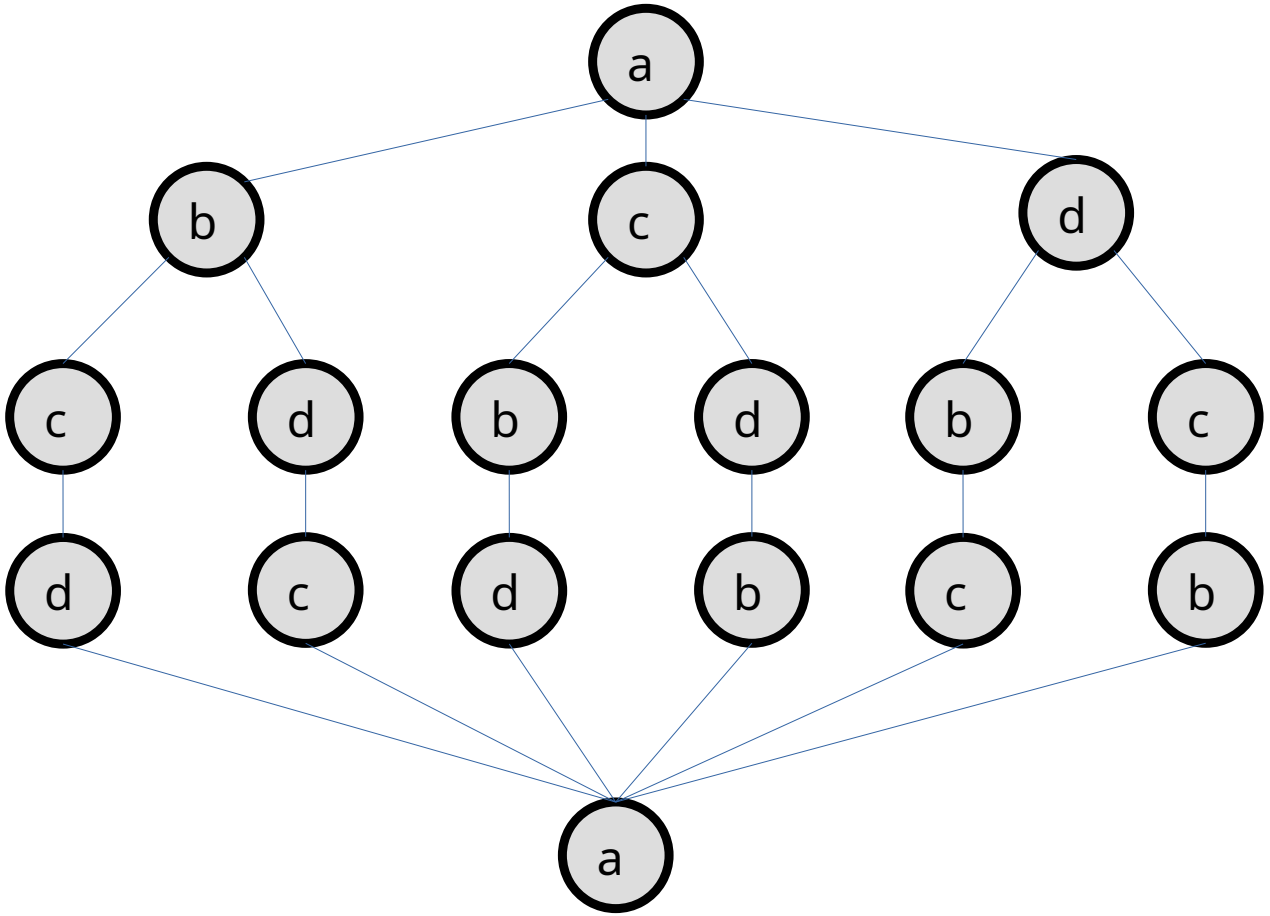
Dado um grafo ponderado $G = (V,E)$, encontrar um ciclo gerador H cuja soma das arestas seja mínima dentre todos os ciclos geradores de G.

*"Dado **N** cidades, achar a caminho mais curto passando por todas as cidades uma única vez."*

Problema do Caixeiro Viajante

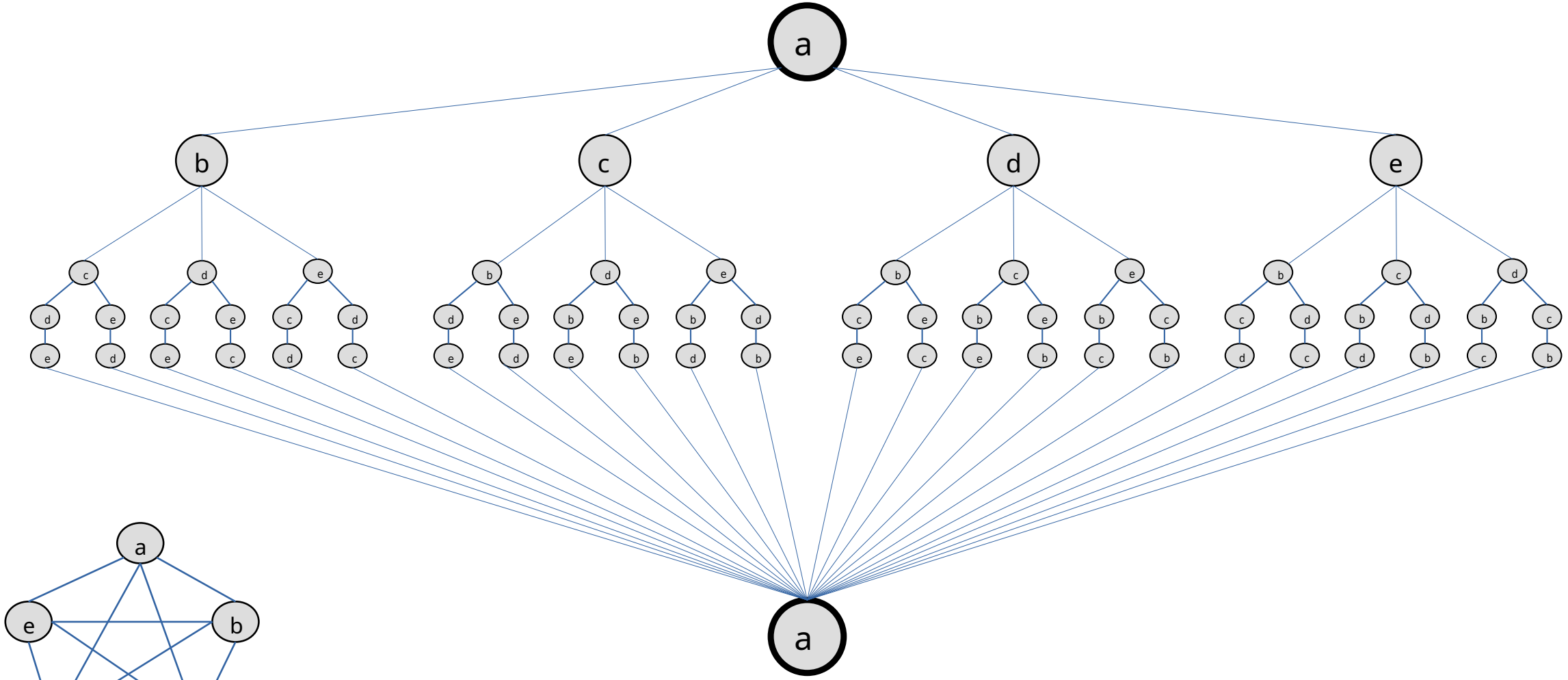
*"Dado **N** cidades, achar a caminho mais curto passando por todas as cidades uma única vez."*





Para N cidades há (N-1)!
Rotas (4-1)! = 3! = 6 rotas

Problema do Caixeiro Viajante



Para N cidades há (N-1)!
Rotas (5-1)! = 4! = 24 rotas

Algoritmos Aproximativos

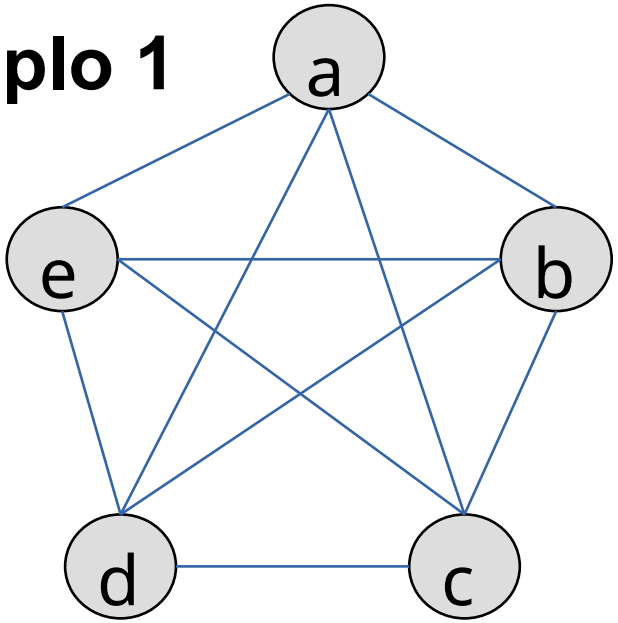
GRASP (FEO e RESENDE)

```
procedure grasp()
1     InputInstance();
2     for GRASP stopping criterion not satisfied →
3           ConstructGreedyRandomizedSolution(Solution);
4           LocalSearch(Solution);
5           UpdateSolution(Solution,BestSolutionFound);
6     rof;
7     return(BestSolutionFound)
end grasp;
```

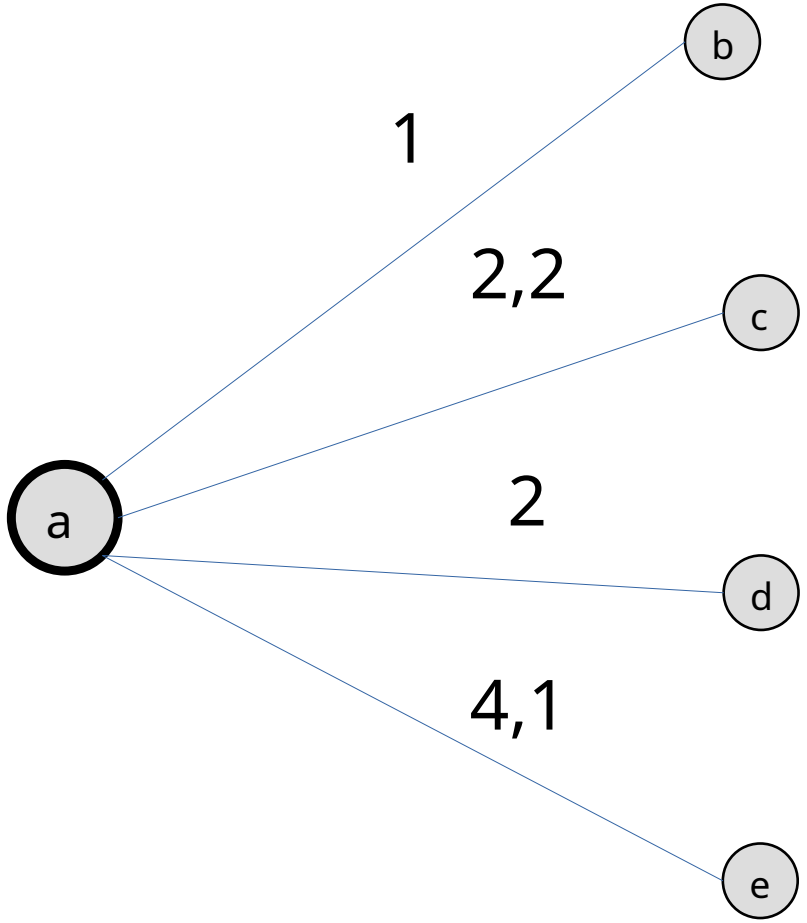
Figure 1. A generic GRASP pseudo-code

Algoritmos Aproximativos

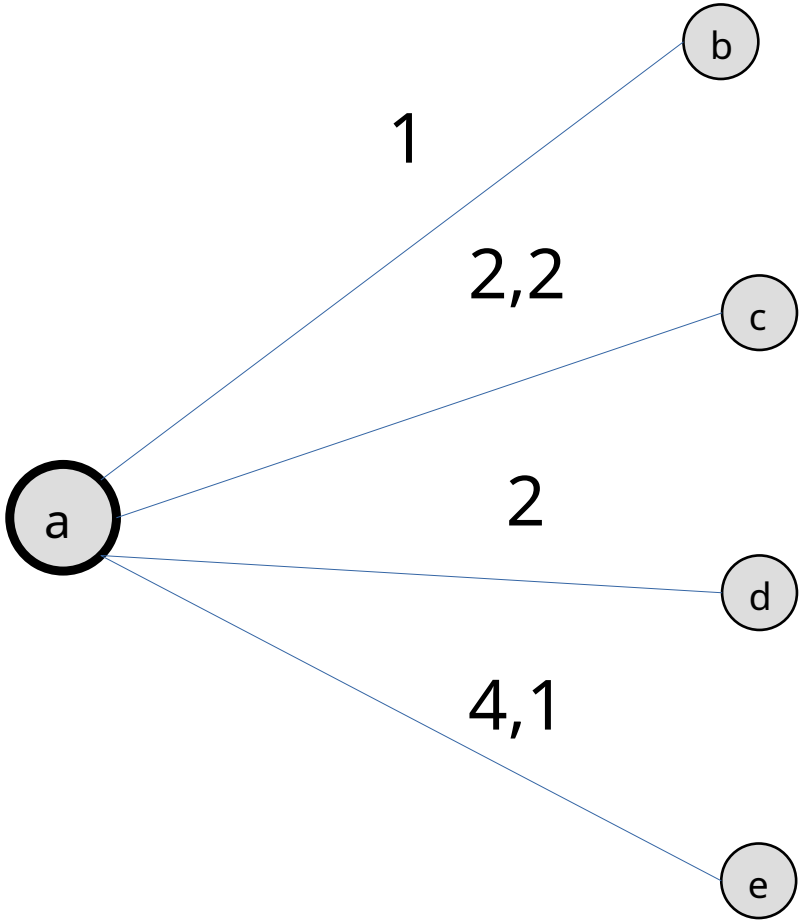
Exemplo 1



c_{ij}	a	b	c	d	e
a	-	1	2,2	2	4,1
b	1	-	1,4	2,2	4
c	2,2	1,4	-	2,2	3,2
d	2	2,2	2,2	-	2,2
e	4,1	4	3,2	2,2	-

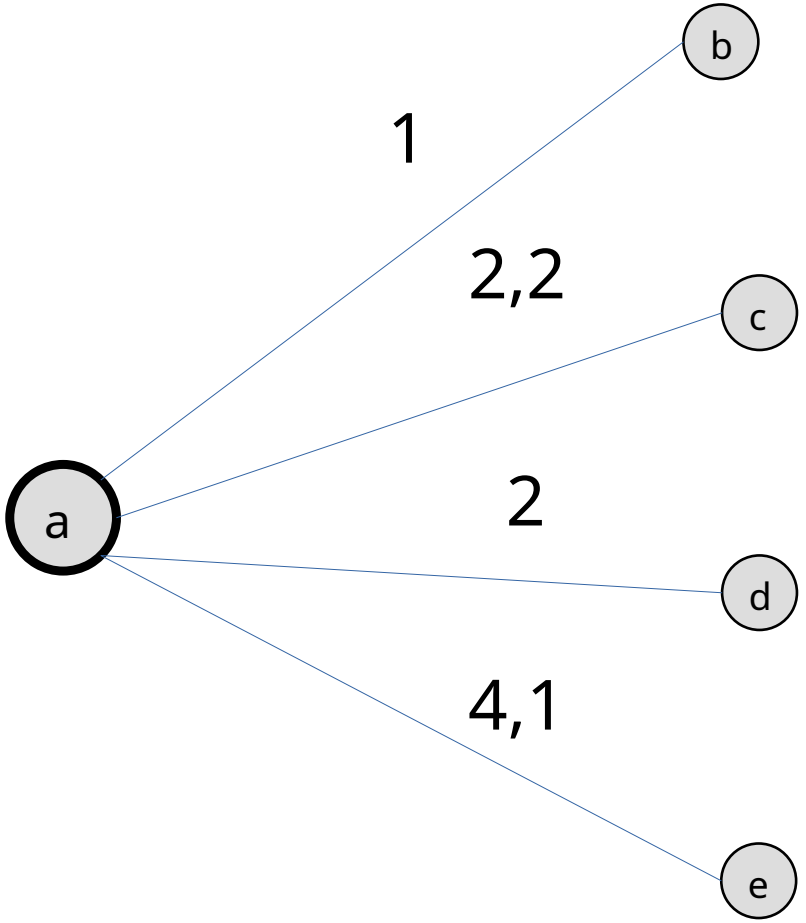


Algoritmos Aproximativos



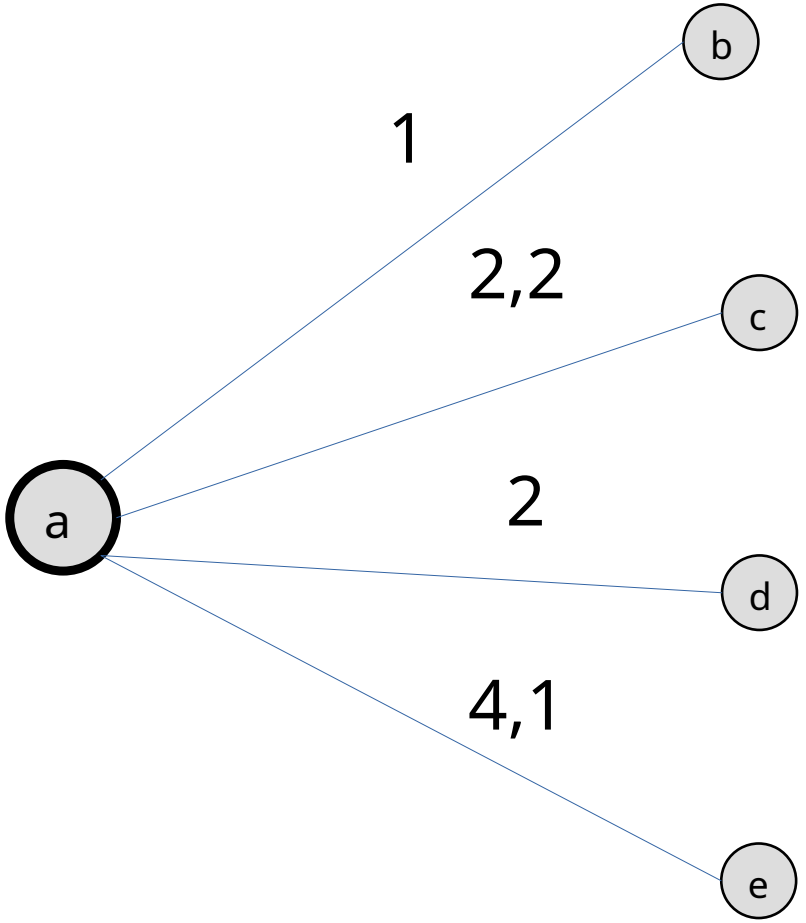
Como determinar a probabilidade de escolha atrelada à distância?

Algoritmos Aproximativos



Lista de Candidatos (ordenada)

Algoritmos Aproximativos



Lista Restrita de Candidatos (LRC)

- *Construção Baseada em Tamanho*
- *Construção Baseada em Qualidade*

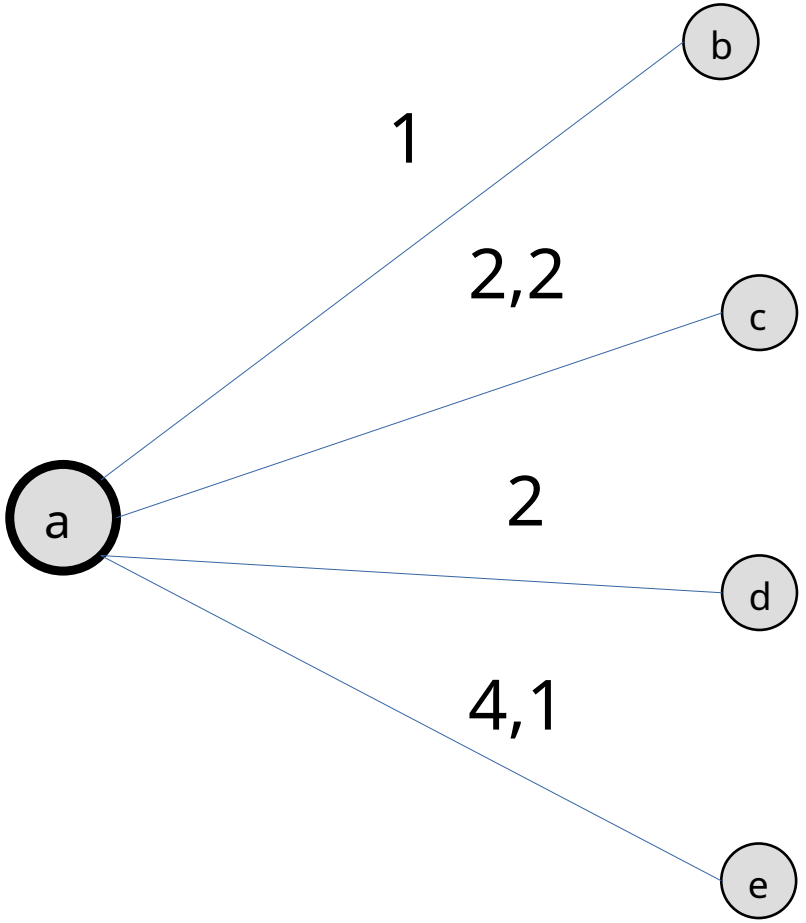
Parâmetro para controle da lista: α

$$0 \leq \alpha \leq 1$$

α = 0 (construção gulosa)

α = 1 (construção aleatória)

Algoritmos Aproximativos



Lista Restrita de Candidatos (LRC)

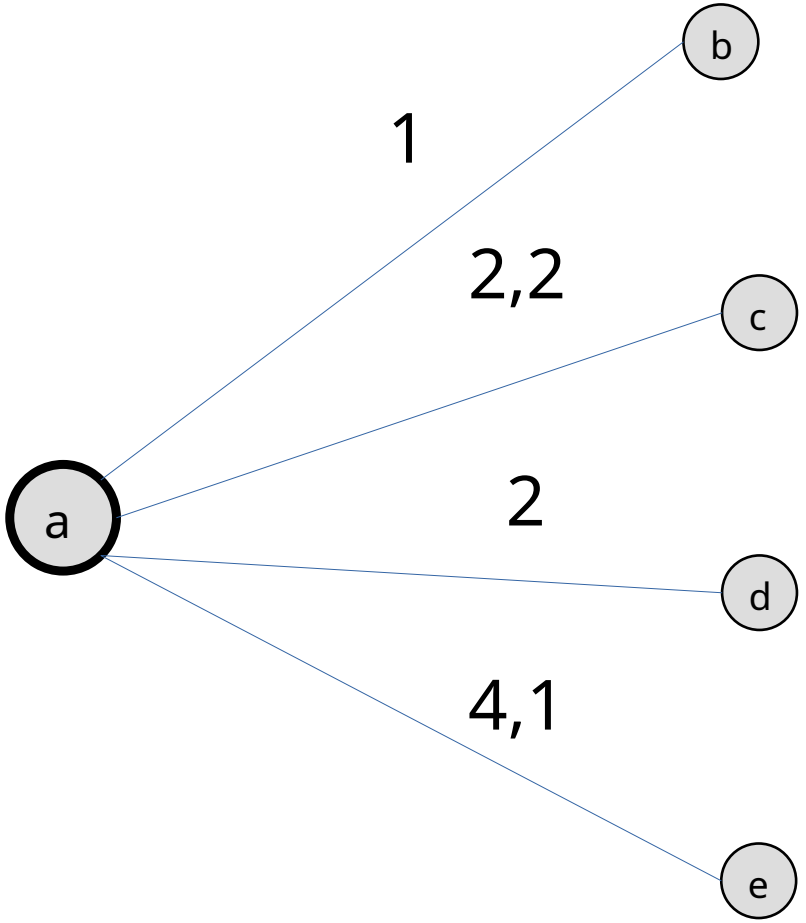
- *Construção Baseada em Tamanho*

$0 \leq \alpha \leq 1$

α = 0 (construção gulosa)

α = 1 (construção aleatória)

Algoritmos Aproximativos



Lista Restrita de Candidatos (LRC)

- *Construção Baseada em Qualidade*
- *Cidades com distância menor ou igual* **min_dist+α·(max_dist−min_dist)** *são incluídas na RCL.*

$$0 \leq \alpha \leq 1$$

α = 0 (construção gulosa)

α = 1 (construção aleatória)

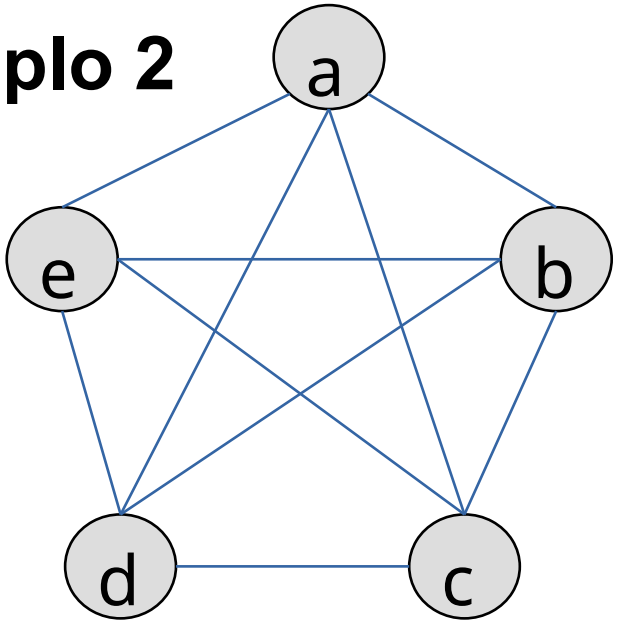
Algoritmos Aproximativos

```
procedure ConstructGreedyRandomizedSolution(Solution)
1     Solution = {};
2     for Solution construction not done  →
3           MakeRCL(RCL);
4           s = SelectElementAtRandom(RCL);
5           Solution = Solution ∪ {s};
6           AdaptGreedyFunction(s);
7     rof;
end ConstructGreedyRandomizedSolution;
```

Figure 2. GRASP construction phase pseudo-code

Algoritmos Aproximativos

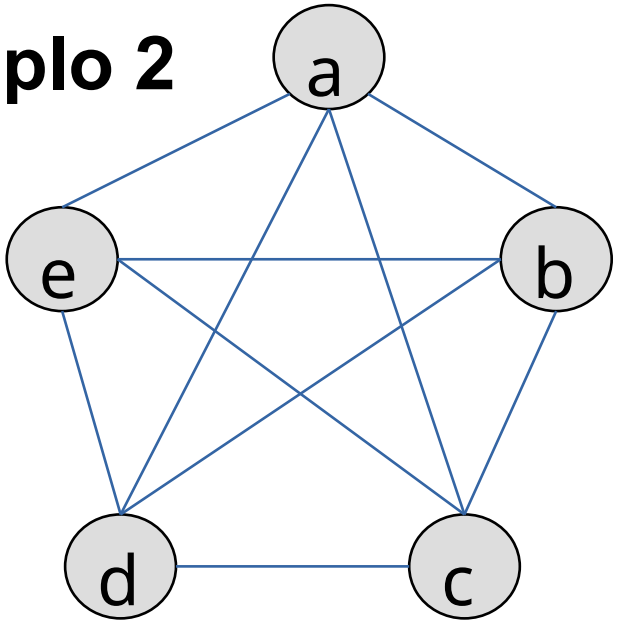
Exemplo 2



c_{ij}	a	b	c	d	e
a	-	1	2,2	2	4,1
b	1	-	1,4	2,2	4
c	2,2	1,4	-	2,2	3,2
d	2	2,2	2,2	-	2,2
e	4,1	4	3,2	2,2	-

Algoritmos Aproximativos

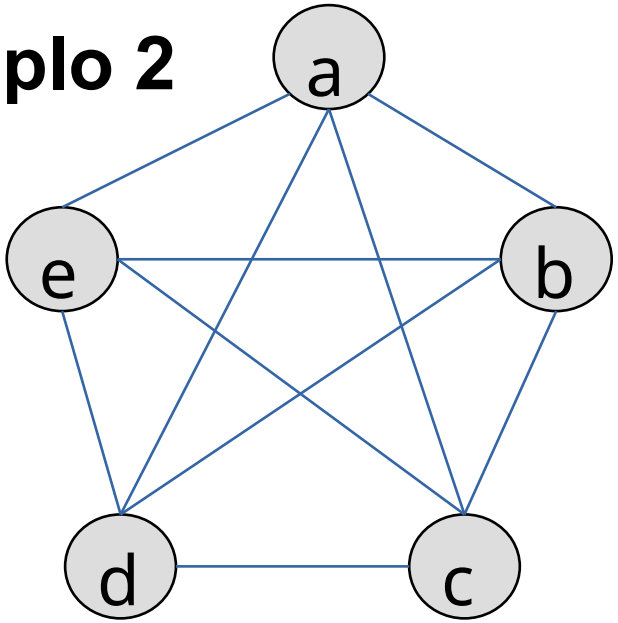
Exemplo 2



c_{ij}	a	b	c	d	e
a	-	1	2,2	2	4,1
b	1	-	1,4	2,2	4
c	2,2	1,4	-	2,2	3,2
d	2	2,2	2,2	-	2,2
e	4,1	4	3,2	2,2	-

Algoritmos Aproximativos

Exemplo 2



c_{ij}	a	b	c	d	e
a	-	1	2,2	2	4,1
b	1	-	1,4	2,2	4
c	2,2	1,4	-	2,2	3,2
d	2	2,2	2,2	-	2,2
e	4,1	4	3,2	2,2	-

Algoritmos Aproximativos

Buscas Locais: Tenta-se trocar as conexões entre as cidades para reduzir o custo total do percurso.

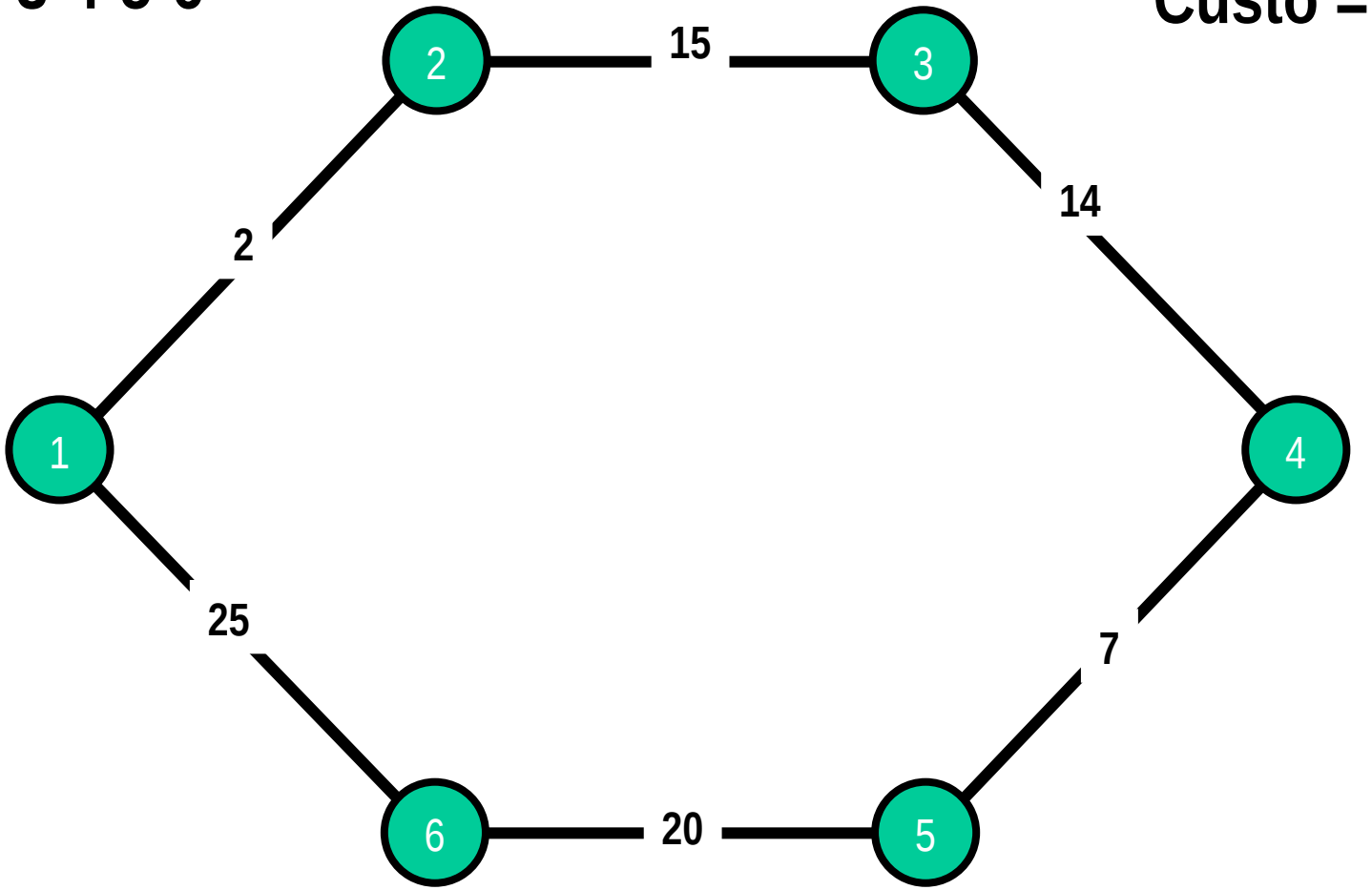
Vizinhança 2-Swap

$$S_0 = \{1,2,3,4,5,6\} \begin{cases} S_1 = \{1,3,2,4,5,6\} \\ S_2 = \{1,2,4,3,5,6\} \\ S_3 = \{1,2,3,5,4,6\} \\ S_4 = \{1,2,3,4,6,5\} \end{cases}$$

Algoritmos Aproximativos

1-2-3-4-5-6

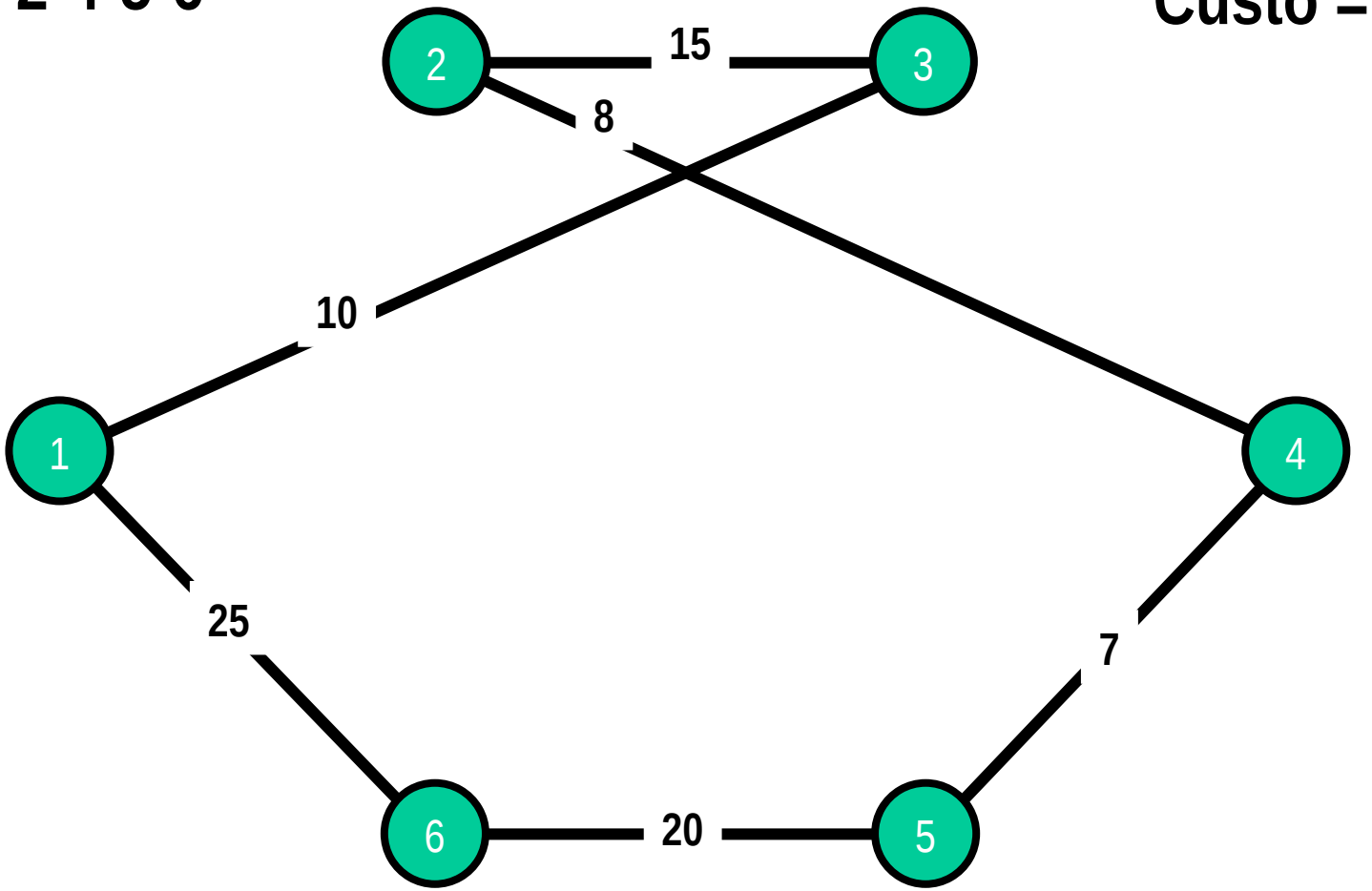
Custo = 83



Algoritmos Aproximativos

1-3-2-4-5-6

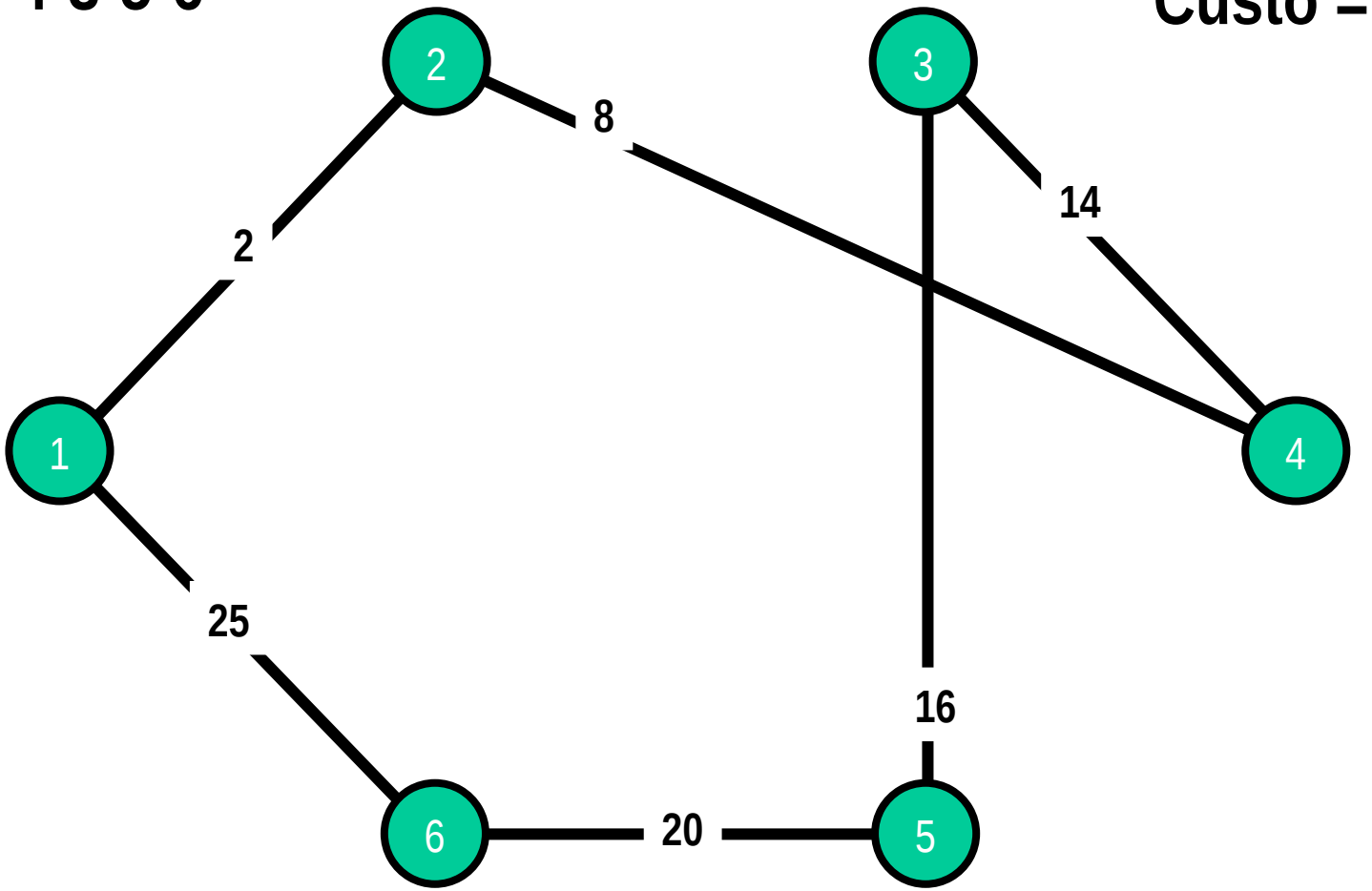
Custo = 85



Algoritmos Aproximativos

1-2-4-3-5-6

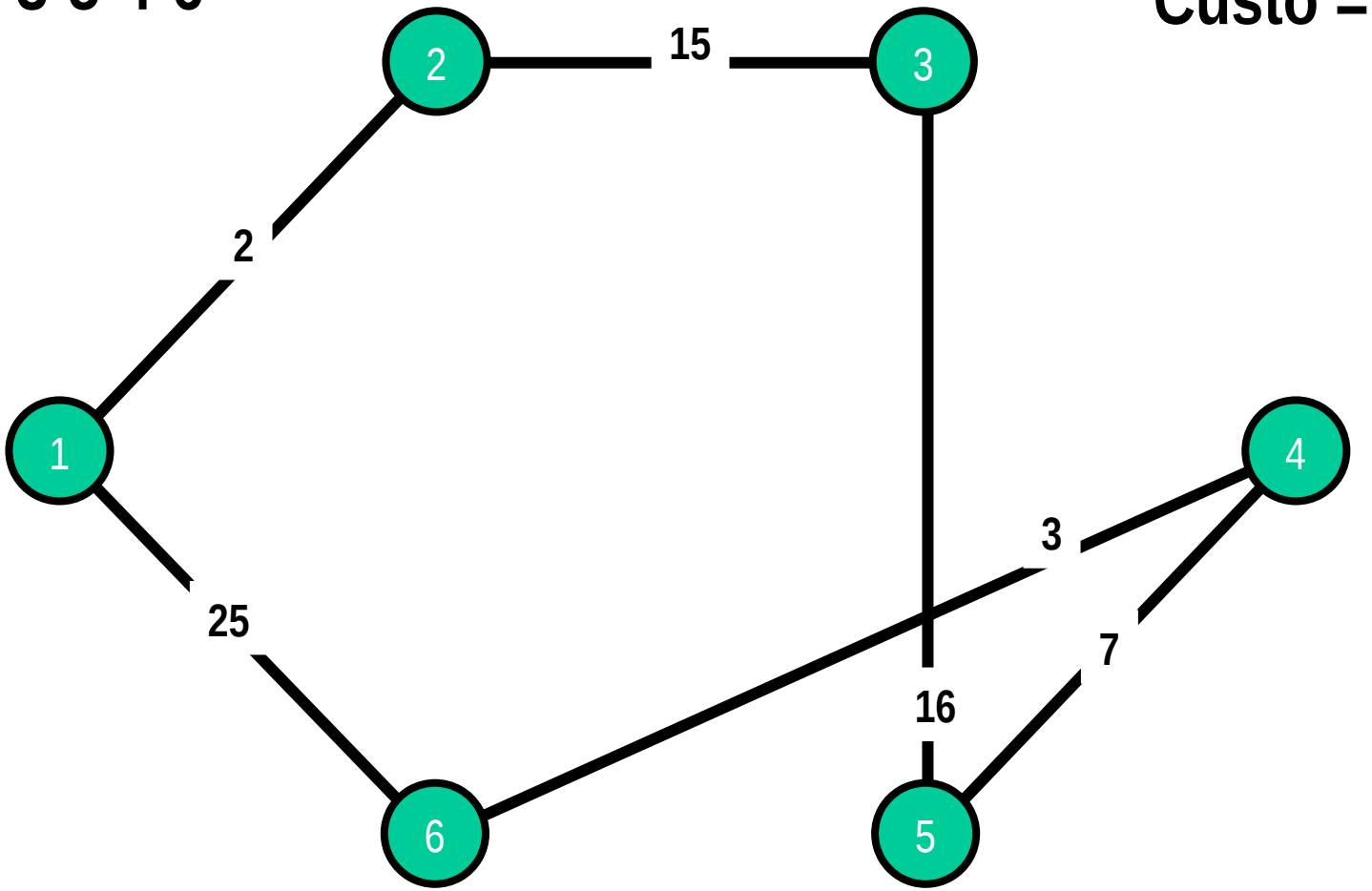
Custo = 85



Algoritmos Aproximativos

1-2-3-5-4-6

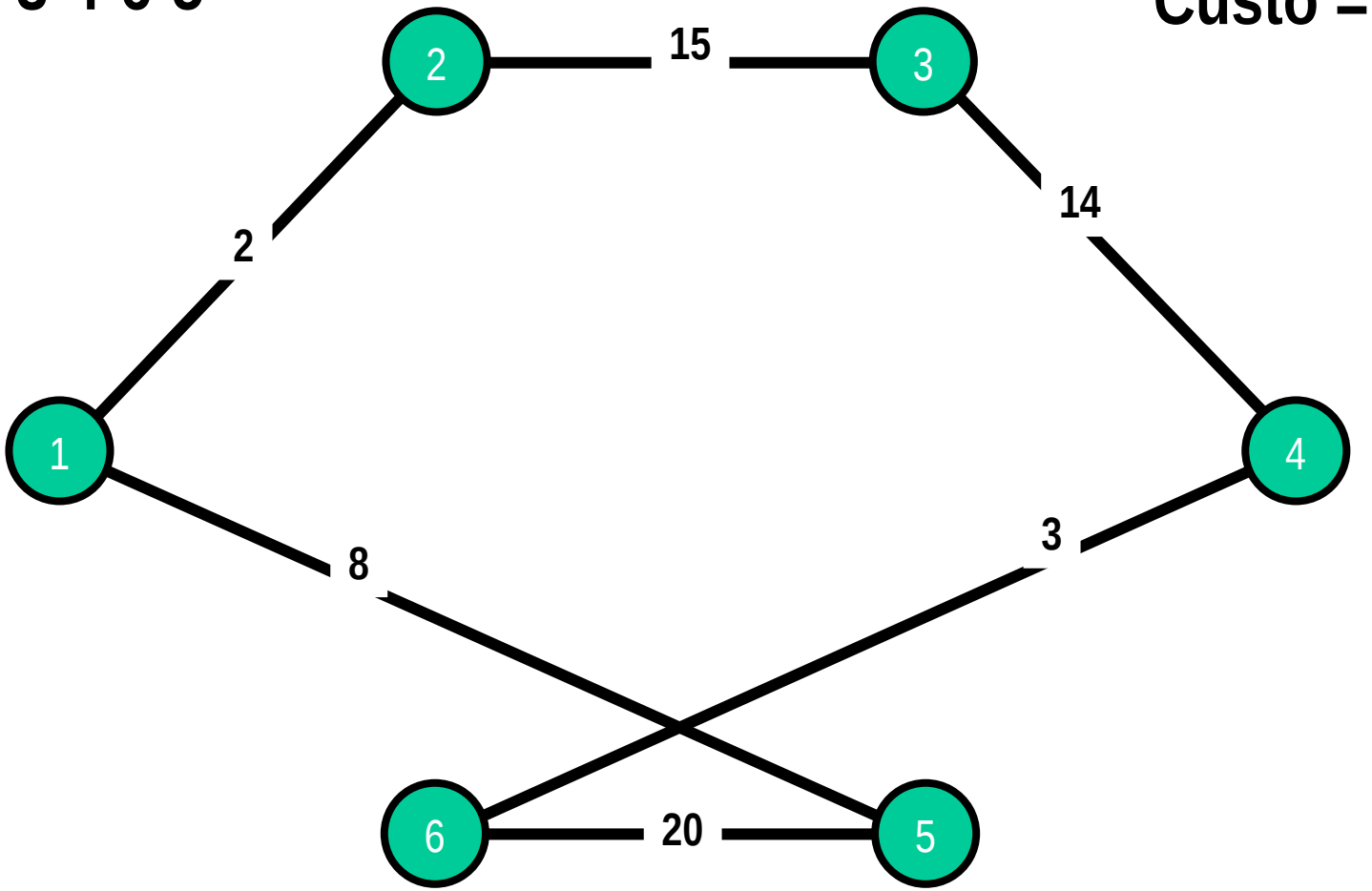
Custo = 68



Algoritmos Aproximativos

1-2-3-4-6-5

Custo = 62



Algoritmos Aproximativos