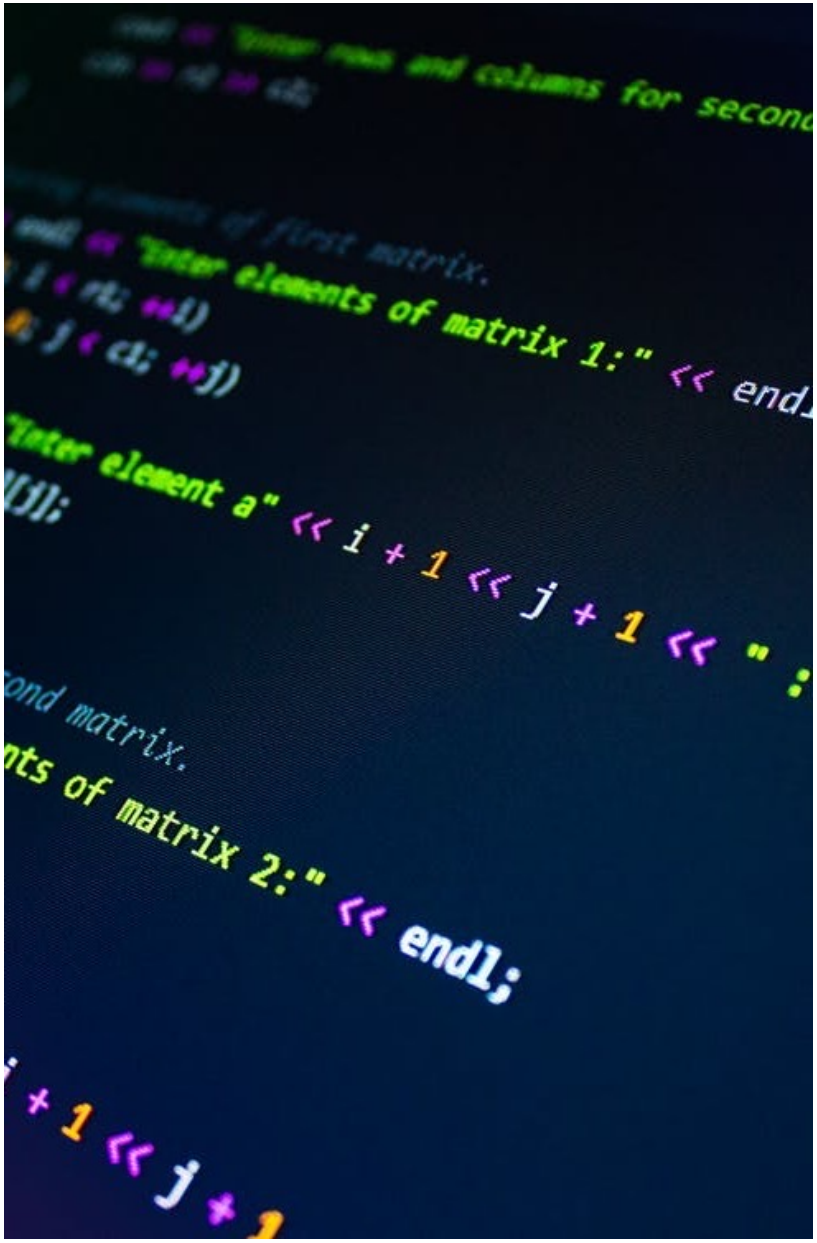
Insper

Supercomputação



Aula - 07

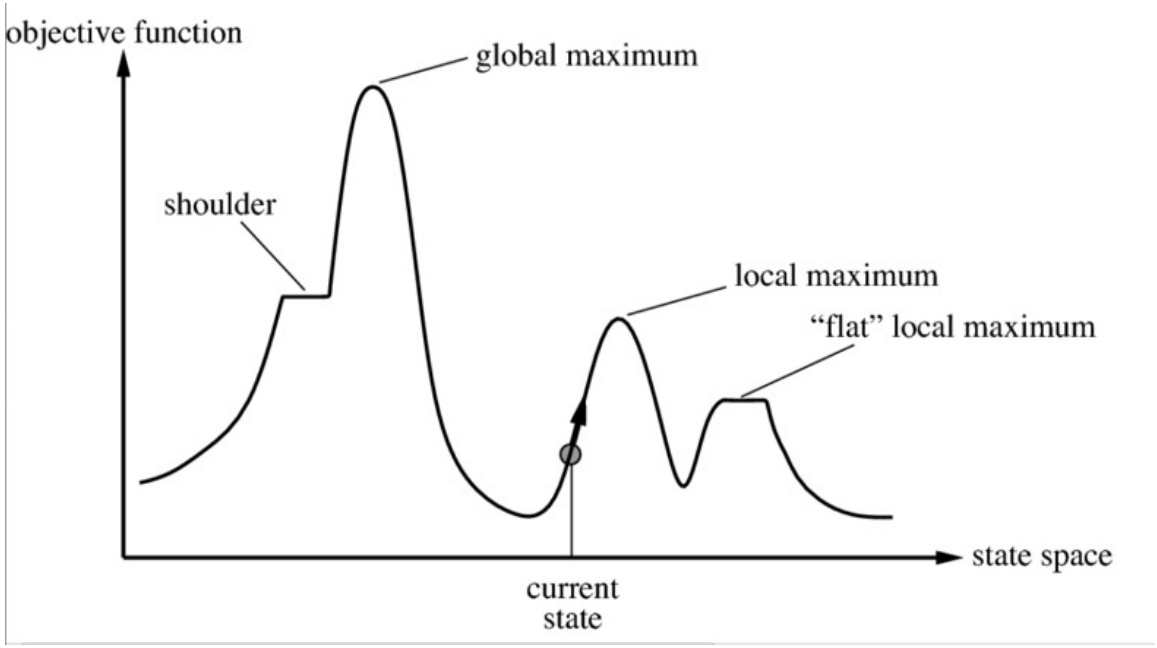
- Busca Local

Recaptulação

- Soluções aleatorizadas
    - Nos permitem balancear entre exploitation vs exploration
- Seria possível fazer uso de aleatoriedade como elemento de varredura no espaço de busca, mas ainda usar algum outro recurso para melhoria da solução encontrada por meio da aleatoriedade?

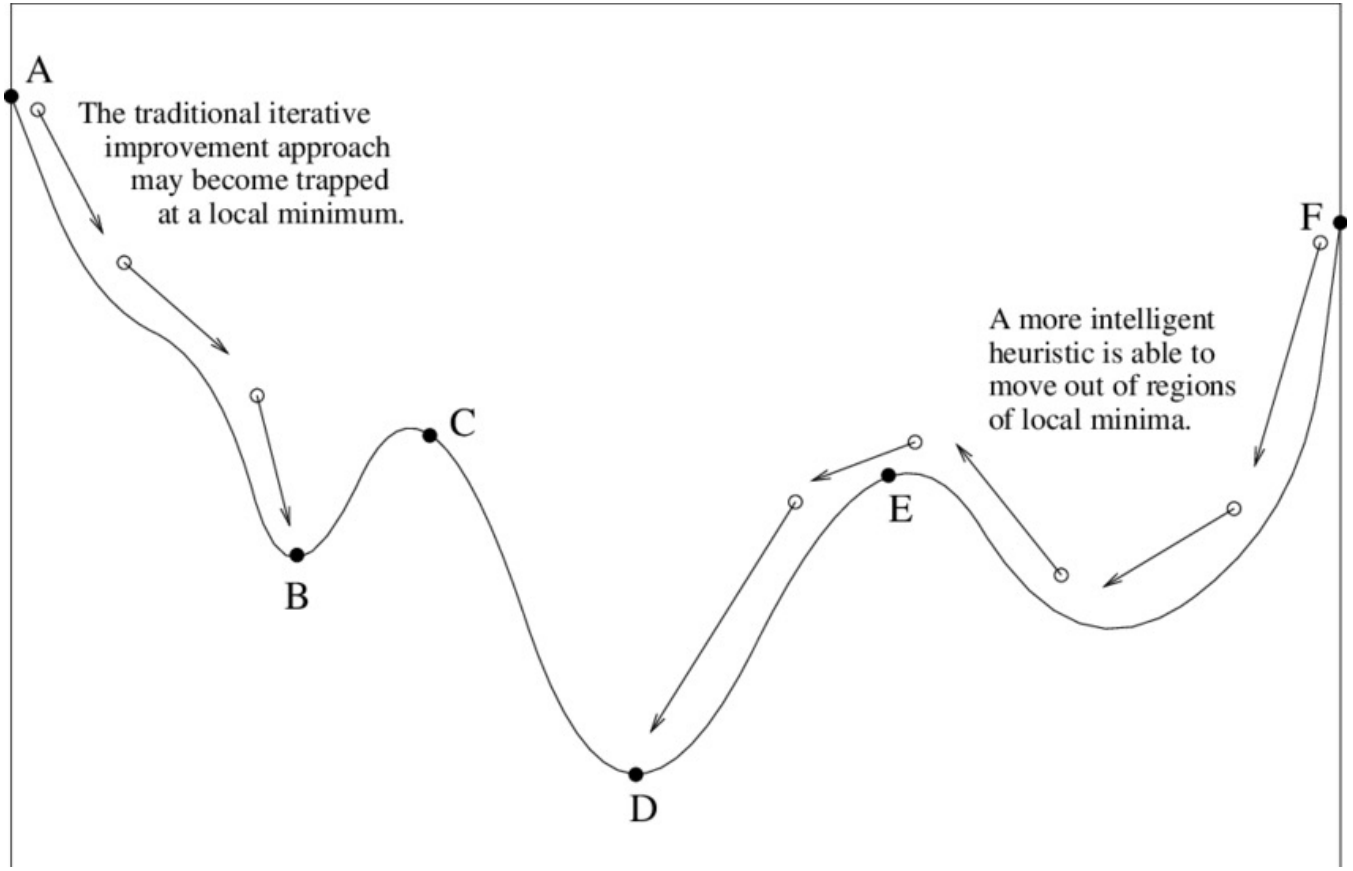
Problemas de otimização

- Em muitos problemas de otimização, o caminho para se atingir o objetivo é irrelevante, desde que a possamos conseguir uma solução para o problema em si



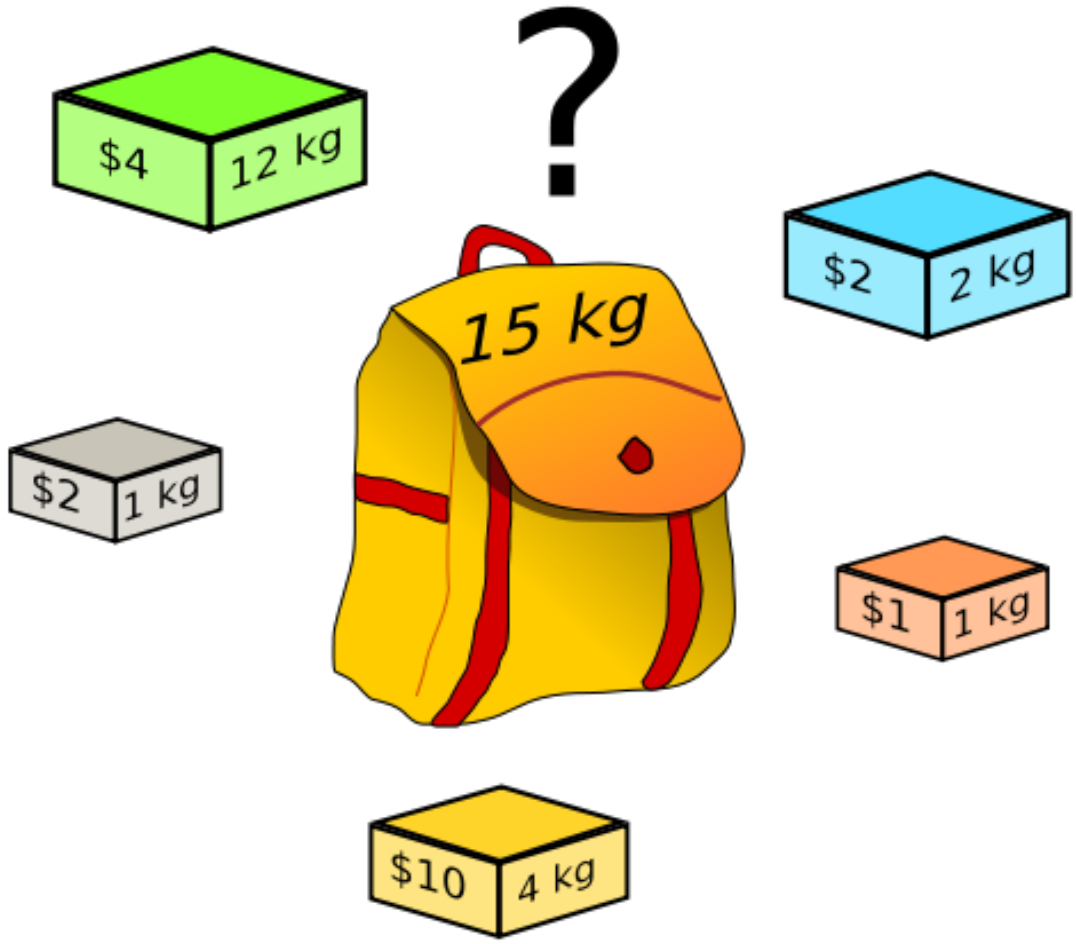
Melhorias após solução aleatória

- Exemplo: minimização



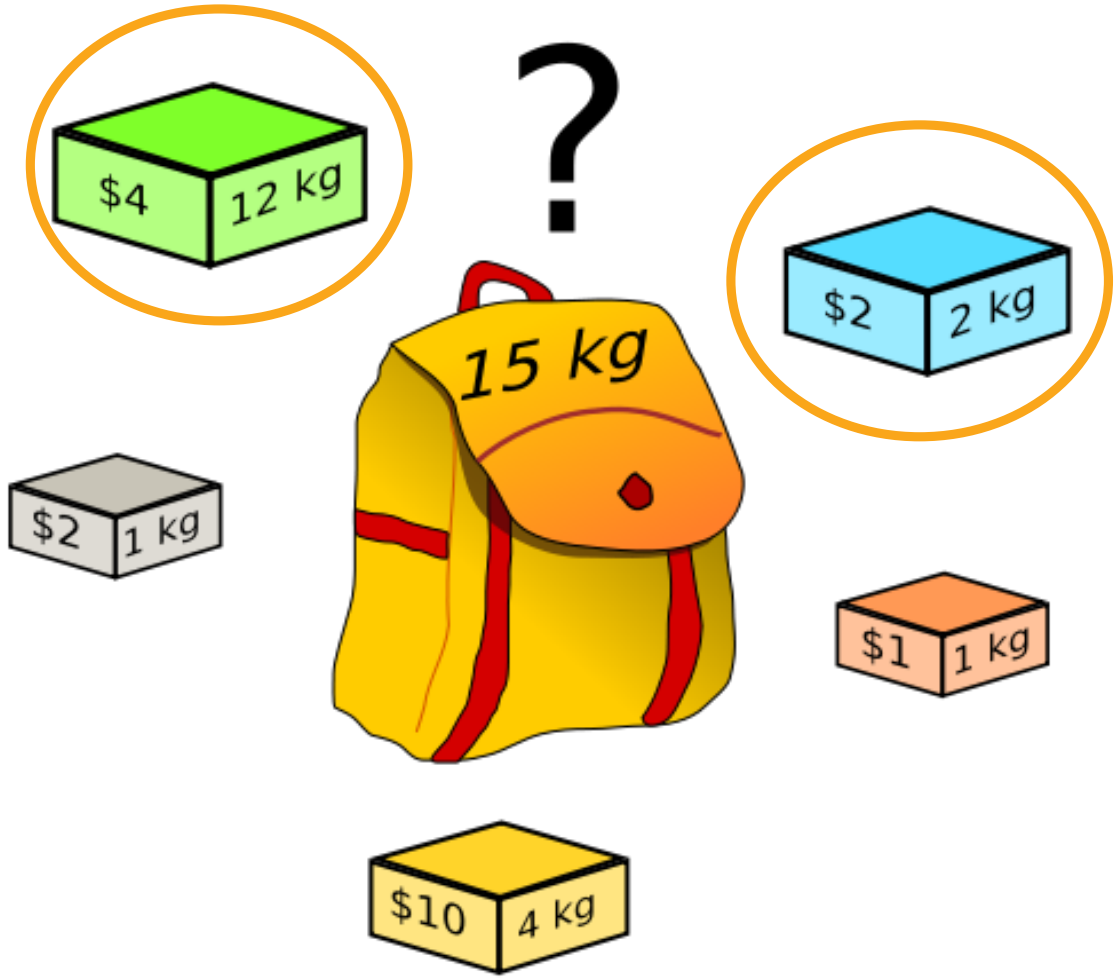
Mochila

- Uma possível solução aleatória:



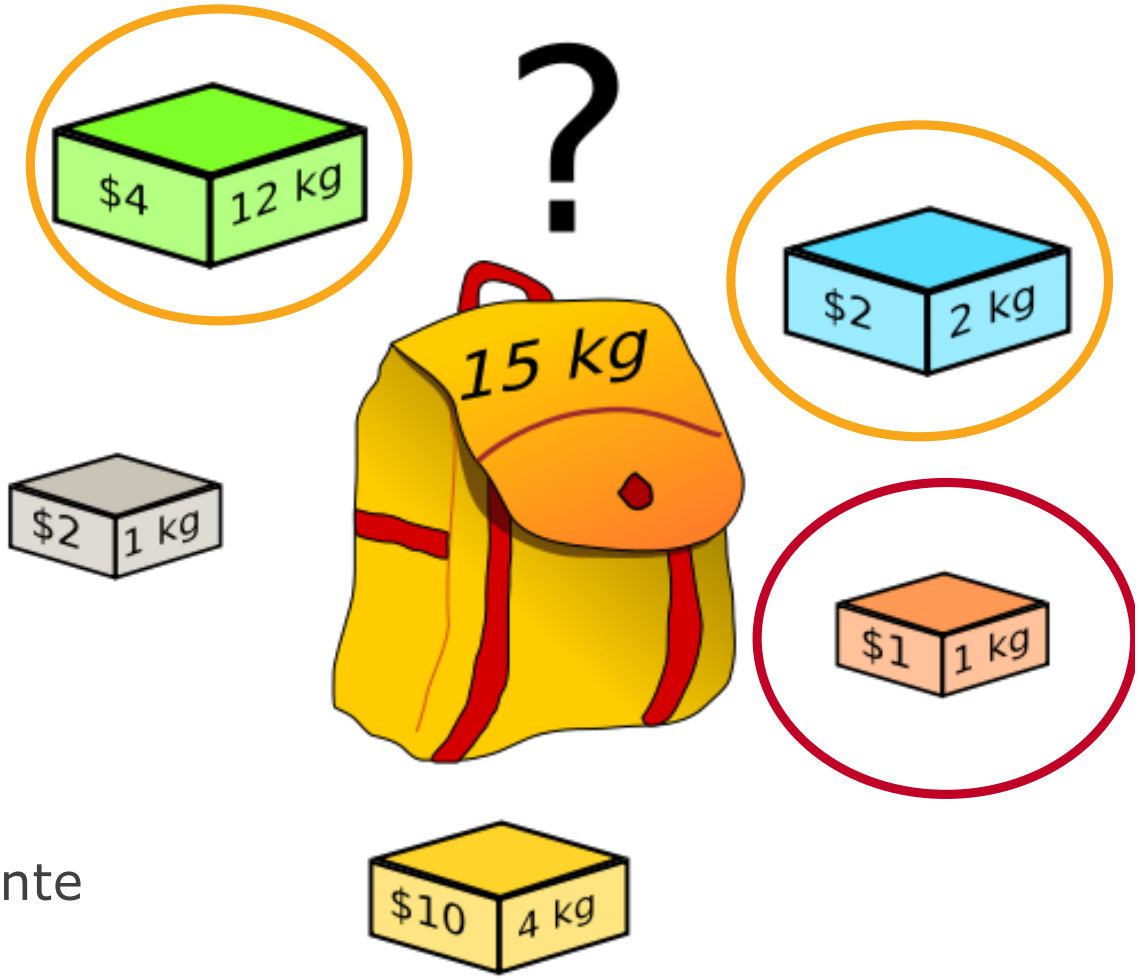
Mochila

- Uma possível solução aleatória:
  - Peso: 14 Kg
    Valor: $6



Mochila

- Uma possível solução aleatória:
  - Peso: 14 Kg
    Valor: $6
- Melhorando:
  - Adicionamos 1 item a mais
    Peso: 15 Kg
    Valor: $7
- Ou seja, conseguimos melhorar uma solução gerada aleatoriamente



Como então podemos melhorar?

- Para a mochila, após gerarmos soluções iniciais aleatórias, podemos fazer duas ações:

1. **Encher a mochila**: verificar se algum objeto não selecionado cabe na mochila

2. **Trocar dois objetos:** verificar se é possível substituir um objeto selecionado por outro de melhor valor que foi deixado de fora

Como então podemos melhorar?

- Para a mochila, após gerarmos soluções iniciais aleatórias, podemos fazer duas ações:

1. **Encher a mochila**: verificar se algum objeto não selecionado cabe na mochila

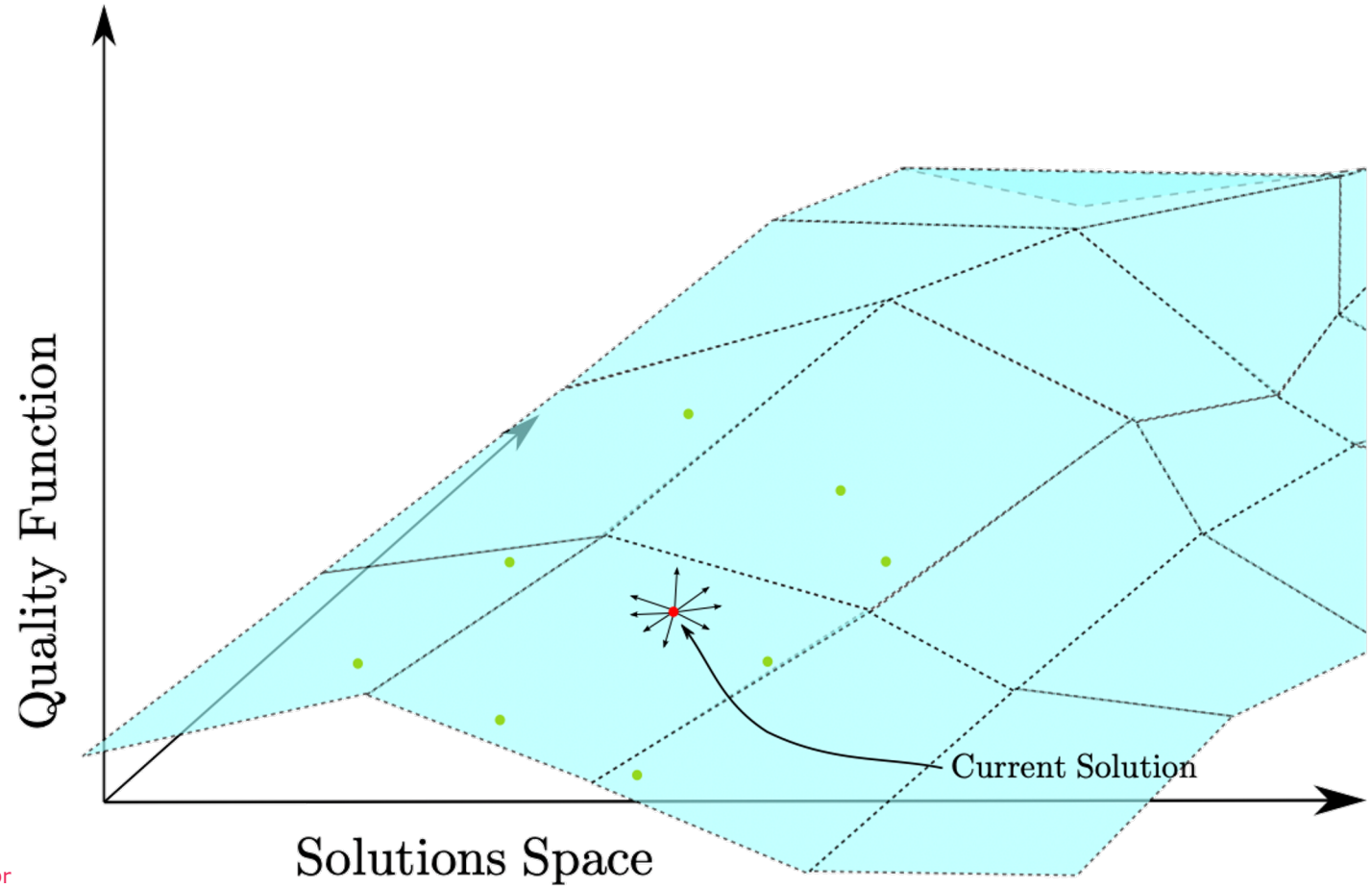
2. **Trocar dois objetos:** verificar se é possível substituir um objeto selecionado por outro de melhor valor que foi deixado de fora

Ambas são condições necessárias, mas não suficientes, para otimalidade
Ou seja, não há garantia de solução ótima global!

Busca Local

- Não precisamos do caminho para o obter a solução, nós buscamos possíveis soluções por meio da aleatoriedade
- Vamos então obter uma solução inicial aleatória, em tempo plausível
- Temos que ter uma função que avalie a qualidade (fitness) dessa solução. Essa qualidade não está relacionada a trajetória da solução, apenas a solução em si
- A partir dessa solução inicial, fazemos uma busca na vizinhança de soluções (local), de modo a melhorar a qualidade da solução

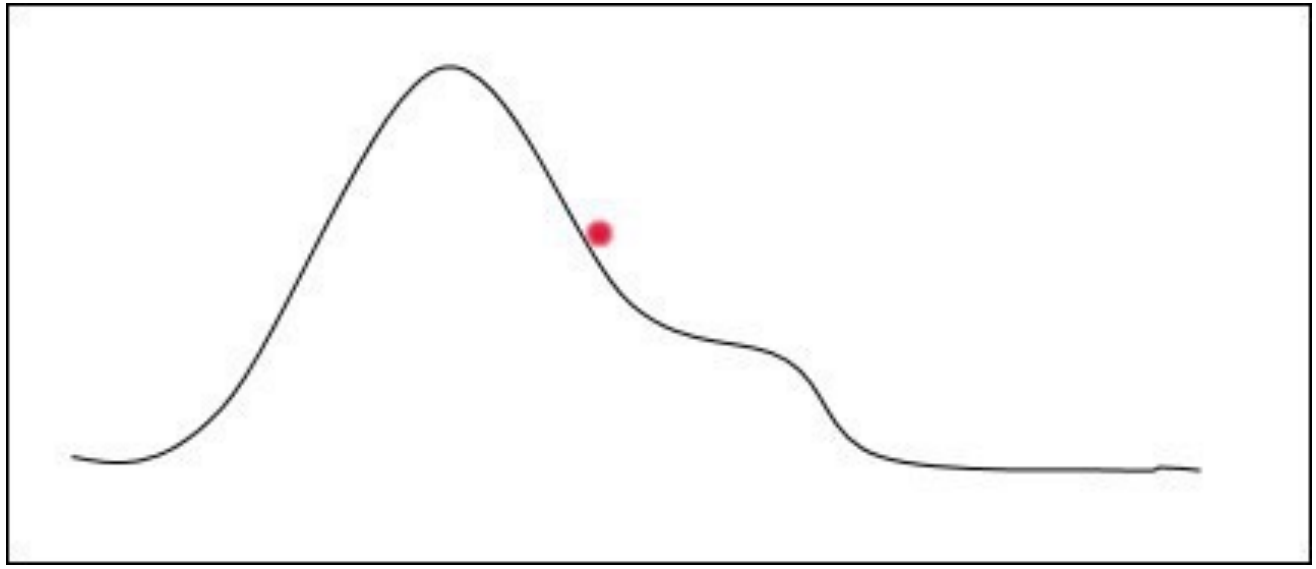
Busca Local



https://www.cs.upc.edu/~bejar/ia/transpas/teoria.mti/2-BH3-Busqueda_local-eng.pdf

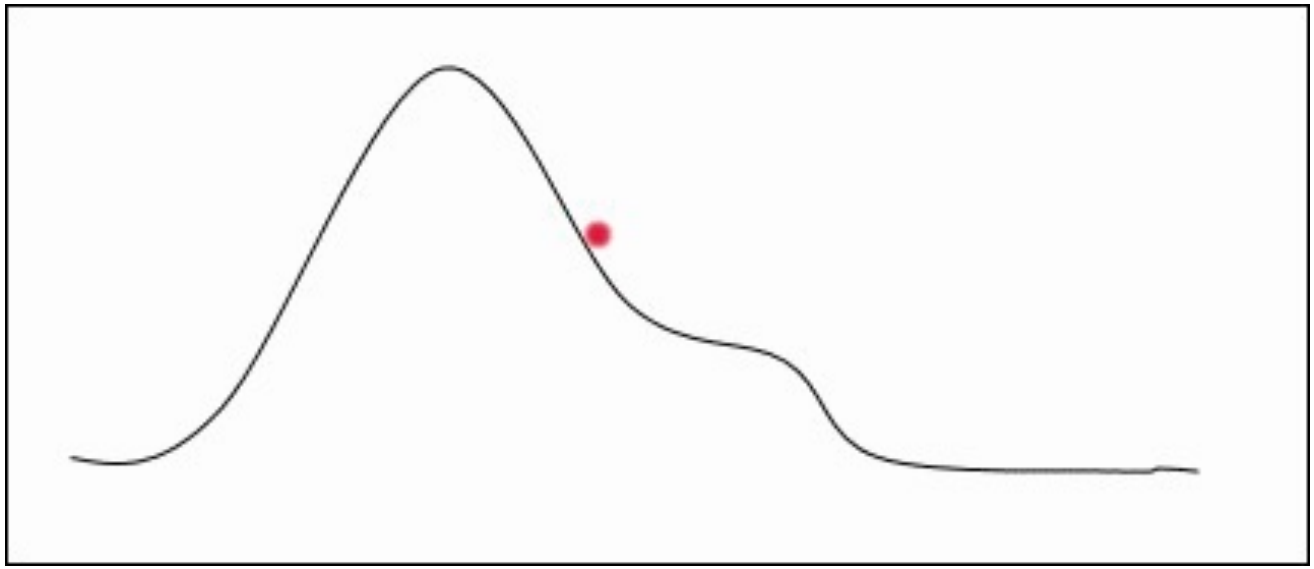
Busca Local – Hill Climbing

- Hill Climbing é um algoritmo clássico para otimização, bastante eficiente na tarefa de encontrar máximos ou mínimos locais
- Nós iniciamos em um ponto aleatório X e fazemos sua avaliação



Busca Local – Hill Climbing

- Nós movemos então do nosso ponto X para um novo ponto vizinho X’
  - Se esse ponto X’ for uma solução melhor que X, ficamos nele e repetimos o processo
  - Caso contrário, voltamos para X e podemos visitar outro vizinho ou interromper



Hill Climbing para a Mochila

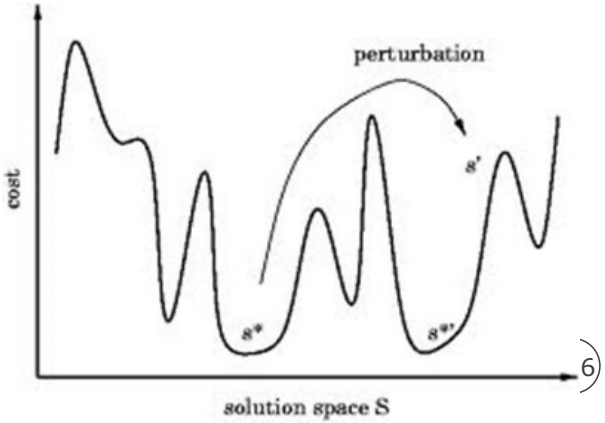
- Suponha uma mochila com capacidade C e diversos itens, com seus respectivos pesos (W) e valores (V)
- Nosso objetivo: maximizar $\sum V$, respeitando a restrição $W \leq C$
- Podemos codificar nosso problema como uma string binária. 0 significa que o item i não foi incluído, enquanto que 1 significa que i foi incluído
- Supondo 10 objetos, nossa string poderia ser: 0010010000
    - Para este exemplo, vamos gerar 10 possíveis vizinhos a partir da modificação de um bit: 1010010000, 0110010000, etc.
    - Vamos computar a qualidade desses 10 vizinhos. Se houver um melhor, ele é a nova solução e repetimos o processo, até que nenhum vizinho melhor seja encontrado

Busca local com pertubação

- Ideia baseada em dois estágios:
  - Gera uma solução inicial aleatória, e realiza busca local (hill climbing) [intensificação]
  - Faz uma perturbação na melhor solução encontrada, para evitar máximo local [diversificação]
- Desafio: controlar intensificação vs diversificação (critério de parada)

Algoritmo Busca local iterada (ILS):

```
INÍCIO
    s = inicializa (s)
    s = hill-climbing (s)
    ENQUANTO NÃO critério_parada
        r = s
        s = perturbação (s)
        s = hill-climbing (s)
        SE s < r
            s = r
FIM
```



Busca local - vantagens

- Rápida
- Resultados bons para N grande
- Oferece garantia fraca de qualidade (máximos/mínimos locais)
- Não ficou bom? Rode mais vezes

Busca local - desvantagens

- Depende da solução inicial
- Aleatorizada (o que nem sempre é um problema)
- Oferece garantia fraca (máximos/mínimos locais) de qualidade
- Estratégia de perturbação pode ajudar

Atividade prática

Resolvendo a mochila binária por meio de busca local

Discussão

- As soluções ficaram melhores que as heurísticas?
- E o tempo de execução?

Obrigado