Tratamento de problemas $\mathcal{NP}$-difíceis: Heurísticas de Busca Local

Cid C. Souza
Eduardo C. Xavier

Instituto de Computação/Unicamp

26 de abril de 2011

Heurísticas de Busca Local

- Temos um problema de otimização para resolver devendo definir:
  - O objetivo do problema.
  - Uma função de avaliação de soluções (se solução é inviável $\Rightarrow \infty$ ou $-\infty$).
  - Espaço de busca.
- Sendo $\mathcal{F}$ o conjunto de todas as possíveis soluções, e $t \in \mathcal{F}$, **a vizinhança** da solução t, $N(t)$, é um subconjunto de $\mathcal{F}$ com soluções que podem ser obtidas ao se realizar um conjunto de transformações pré-determinadas sobre t.

Heurísticas de Busca Local

Exemplo 1:

- Uma solução é representada por uma tupla que é um vetor binário de tamanho n.
- $N_1(t)$: conjunto de todas as tuplas obtidas de t "flipando" uma de suas componentes.
- Tamanho da vizinhança é n.

Heurísticas de Busca Local

Exemplo 2:

- Uma solução é um vetor representando uma permutação de $\{1, \ldots, n\}$.
- $N_2(t)$: conjunto de todas as tuplas obtidas trocando-se as posições de dois elementos da permutação.
- Tamanho da vizinhança é $\binom{n}{2}$, ou seja, $\Theta(n^2)$.

Heurísticas de Busca Local

Uma heurística de busca local funciona basicamente assim:

1. Encontrar uma solução inicial t e avaliar seu custo. Defina t como a solução *corrente*.
2. Encontre $N(t)$ e escolha a melhor solução $t' \in N(t)$.
3. Se o custo de t' é menor que o custo de t, fazer $t \leftarrow t'$ e caso contrário descarte t'.
4. Repetir os passos 2 e 3 até que nenhuma solução t' da vizinhança seja melhor que a solução corrente t.

Heurísticas de Busca Local

Considerações:

- O processo ocorre até encontrarmos um *ótimo local*.
- Isto pode ser rápido, mas também pode demorar muito tempo!
- Você pode repetir o processo escolhendo soluções iniciais diferentes.
- Deve-se avaliar o tamanho da vizinhança:
  - Se a vizinhança for muito grande muito tempo será gasto tentando achar o melhor vizinho (ex: vizinhança é todo o espaço de busca).
  - Se a vizinhança for muito pequena podemos cair rapidamente em um ótimo local.

Heurísticas de Busca Local

```
Busca-Local
    S ← Gera-Solucao-Inicial;
    min-local ← false;
    Enquanto (não atingir critério de parada) e (não min-local) faça
        N(S) ← Calcula-Vizinhos(S);
        S' ← Sol. de menor custo em N(S);
        Se custo(S') < custo(S) então
            S ← S' ;
        Senão
            min-local ← true
    fim-enquanto
    Retornar S.
```

Heurísticas de Busca Local

Vertex-Cover

- Temos um grafo $G = (V, E)$ e devemos encontrar um subconjunto de vértices S de tamanho mínimo, tal que todas arestas incidem em pelo menos um vértice de S.
- O espaço de soluções são todos os possíveis subconjuntos de V.
- O custo de S, $c(S)$ é:
  - o tamanho de S se for viável
  - $|S| + \alpha \cdot$("número de arestas não cobertas") caso contrário.
- Uma vizinhança $N(S)$ será definida pelas operações de inserção ou remoção de um vértice.
- O tamanho da vizinhança é $|V|$.

Heurísticas de Busca Local

Vertex-Cover

- A solução inicial é o conjunto com todos os vértices. Qual custo?
- O que acontece se o grafo não tiver arestas? Qual solução é devolvida?
- O que acontece se o grafo for uma estrela (um vértice central com arestas saindo deste para outros vértices)? Quais soluções podem ser retornadas?
- O que acontece se o grafo for bipartido (uma partição com k vértices e outra com l)?

Heurísticas de Busca Local

SAT na CNF.

- Um bom procedimento de busca local para o SAT foi proposto por Selman, Levesque e Mitchell (1992). É conhecido como GSAT.
- O espaço de soluções é representado por um vetor binário de tamanho n (nº de variáveis).
- A partir de uma solução inicial t, é escolhido $t' \in N(t)$ que deixa o maior número de cláusulas satisfeitas.
- O processo é repetido para várias soluções iniciais.
- Uma diferença importante é que a solução da vizinhança se torna solução corrente mesmo que ela tenha menos cláusulas satisfeitas.

Heurísticas de Busca Local

SAT na CNF.

```
Retorna-Vizinho(t)
    t* ← ∅;
    Para i = 1 até n Faça
        t' ← (t com bit t[i] invertido)
        Se (t' satisfaz mais cláusulas do que t*) então
            t* ← t'.
    Retornar t*.
```

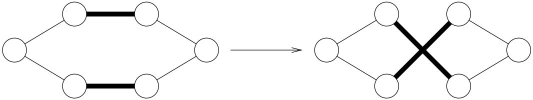
Heurísticas de Busca Local

SAT na CNF.

```
Busca-Local-GSAT
    Para i = 1 até MAX-TENTATIVAS faça
        t ← Gera-Solucao-Inicial-Aleatória
        Para j = 1 até MAX-BUSCA faça
            Se (t satisfaz todas cláusulas) então
                Retorna t
            t ← Retorna-Vizinho(t)
```

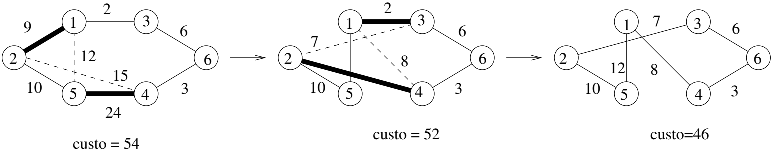
Heurísticas de Busca Local (TSP)

- Heurística da 2-troca (conhecido como 2-opt) para o TSP (Lin e Kernigham).
- Ciclo representado por uma permutação dos n vértices.
- Vizinhança: substituir pares de arestas.



- Tamanho da vizinhança é $\Theta(n^2)$.

Heurísticas de Busca Local (TSP)



- Solução são vetores com permutações de 1 até n.
- Vizinhança:
  inverte seqüência entre posições i e j $(1 \leq i, j \leq n)$ $(j \geq i+3)$.
- No exemplo:
  $(1, 3, 6, \underline{4}, 5, 2, \underline{1}) \Longrightarrow (\underline{1}, 3, 6, 4, \underline{2}, 5, 1) \Longrightarrow (1, 4, 6, 3, 2, 5, 1)$

Heurísticas de Busca Local (Partição de Grafos)

- Entrada: grafo não orientado $G = (V, E)$, com $|V| = 2n$, e custos c_{ij} para toda aresta $(i, j) \in E$.
- Saída: um subconjunto $V' \subseteq V$, com $|V'| = n$ e que minimize o valor de $\sum_{i \in V'} \sum_{j \notin V'} c_{ij}$.
- Solução representada por um vetor a de $2n$ com os valores de 1 até $2n$. Nas n primeiras posições estão os vértices de V' e nas n seguintes os vértices de $\overline{V'}$.
- Vizinhança: todas as trocas possíveis de pares de vértices $(a[i], a[j])$, onde $1 \leq i \leq n$ e $(n+1) \leq j \leq 2n$.
- Tamanho da vizinhança é $\Theta(n^2)$.

Heurísticas de Busca Local (Partição de Grafos)

Exemplo: grafo completo com 6 vértices (K_6).

$$c = \begin{bmatrix} - & 9 & 2 & 8 & 12 & 11 \\ 9 & - & 7 & 19 & 10 & 32 \\ 2 & 7 & - & 29 & 18 & 6 \\ 8 & 19 & 29 & - & 24 & 3 \\ 12 & 10 & 18 & 24 & - & 19 \\ 11 & 32 & 6 & 3 & 19 & - \end{bmatrix}$$

Solução inicial:
$a = \{1, 4, 6, 2, 3, 5\}$.

vizinhos	(1, 2)	(1, 3)	(1, 5)	(4, 2)	(4, 3)	(4, 5)	(6, 2)	(6, 3)	(6, 5)
ganho	−29	−12	−7	$\underline{-66}$	−15	−40	−22	−43	−32

- Nova solução: $a = \{1, 2, 6, 4, 3, 5\}$.

vizinhos	(1, 4)	(1, 3)	(1, 5)	(2, 4)	(2, 3)	(2, 5)	(6, 4)	(6, 3)	(6, 5)
ganho	37	34	23	66	51	26	44	59	54

- *Ótimo Local !*

Heurísticas de Busca Local

- Pode ser vantajoso que a busca local passe por soluções inviáveis !
- Nesses casos a função objetivo é composta de duas parcelas:

$$g(.) = f(.) + \alpha h(.),$$

onde f é função original, h é uma função que mede quão inviável é a solução e α é um fator de penalização.

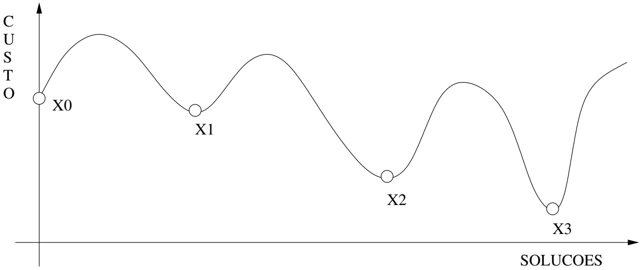
- Exemplo: No Vertex-Cover o custo da solução inviável era $|S|$ mais o número de arestas não cobertas multiplicado por α.
- Exemplo: no problema da partição de grafos, considere a vizinhança onde só um vértice muda de V' para $\overline{V'}$ ou vice-versa.
- Penalizar as soluções inviáveis usando $\alpha > 0$ grande e definindo:

$$h(V', \overline{V'}) = ||V'| - |\overline{V'}||^2.$$

- Se acabar em uma solução inviável, aplicar um algoritmo guloso que rapidamente restaura a viabilidade.

Heurísticas de Busca Local

Busca local retorna solução que é ótimo local.



- Escapando de ótimos locais: mover para melhor vizinho mesmo se o custo for pior.
- *Meta-heurísticas*: Busca Tabu, Simulated Annealing, Algoritmos genéticos,etc.