Inteligência Computacional para Otimização

Marcone Jamilson Freitas Souza
Departamento de Computação
Universidade Federal de Ouro Preto
http://www.decom.ufop.br/prof/marcone

Roteiro

Métodos de refinamento:

- Método da Descida/Subida
  - Aplicação ao Problema da Mochila
- Método de Descida/Subida com Primeira Melhora
- Método de Descida/Subida Randômica
- Método não ascendente/não descendente Randômico

Heurísticas de refinamento

- Técnicas de busca local
- Baseadas na noção de vizinhança
- Seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a otimizar (minimizar ou maximizar)
- Seja s uma solução qualquer do problema, isto é, $s \in S$

Heurísticas de refinamento

- Seja N uma função que associa a cada solução $s \in S$, sua vizinhança $N(S) \subseteq S$
- N depende do problema tratado
- Cada solução $s' \in N(s)$ é chamada vizinho de s
- Denomina-se movimento a uma modificação m que transforma uma solução s em outra, s', que esteja em sua vizinhança:

$$s' \leftarrow s \oplus m$$

Heurísticas de refinamento (Princípio de funcionamento)

- Partir de uma solução inicial qualquer
- Caminhar, a cada iteração, de vizinho para vizinho de acordo com a definição de vizinhança adotada, tentando melhorar a solução construída

Método da descida/subida (*Descent/Uphill Method*)

- Parte de uma solução inicial qualquer
- A cada passo analisa todos os possíveis vizinhos
- Move somente para o vizinho que representa uma melhora no valor atual da função de avaliação
- O método pára quando um ótimo local é encontrado

Método da descida (*Descent Method*)

procedimento $Descida(f(.), N(.), s)$;
1 $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
2 enquanto $(|V| > 0)$ faça
3 Selecione $s' \in V$, onde $s' =$ arg min$\{f(s') \mid s' \in V\}$;
4 $s \leftarrow s'$;
5 $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
6 fim-enquanto;
7 Retorne s;
fim $Descida$;

Método da Subida aplicado ao Problema da Mochila

Seja uma mochila de capacidade b = 23

Objeto (j)	1	2	3	4	5
Peso (w_j)	4	5	7	9	6
Benefício (p_j)	2	2	3	4	4

Representação de uma solução: s = $(s_1,s_2,\ldots,s_5)$, onde $s_j \in \{0,1\}$

Movimento *m* = troca no valor de um bit

Método da Subida aplicado ao Problema da Mochila

Função de avaliação:

$$f(s) = \sum_{j=1}^{n} p_j s_j - \alpha \times \max\{0, \sum_{j=1}^{n} w_j s_j - b\} \quad (31)$$

sendo α uma penalidade, por exemplo, $\alpha = \sum_{j=1}^{n} p_j = 15$.

Método da Subida aplicado ao Problema da Mochila

Passo 0 : Seja uma solução inicial qualquer, por exemplo:

$s = (01010)^t$

$f(s) = 6$

Peso corrente da mochila = 14

Passo 1 : Devemos, agora, analisar todos os vizinhos de s e calcular a função de avaliação deles usando a função de avaliação (31).

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}1010)^t$	18	8	8
$(0\mathbf{0}010)^t$	9	4	4
$(011\mathbf{1}0)^t$	21	9	9
$(0100\mathbf{0})^t$	5	2	2
$(0101\mathbf{1})^t$	20	10	10

Melhor vizinho: $s' = (01011)^t$

$f(s') = 10$

Como s' é melhor que s, pois $f(s') > f(s)$, então $s \leftarrow s'$, isto é, a nova solução corrente passa a ser:

$s = (01011)^t$

Método da Subida aplicado ao Problema da Mochila

Passo 2 : Determinemos, agora, o melhor vizinho de $s = (01011)^t$:

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}1011)^t$	24	12	-3
$(0\mathbf{0}011)^t$	15	8	8
$(011\mathbf{1}1)^t$	27	13	-47
$(0100\mathbf{1})^t$	11	6	6
$(0101\mathbf{0})^t$	14	6	6

Melhor vizinho: $s' = (00011)^t$
$f(s') = 8$
Como $f(s')$ é pior que $f(s)$, pois $f(s') < f(s)$, então PARE. A solução anterior é um ótimo local, isto é, o método da subida retorna $s^* = (01011)^t$, com $f(s^*) = 10$ como solução final.

Método da Subida aplicado ao Problema da Mochila

$$f(s) = \sum_{j=1}^{n} p_j s_j - \alpha \times f_1(s) \tag{32}$$

sendo $f_1(s) = \begin{cases} 1 & \text{se } \sum_{j=1}^{n} w_j s_j - b > 0 \\ 0 & \text{se } \sum_{j=1}^{n} w_j s_j - b \leq 0 \end{cases}$ e α uma penalidade, definida como antes, por exemplo, $\alpha = \sum_{j=1}^{n} p_j = 15$.

Observa-se que nesta formulação não se faz diferença entre o nível de inviabilidade, pois qualquer que seja o excesso de peso na mochila a penalização é a mesma. Esta modelagem pode dificultar a exploração do espaço de soluções, pois conduz a regiões planas, ditas platôs, regiões nas quais as heurísticas têm dificuldade para escapar. Em [26] os autores argumentam que uma forma comum de evitar essa topologia no espaço de busca é adicionar componentes à função de avaliação de forma a discriminar soluções que teriam o mesmo valor da função custo original. Assim, no exemplo mencionado da mochila, das duas funções de avaliação apresentadas, a mais adequada para guiar a busca é a da fórmula (31).

Método de Primeira Melhora (*First Improvement Method*)

- Variante do Método de Descida/Subida
- Evita a pesquisa exaustiva pelo melhor vizinho
- Consiste em interromper a exploração da vizinhança quando um vizinho melhor é encontrado
- Desta forma, apenas no pior caso, toda a vizinhança é explorada
- A solução final é um ótimo local com respeito à vizinhança considerada

Método de Descida/Subida Randômica (*Random Descent/Uphill Method*)

- Variante do Método de Descida/Subida
- Evita a pesquisa exaustiva pelo melhor vizinho
- Consiste em escolher um vizinho qualquer e o aceitar somente se ele for de melhora
- Se o vizinho escolhido não for de melhora, a solução corrente permanece inalterada e outro vizinho é gerado
- O procedimento é interrompido após um certo número fixo de iterações sem melhora no valor da melhor solução obtida até então
- A solução final não é necessariamente um ótimo local

Método de Descida Randômica (*Random Descent Method*)

```
procedimento DescidaRandomica(f(.), N(.), IterMax, s);
1   Iter ← 0;      {Contador de iterações sem melhora }
2   enquanto (Iter < IterMax) faça
3       Iter ← Iter + 1;
4       Selecione aleatoriamente s' ∈ N(s);
5       se (f(s') < f(s)) então
6           Iter ← 0;
7           s ← s' ;
8       fim-se;
9   fim-enquanto;
10  Retorne s;
fim DescidaRandomica;
```