Inteligência Artificial - Relatório
Temperatura Simulada

1 - Experimentação -Têmpera Simulada

O algoritmo funciona como segue:

1 - Pegue uma solução inicial.

2 - Escolha uma temperatura inicial.

3 - Escolha o próximo vizinho de modo aleatório da solução atual:

3.1 - Se o vizinho é melhor, o escolha para compor a solução atual.

3.2 - Se o vizinho é pior, probabilisticamente fazer este vizinho a solução atual, com base na temperatura atual e quanto "pior" o vizinho é.

4 - Diminuir ligeiramente ou não a temperatura.

5 - Voltar para a etapa 3.

Ao diminuir lentamente a temperatura, obtém-se menos chance de escolher soluções piores ao longo do tempo. Inicialmente, somos capazes de fazer alguns saltos muito grandes em torno do espaço de solução. No decorrer dos procedimentos estes salto serão cada vez menores, de forma que ao baixarmos a temperatura o suficiente até encontrarmos um ponto de equilíbrio, acabaremos com uma simples subida em encosta.

2 - Variações de Implementação

2.1 - Funções de Temperatura

Foram escolhidas 3(Três) funções de temperatura. Estas funções auxiliam no cálculo da probabilidade para aceitação ou rejeição de um novo estado. As funções escolhidas e o porquê da escolha são listadas abaixo.

Critério de aceitação de Boltzmann

Esta função de temperatura foi escolhida tanto por ser uma das mais utilizadas quando se trata de Temperatura Simulada, quanto pela sua utilidade para o algoritmo. Assim a mesma pode ser usada na técnica pois ela sendo aplicada sobre uma temperatura bem definida fornece a probabilidade para um estado específico.

Para esta função, quanto maior for a temperatura mais chances de um estado terá de ser aceito e conforme a temperatura vai baixando as chances do mesmo vão diminuindo. A altas temperaturas, cada estado tem (praticamente) a mesma chance de ser o estado corrente, a baixas temperaturas, somente estados com baixo custo têm alta probabilidade de se tornar o estado corrente.

```
# Critério de aceitação de Boltzmann, primeira função de temperatura.
def F1(p_atual, p_proximo, temperatura):
  if p_proximo > p_atual:
    return  1.0
  #Variação da energia
  dE = abs(p_proximo - p_atual)
  return math.exp(-dE/temperatura)
```

Logaritmo de "Boltzmann"

Função de temperatura baseada no critério de aceitação de Boltzmann com algumas modificações. Porém a ideia geral é muito parecida, ou seja, para altas temperaturas a probabilidade de aceitação desse estado será alta e baixa em caso de baixa temperatura.

```
# Baseada no critério de aceitação de Boltzmann, segunda função de temperatura
def F2(p_atual, p_proximo,temperatura):
  if p_proximo > p_atual:
    return  1.0
  dE = abs(p_proximo - p_atual)
  if dE > 0:
    return math.log10((dE/temperatura))
  else:
    return -1
```

Critério de aceitação de Kirkpatrick

Esta também é uma função de temperatura muito utilizada para a temperatura simulada. A função é geralmente escolhida de modo que a probabilidade de aceitar um movimento diminua quando a diferença aumenta(Diferença entre o próximo estado e o estado atual). Ou seja, pequenos movimentos em subida são mais prováveis do que os grandes.

```
# Critério de aceitação de Kirkpatrick, terceira função de temperatura.
def F3(p_atual, p_proximo, temperatura):
  if p_proximo < p_atual:
    return  1.0
  dE = abs(p_proximo - p_atual)
  return math.exp(-dE/temperatura)
```

Estas funções impedem que o método fique preso em um mínimo local que seja é pior que o mínimo global.

2.2 - Funções de Resfriamento

Foram propostas 2 (Duas) funções de resfriamento geométrico(Uma das funções mais utilizadas no algoritmo), as quais ajudam o algoritmo a passar por mínimos locais, visto que proporcionam um "chacoalhamento" que possibilita o mesmo a sair de um mínimo local e ir assim verificar um outro estado. Segue abaixo as funções propostas.

Função 1:

É uma das funções de resfriamento mais comuns e utilizadas, a mesma fornece um resfriamento lento, ou seja, a temperatura diminui lentamente de acordo com a seguinte fórmula: T = alpha * T, onde alpha é uma constante tal que 0 < alpha < 1 e T é a temperatura. Normalmente utiliza-se valores entre 0.88 a 0.99 para a constante alpha.

O número de iterações a cada T pode ser variável,visto que está ligada a uma taxa fixa de aceitação de movimentos, alta T implica em poucas iterações.

Função 2:

É uma função muito utilizada, porém não tanto como a primeira, proporciona um resfriamento rápido, com somente uma iteração por temperatura. Sua fórmula é a seguinte: T = T / (1 + ß*T), onde T é a temperatura e ß é uma constante que possui um valor relativamente pequeno.

Baseado no fato desse resfriamento da temperatura ajudar a sair de mínimos locais, a função 1(Um) deve ser melhor do que a função 2(Dois). Visto que ela fará mais movimento em prol de sair desses mínimos locais, enquanto que a função dois fará menos movimento, pois a temperatura irá diminuir mais rápido.

Portanto, foi proposto uma função que faz o resfriamento lentamente e outra que faz esse resfriamento rapidamente, ambas são muito utilizadas e a utilização de uma das duas depende do quão rápido você quer efetuar esse resfriamento em busca de um ponto de equilíbrio.

3 - Comparações entre as variações

Teste com uma instância de 7 (Sete) cidades.

Variação 01:
Melhor Caminho: *[(3.0, 2.0), (2.0, 0.0), (0.0, 0.0), (1.0, 1.0), (1.0, 2.0), (0.0, 2.0), (1.0, 3.0)]*
Custo: *9.06449510225*
Iterações: *10*

Variação 02:
Melhor Caminho: *[(2.0, 0.0), (0.0, 0.0), (0.0, 2.0), (1.0, 2.0), (1.0, 1.0), (1.0, 3.0), (3.0, 2.0)]*
Custo: *10.2360679775*
Iterações: *25*

Variação 03:
Melhor Caminho: *[(3.0, 2.0), (1.0, 3.0), (1.0, 2.0), (0.0, 2.0), (0.0, 0.0), (1.0, 1.0), (2.0, 0.0)]*
Custo: *9.06449510225*
Iterações: *10*

Variação 04:
Melhor Caminho: *[(0.0, 2.0), (0.0, 0.0), (1.0, 1.0), (1.0, 2.0), (1.0, 3.0), (3.0, 2.0), (2.0, 0.0)]*
Custo: *9.88634951737*
Iterações: *25*

Variação 05:
Melhor Caminho: *[(3.0, 2.0), (1.0, 3.0), (1.0, 2.0), (0.0, 2.0), (1.0, 1.0), (2.0, 0.0), (0.0, 0.0)]*
Custo: *9.06449510225*
Iterações: *10*

Variação 06:
Melhor Caminho: *[(3.0, 2.0), (1.0, 3.0), (1.0, 2.0), (2.0, 0.0), (1.0, 1.0), (0.0, 2.0), (0.0, 0.0)]*
Custo: *10.3005630797*
Iterações: *25*

Variação 07:
Melhor Caminho: *[(3.0, 2.0), (1.0, 3.0), (1.0, 2.0), (0.0, 2.0), (0.0, 0.0), (1.0, 1.0), (2.0, 0.0)]*

Custo: *9.06449510225*
Iterações: *10*
Variação 08:
Melhor Caminho: *[(0.0, 2.0), (1.0, 1.0), (1.0, 2.0), (3.0, 2.0), (2.0, 0.0), (0.0, 0.0), (1.0, 3.0)]*
Custo: *11.8125592*
Iterações: *25*

Variação 09:
Melhor Caminho: *[(3.0, 2.0), (2.0, 0.0), (0.0, 0.0), (1.0, 1.0), (1.0, 2.0), (1.0, 3.0), (0.0, 2.0)]*
Custo: *9.06449510225*
Iterações: *10*

Variação 10:
Melhor Caminho: *[(0.0, 2.0), (0.0, 0.0), (2.0, 0.0), (1.0, 1.0), (1.0, 2.0), (1.0, 3.0), (3.0, 2.0)]*
Custo: *9.65028153987*
Iterações: *25*

Variação 11:
Melhor Caminho: *[(0.0, 2.0), (1.0, 3.0), (1.0, 2.0), (1.0, 1.0), (0.0, 0.0), (2.0, 0.0), (3.0, 2.0)]*
Custo: *9.06449510225*
Iterações: *10*

Variação 12:
Melhor Caminho: *[(2.0, 0.0), (3.0, 2.0), (1.0, 2.0), (1.0, 3.0), (0.0, 2.0), (1.0, 1.0), (0.0, 0.0)]*
Custo: *9.47870866462*
Iterações: *25*

Baseado na análise dos resultados obtidos com a experimentação, foi possível perceber que na maioria das vezes as variações que possuem uma função de resfriamento lenta obtiveram um melhor resultado do que as que possuem uma função de resfriamento rápida. O resultado das variações que utilizam uma função de resfriamento lento não muda, enquanto que os das outras variações variam, porém permanecem relativamente próximo aos outros.

No entanto, essas funções demoraram um tempo maior para encontrar uma solução. Logo, podemos concluir que o esperado das funções antes da experimentação foi concretizado, ou seja, as funções de resfriamento lento, encontram melhores soluções, porém demoram muito para encontrar tal solução. Enquanto que as funções de resfriamento rápido, encontram soluções rapidamente, mas não tão boas quanto as outras.

Contudo, ficou evidente que os parâmetros iniciais determinam o quão bem o algoritmo funciona, visto que uma boa escolha da temperatura e da quantidade de iterações por temperatura influencia diretamente no algoritmo. Todavia, para uma boa determinação desse conjunto de dados iniciais é realizado diversos testes com diferentes combinações de temperatura e quantidade de iterações, com a intenção de determinar aqueles valores iniciais que geram melhores resultados para o problema.

4 - Conclusões

A principal vantagem da Temperatura Simulada é a tentativa de convergir para uma solução ótima. Como desvantagem, relaciona-se ao tempo de processamento e a calibragem dos parâmetros. Pois a escolha da taxa de redução de temperatura, temperatura inicial e final é muito importante. Escalas de temperatura (taxa de redução) que permite um resultado bom (ótimo global) não são muito adotadas em aplicações por serem lentas.

Contudo, ela não garante que a solução encontrada seja a melhor possível. Depende do estado inicial e do problema onde está sendo aplicado. Visto que os parâmetros mais adequados para uma dada aplicação podem ser estabelecidos por experimentação e isso pode demorar muito.

A Temperatura Simulada é semelhante a Subida de Encosta quando a temperatura está estabilizada. Se a mesma não aceita piores estados, então é Subida de de Encosta. No entanto, ela é ótima e completa quando o resfriamento é suave com muitas iterações para alcançar o equilíbrio da temperatura.