# RESOLVENDO O CLÁSSICO PROBLEMA DOS ROBÔS MINERADORES COM A HEURÍSTICA DO *Tabu Search*[*]

ANDERSON DE REZENDE ROCHA
JÚLIO CÉSAR ALVES

UFLA - Universidade Federal de Lavras
DCC - Departamento de Ciência da Computação
Campus Universitário, CEP 37.200-000, Lavras (MG)
{undersun,jcalves}@comp.ufla.br

8 de junho de 2003

## 1 O problema dos robôs mineradores

Um exemplo clássico do modelo da funcionalidade emergente é o dos robôs mineradores. Este exemplo foi usado pela primeira vez por Luc Steels [Steels, 1990], mas tem sido desde então usado em várias áreas como na robótica, algoritmos genéticos e na simulação do comportamento de animais.

O objetivo é fazer um conjunto de robôs encontrar e levar para uma ou mais bases amostras de minerais. A localização das amostras e o ambiente são desconhecidos, mas tipicamente as amostras encontram-se agrupadas em "depósitos". A Figura 1 mostra os elementos do problema.

Segundo [Sichman e Alvares, 1997], supõe-se que a base central emita um sinal decrescente com a distância (gradiente), que pode ser detectado pelos robôs e que indica o caminho de volta à base.

Para resolver o problema proposto, Steels propõe robôs bem simples, capazes de realizar apenas cinco comportamentos elementares:

1. evitar obstáculos;
2. se perceber um mineral e não estiver carregado (já com mineral), pegá-lo;
3. se perceber a base central e estiver carregado, descarregar;

[*] Artigo escrito para a disciplina de Inteligência Artificial lecionada por Guilherme Bastos Alvarenga

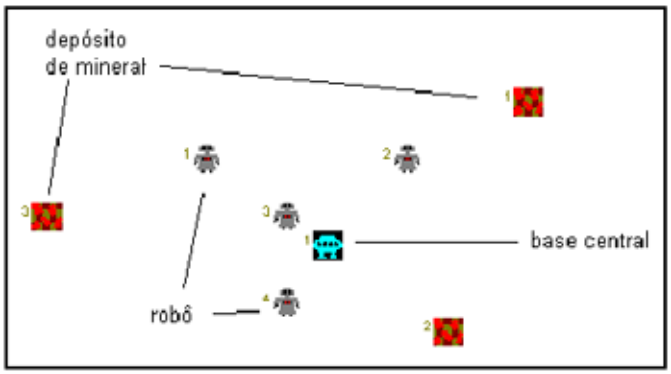


Figura 1: elementos do problema dos robôs coletores de mineral

4. se estiver carregado, seguir o sinal da base central (na direção do maior gradiente);
5. realizar movimento randômico.

As simulações realizadas mostram que estes robôs realizam a tarefa de levar todos os minerais para a base central. O problema é que quando um robô descarrega o mineral na base passa a procurar aleatoriamente mais minerais.

Outro problema verificado com esta abordagem é que o mecanismo de desvio dos obstáculos do robô é bastante rudimentar. Pensando nisso, este trabalho apresenta a solução encontrada pelos autores para resolver o mesmo problema. Utilizou-se a heurística do *Tabu Search* para melhorar o processo de locomoção e desvio dos robôs no ambiente.

A seção 2 dá uma visão geral da heurística adotada e, ’ posteriormente, a seção 3 apresenta a solução dos autores.

## 2 A heurística Tabu Search

Segundo [Glover, 1993], *Tabu Search* é uma meta-estratégia para guiar heurísticas conhecidas a superar o problema da optimimalidade local. De forma geral, é uma heurística computacional de busca conhecida por geralmente superar o problema da convergência local em problemas de otimização.

Embora ainda esteja em sua infância, esta meta-heurística tem aparecido na literatura nos últimos anos como uma abordagem de solução para uma ampla variedade de problemas.

## 2.1 O desenvolvimento histórico

A heurística tabu é relativamente recente, teve origem em meados da década de 70. A sua forma atual é devida, principalmente, aos trabalhos de Glover, Hansen e Laguna. Esta heurística não é estática e, atualmente, encontra-se em constante melhoria.

## 2.2 O funcionamento da heurística

A palavra tabu sugere algo proibido, ou pelo menos inibido. Desta forma, a heurística básica tem esse nome por aplicar restrições tabu para inibir certos movimentos. Alguns procedimentos denominados critérios de aspiração são utilizados para decidir quando movimentos classificados como tabu podem ser executados. Desta forma, a busca tabu conduz a busca para áreas ainda não analisadas do espaço de busca, tendendo a evitar a convergência da solução para um máximo (ou mínimo) local. As restrições tabu são geralmente controladas por uma lista que memoriza os últimos movimentos executados. O tempo que um movimento deve permanecer nesta lista, em geral, está relacionado com o número de iterações do algorimo e com o conjunto de movimentos possíveis a partir da solução candidata atual (solução que está sendo analisada). A implementação de uma heurística como essa envolve, de maneira geral, decisões do tipo:

- como os movimentos são realizados;
- como gerar novas soluções;
- quais serão os critérios de aspiração;
- como será feito o gerenciamento da memória de movimentos (lista tabu)

A busca tabu pode ser convenientemente caracterizada como sendo uma busca através de soluções vizinhas. Cada solução $x \in X$ tem um conjunto associado de soluções vizinhas $V(x) \subset X$ chamadas soluções vizinhas a $x$. Toda solução $x' \in V(x)$ pode ser gerada a partir de $x$ por um certo tipo de operação denominada *movimento*. Normalmente, na busca tabu, soluções vizinhas são simétricas, ou seja, $x'$ é vizinha a $x$, se e somente se, $x$ é solução vizinha a $x'$.

Os critérios de aspiração são introduzidos na busca tabu para determinar quando uma restrição tabu pode ser quebrada. Isto é, a restrição é ignorada e o movimento, mesmo classificado como proibido, é executado. Um dos critérios de aspiração mais utilizados é o de ignorar a restrição tabu sempre que a solução formada por um determinado movimento proibido for melhor que a melhor solução encontrada até o momento. A aplicação adequada deste procedimento é fundamental para se atingir altos níveis de performance.

### 2.3 Desenho do algoritmo

Suponha:
$S \rightarrow$ Solução corrente
$S^* \rightarrow$ Melhor solução conhecida
$N(S) \rightarrow$ {Todas as soluções obtidas a partir de uma simples transformação em $S$}
$\tilde{N}(S) \rightarrow$ {Subconjunto admissível de $N(S)$, ou seja, região não tabu mais a região definida pelo critério de aspiração}

Deste modo, o algoritmo pode ser definido como:

```
S  <- S0;
f* <- f(S0);
T  <- 0;
Enquanto not (criterio de parada) faça
    Selecione S em argMin[f(S')]
        S' pertence N(S);
    Se f(S) < f* então
        f* <- f(S);
        S* <- S;
    Grave a lista tabu relacionada a S em T;
Apague a lista T, se necessário;
Fim;
```

## 3 A solução dos autores

Para resolver o problema dos robôs mineradores segundo a heurística da busca tabu algumas decições foram tomadas. A seguir, tem-se como foram feitas as representações dos robôs, do ambiente, obstáculos entre outras decisões.

### 3.1 Representação

#### 3.1.1 Minas

Uma mina é representada por uma posição fixa no ambiente, um identificador único e uma quantidade de pedras preciosas.

#### 3.1.2 Depósito

Um depósito é representado por uma posição fixa no ambiente, um identificador único, uma quantidade de pedras preciosas e uma capacidade.

### 3.1.3 Robôs

Um robô é representado por uma posição no ambiente, um identificador único, uma capacidade fixa de uma unidade de pedra preciosa. O robô pode estar carregado ou vazio em um dado instante.

### 3.1.4 Ambiente

Um ambiente é representado por uma região retangular, e conjuntos de: depósitos, minas, robôs e obstáculos.

## 3.2 O funcionamento dos robôs

A mina, quando tem alguma pedra preciosa, emite um sinal que pode ser captado por um robô. Com isso, o robô é capaz de saber se está se aproximando de uma mina. Os depósitos, quando não estão cheios, também emitem um sinal permitindo o mesmo funcionamento.

Os robôs também são capazes de perceber que encostaram em um obstáculo.

### 3.2.1 Locomoção

O robô se locomove da seguinte forma: se estiver vazio ele detecta qual mina está mais próxima e então se dirige a ela, ao chegar nela, o robô é carregado. Caso contrário, faz o mesmo procedimento em relação aos depósitos. Ao chegar a um depósito ele descarrega.

Isto foi implementado da seguinte forma:

- É calculada distância euclidiana entre os pontos que representam a posição do robô e a posição de cada mina ou depósito. A mina ou depósito que estiver a uma menor distância será então o destino do robô.
- O robô se locomove um ponto, representado por um pixel, por vez.

### 3.2.2 Tratamento de Colisão

Ao se dirigir a uma mina ou depósito um robô pode encontrar um obstáculo. Isto foi implementado utilizando uma matriz dos mesmos do ambiente. Tais pontos podem ter dois estados: estar ocupado ou não. Cada entidade do ambiente ocupa uma determinada região quadrangular. Deste modo, para saber se um robô encostou em um obstáculo verifica-se se há interseção entre a região ocupada pelo robô e a região que ele está tentando ocupar.

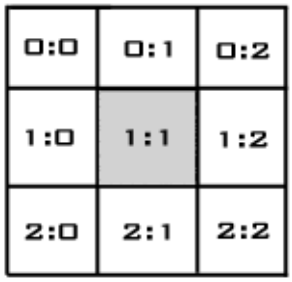


Figura 2: possibilidades de movimento para um robô a partir de uma posição

### 3.2.3 Aplicaçação do Tabu Search

Dado que um robô tenha detectado um obstáculo se faz necessária a aplicação de uma heurística de desvio. Isto foi implementado utilizando-se a heurística Tabu Search. Esta foi adaptada a esta situação como será descrito a seguir.

Um robô ao encostar em um obstáculo sorteia um determinado número de pontos aleatórios candidatos que não pertençam ao conjunto de pontos tabu (lista tabu). Os pontos sorteados pertencem a uma região quadrangular de lado igual 2 x (um determinado raio de procura) + 1, que tem como centro a posição ocupada pelo robô.

O robô então irá se dirigir ao ponto candidato que esteja a uma menor distância do seu objetivo. Todos os outros pontos candidatos passarão a fazer parte do conjunto de pontos tabu. Fazem parte deste conjunto não apenas os pontos em si, mas uma região ao redor de cada ponto determinada por um raio tabu.

Ao sortear-se um ponto candidato caso este pertença a região tabu, outro ponto é sorteado. Este processo é repetido um determinado número máximo de vezes. Caso este número seja alcançado o raio de procura é dobrado.

# 4 Funcionamento do Software

A seguir tem-se a interface do *software* desenvolvido.

1. *Botão Iniciar*. Este botão inicia a simulação. Um segundo clique pausa a simulação
2. *Botão Gerar Elementos*. Permite que robôs, minas e depósitos sejam inseridos no ambiente em posições aleatórias. O usuário escolhe a quantidade de cada um.
3. *Botão Limpar Ambiente*. Reinicia o ambiente retirando todos os robôs, minas, depósitos e obstáculos.
4. *Botão Configurações*. Permite que o usuário modifique o raio de procura, o número pontos sorteados por vez e o raio tabu.

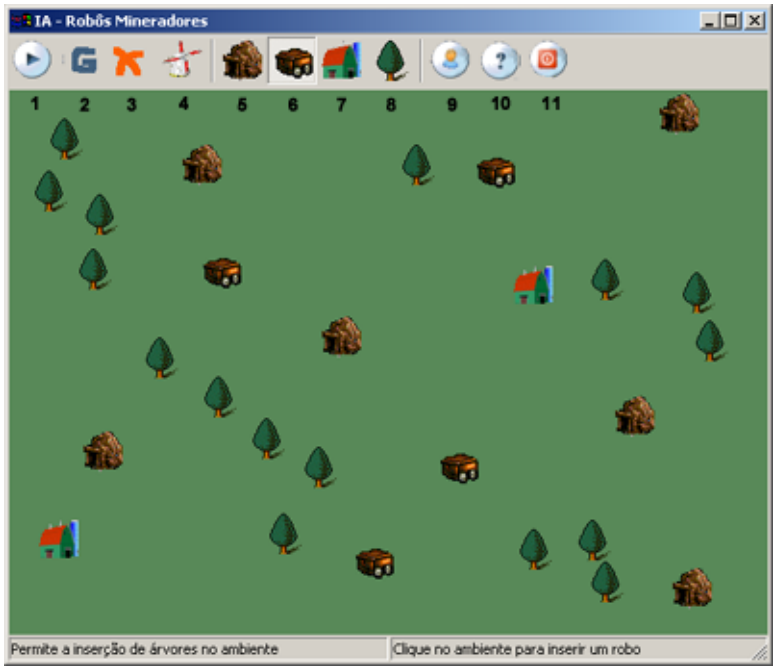


Figura 3: interface da ferramenta desenvolvida

5. *Botão Mina*. Permite que o usuário insira uma mina no ambiente com o uso do mouse.
6. *Botão Robô* . Permite que o usuário insira um robô no ambiente com o uso do mouse.
7. *Botão Depósito*. Permite que o usuário insira um depósito no ambiente com o uso do mouse.
8. *Botão Árvore*. Permite que o usuário insira uma árvore (obstáculo) no ambiente com o uso do mouse.
9. *Botão Sobre*. Exibe informações sobre o desenvolvimento do software
10. *Botão Ajuda*. Exibe uma ajuda.
11. *Botão Sair*. Sai do programa.

# 5 Considerações finais

Com o presente trabalho pudemos verificar que por ser uma solução heurística, os robôs não seguem os melhores caminhos. Mas verificou-se que o algoritmo é realmente aplicável a situações reais.

Seria interessante que os robôs possuíssem uma certa capacidade de aprendizado. Deste modo, ao descobrirem um caminho utilizando o Tabu Search passariam a memorizar aquele caminho.

# Referências


[Glover, 1993] GLOVER, Fred and Laguna, M. *Tabu Search, Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, Oxford, pps 70–150.

[Sichman e Alvares, 1997] SICHMAN, Jaime Simão e Alvares, Luis Otávio. *Introdução aos Sistemas Multiagentes*. Anais da Jornada de Atualização em Informática - JAI 97. São Paulo, 1997, editora da Sociedade Brasileira de Computação, SBC.

[Steels, 1990] STEELS, Luc. *Cooperation between Distributed Agents through Self-Organization*. In: Decentralized A.I. Demazeau, Y. e Muller, J-P. (Eds), North-Holland, Amsterdam, 1990.

[Weiss, 1999] WEISS, Gerhard. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, 1999.