24-12-2016

Tiago Miguel Matrola Simões

A 21240385 / **ANO LETIVO:** 2016-2017

***Equitable Dispersion Problem***

Introdução à Inteligência Artificial (IIA)

**ÍNDICE**

1. **Representação usada para o problema…………………………………………..**.2
   1. Descrição da função de avaliação e objetivo da otimização……………………..….…………………………………………….….………2
2. **Descrição dos algoritmos e/ou das heurísticas utilizadas……….……..**.2
   1. Explicar quais as vizinhanças métodos de seleção e operadores genéticos implementados...…………………………………………….….………2
3. **Resultados dos testes efetuados……………………………………………………..**.2
   1. Conclusão……………………..….…………………………………………….….………2
4. **Representação usada para o problema**
   1. **Descrição da função de avaliação e objetivo da otimização**

No exemplo dado no enunciado a primeira solução é S1 = {1,2,4}, este exemplo tem n = 5, portanto no modelo usado esta denominação foi convertida para binário, neste caso, a solução seria 11010. No caso da segunda solução, S2 = {2,3,4,5}, neste caso seria 01111.

O número de bits é equivalente ao número de elementos, por exemplo, no exemplo dado n = 5, todas as soluções para este exemplo terão cinco bits.

Por exemplo no caso do ficheiro “MDPI1\_20.txt” com n = 20, o número de bits seriam também vinte. Uma das soluções poderia ser 01101110001010101010 na representação do enunciado esta solução seria S = {2,3,5,6,7, 11, 13,15,17, 19}.

O objetivo da otimização é encontrar a maior distância entre os subconjuntos do conjunto. Por exemplo no exemplo do enunciado a melhor solução seria 11111, isto é, S = {1,2,3,4,5} com qualidade ou fitness igual a 4.80, mas nem sempre a melhor solução é aquela com todos os bits a um, por exemplo nos ficheiros dados existem distâncias negativas para o exemplo com n = 20 a melhor solução não tem todos os bits a um, apenas sete com qualidade = 13.88, devido à existência de distâncias negativas.

Para calcular a fitness ou a qualidade da solução dever-se-á calcular a distância entre todos os subconjuntos a partir de um deles e retirar as somas repetidas. Por exemplo no caso da S1 para n = 5, seria o somatório das distâncias de todos os pontos, calcular a distância entre 1 e 2, 1 e 4 depois entre 2 e 4 (entre 2 e 1 repetição), e as distâncias seguintes seriam repetições por isso são excluídas e posteriormente dividir esse valor, o somatório, pelo número de conjuntos que neste caso é cinco.

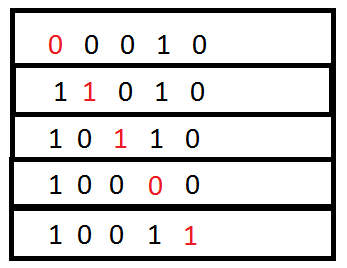
O objetivo da otimização é encontrar a solução com a melhor fitness, isto é, encontrar a solução que tem a maior distância entre os subconjuntos/pontos.

1. **Descrição dos algoritmos e/ou das heurísticas utilizadas.**
   1. **Explicar quais as vizinhanças métodos de seleção e operadores genéticos implementados**

O algoritmo escolhido para a pesquisa local foi o algoritmo “Trepa-colinas”.

Existem três variantes do “Trepa-colinas” o normal, o que aceita soluções iguais e Recristalização simulada.

O Trepa-colinas normal cria uma solução inicial, por exemplo, se “n” for igual a cinco, a solução S = 10010 depois cria uma vizinhança a partir de S alterando apenas um dos bits correspondente à sua posição e posteriormente escolhe o melhor vizinho e a partir desse melhor vizinho volta a criar uma nova vizinhança e escolhe o melhor vizinho do melhor vizinho e faz este processo até quando gerar uma nova vizinhança em que todos os vizinhos são piores que a solução atual e essa solução atual é melhor solução, abaixo encontra-se a vizinhança para a solução S:



O Trepa-colinas normal apresenta um problema que é, apenas obtém a melhor solução se não existirem máximos locais, se existirem este fica “preso” a uma solução melhor local pois a sua vizinhança é pior ou igual à solução atual. Para contornar este problema o trepa-colinas seguinte usa o mesmo conceito com a diferença deste aceitar soluções iguais para tentar sair de melhores soluções locais.

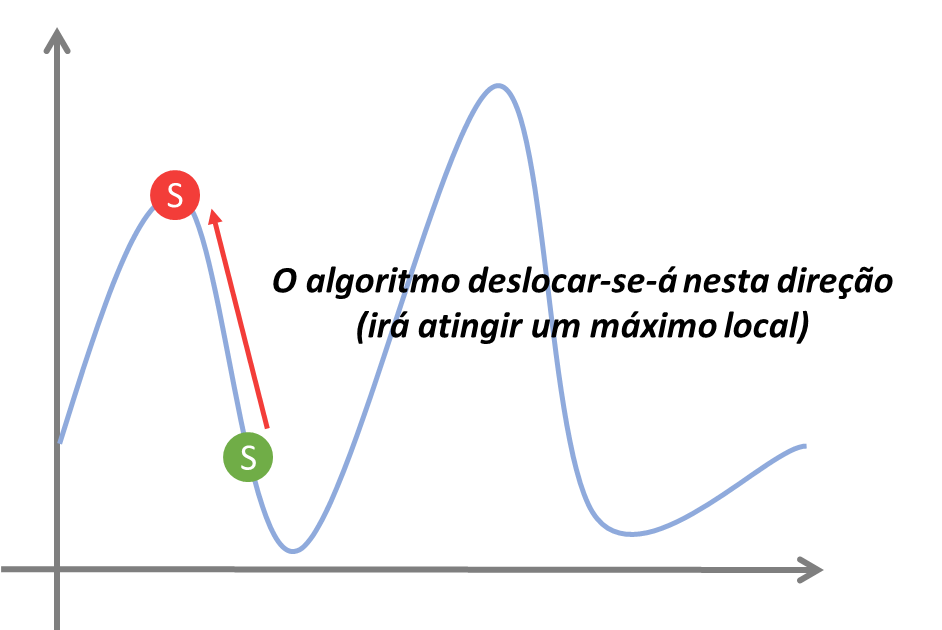


Figura 1 – Trepa-colinas normal

O Trepa-colinas que aceita soluções iguais usa o mesmo processo do normal, mas este aceita também soluções iguais com finalidade de sair de melhores soluções locais. Este algoritmo é ligeiramente melhor que o anterior, mas mesmo assim poderá atingir um máximo local, para contornar dever-se-ia aceitar soluções piores para o algoritmo tentar “vaguear” pelas soluções para não atingir o máximo local, mas esse algoritmo deverá deixar de aceitar soluções piores passado algum tempo.

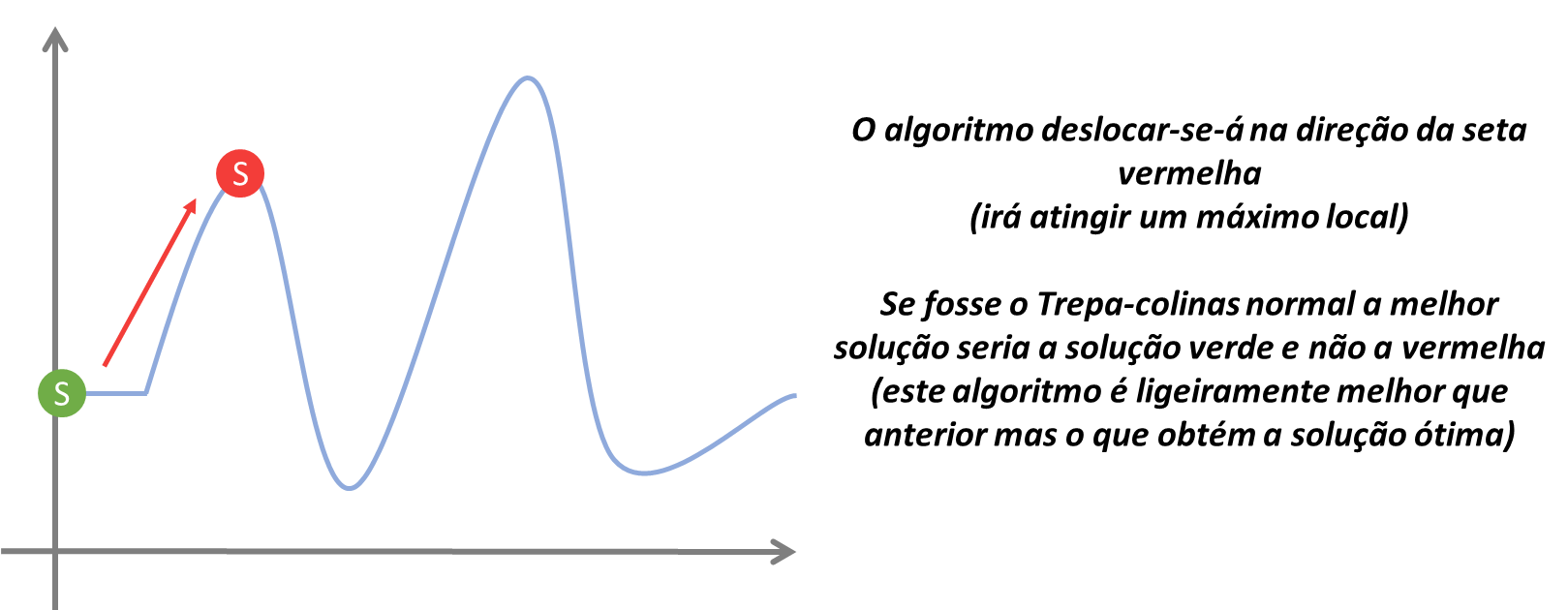


Figura 2 - Trepa-colinas que aceita soluções iguais

A Recristalização Simulada é um algoritmo começa por selecionar uma solução da vizinhança que poderá ser melhor, igual ou pior com uma determinada probabilidade de aceitação de soluções piores, mas ao longo do tempo esta probabilidade de selecionar uma solução pior começa a decrescer permitindo atingir o máximo global.

Se o “arrefecimento”, neste caso a diminuição do valor da probabilidade, for suficientemente lento, este algoritmo consegue ultrapassar os máximos locais e consegue sempre atingir o máximo/ótimo global.

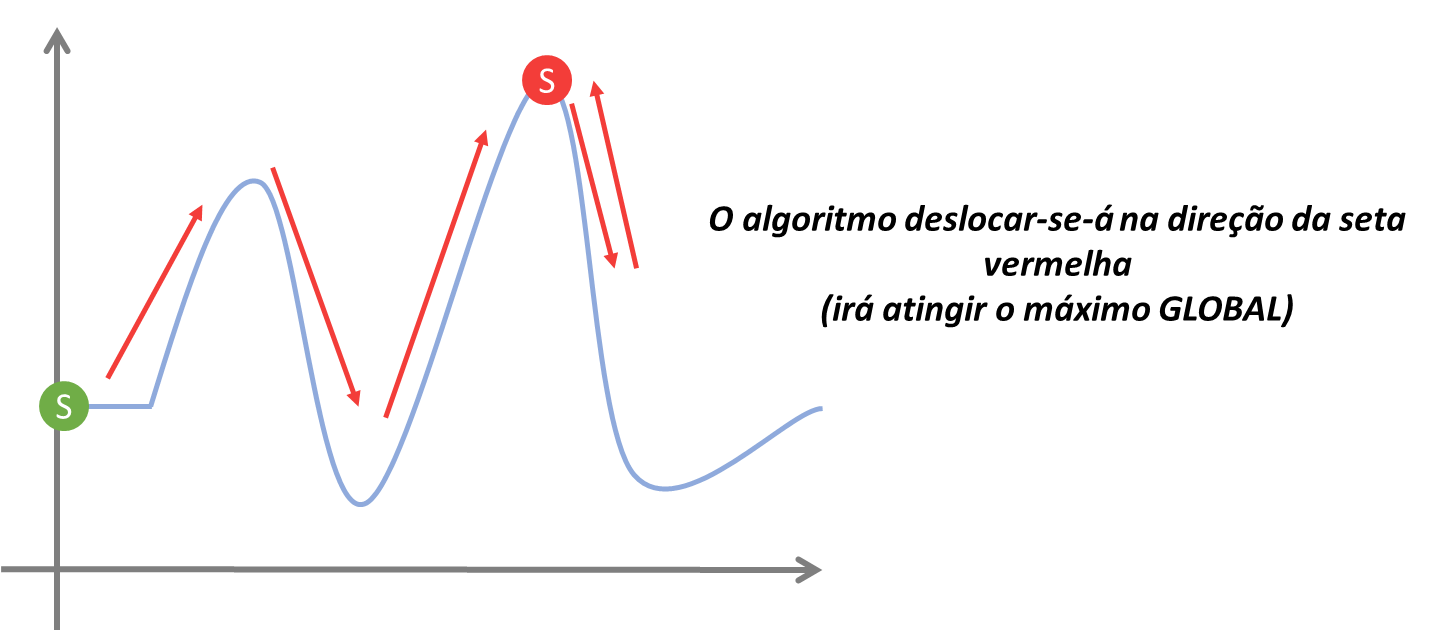


Figura 3 - Recristalização Simulada

Este gráfico mostra que o algoritmo começa por “vaguear” por a algum tempo, mas passado algum tempo deixa de “vaguear” e começa-se a comportar como um trepa-colinas normal aceitando apenas soluções melhores atingindo assim o máximo global.

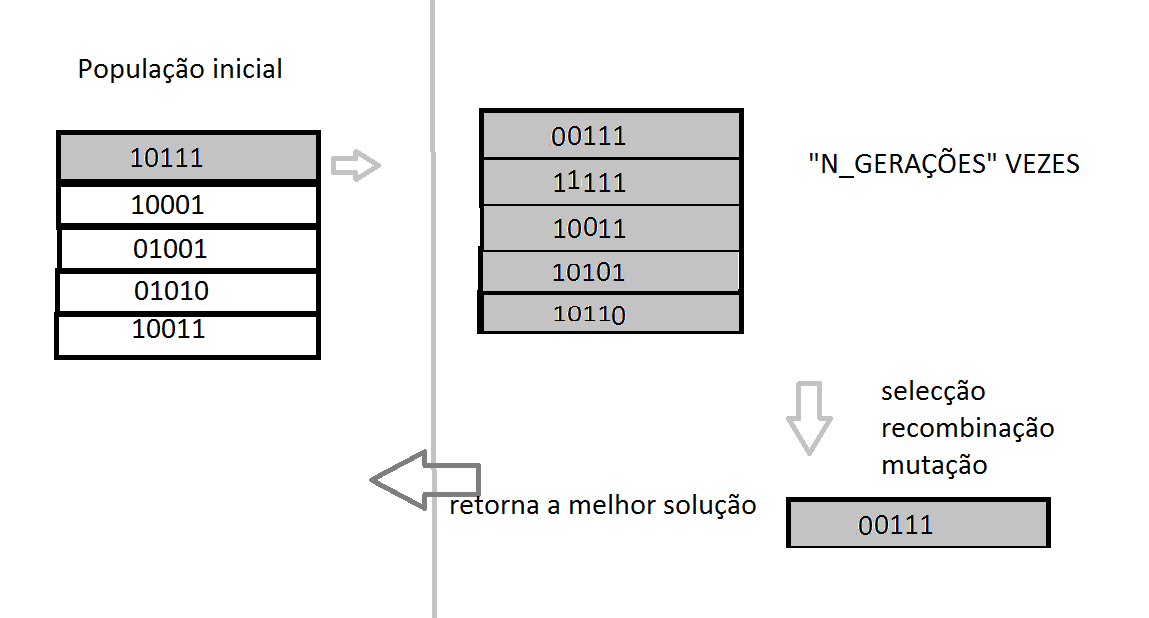
No trabalho prático foi usada o inverso da função exponencial, pois esta função começa com valores de probabilidade de aceitar soluções piores muito elevado e no fim essa probabilidade fica muito próxima de zero e é este conceito que é pretendido na recristalização simulada.

O algoritmo evolutivo cria uma população com soluções escolhidas aleatoriamente depois seleciona as melhores soluções, as soluções com maior fitness, a seguir faz a recombinação entre elas, sequentemente faz a mutação, alterando um ou mais dos bits das soluções e finalmente substitui a população anterior com a nova população. O algoritmo faz este processo até um número de gerações, que é dado pelo utilizador, por exemplo, dez vezes e retorna a melhor solução.

Este trabalho prático usa dois tipos de operadores genéticos a mutação de apenas um dos bits ou a mutação de nenhum ou mais de um dos bits.

A recombinação é feita da seguinte forma: os primeiros ¼ e ¾ dos bits da primeira solução irão para a segunda, e os primeiros ¼ e ¾ dos bits da segunda irão para a primeira.

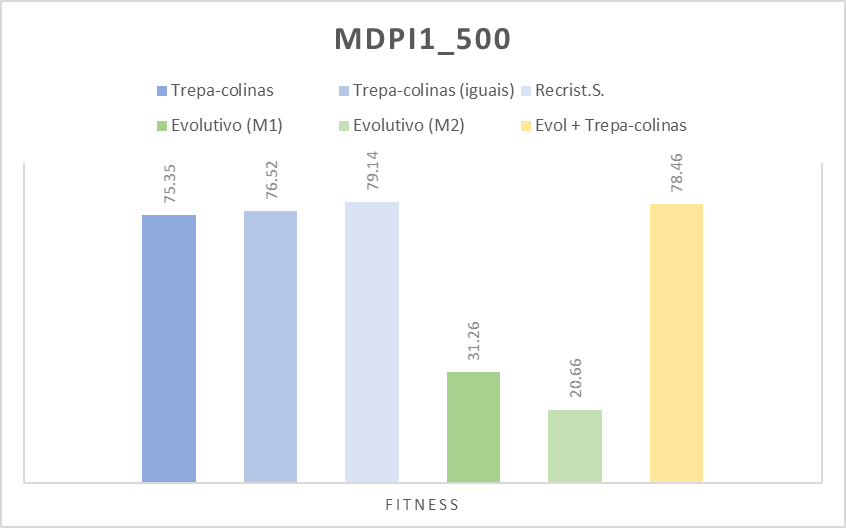
O algoritmo evolutivo com o trepa-colinas usa os dois algoritmos, isto é, cria uma população inicial aleatoriamente, depois altera um dos bits e posteriormente faz a seleção, recombinação e a mutação durante o número de gerações dadas pelo o utilizador e retorna a solução com a melhor qualidade.



1. **Resultados dos testes efetuados**
   1. **Conclusão**

A partir dos gráficos observa-se que o trepa-colina é mais eficaz do que o algoritmo evolutivo em conjuntos com o valor “n” elevado. No entanto quando os dois algoritmos são combinados produzem melhores fitnesses. A recristalização simulada neste estudo, por vezes, chega a obter melhores soluções do que o trepa-colina com o algoritmo evolutivo.

A partir dos resultados foi possível observar que a mutação de apenas um dos bits obtém melhores fitnesses que a mutação que troca um ou mais dos bits, pois um elevado nível de troca de bits faz com que a melhor solução se “perca”. Também se observou que o trepa-colinas que aceita soluções iguais é ligeiramente melhor que o normal, porque ao aceitar soluções iguais por vezes o algoritmo pode sair de máximos locais, mas poucas vezes é melhor.

A recristalização simulada, no entanto, obtém melhores resultados que o trepa-colinas normal e o que aceita soluções iguais, pois consegue ultrapassar os máximos locais por aceitar vizinhos piores nas primeiras iterações do algoritmo e muitas vezes este algoritmo chega a atingir o ótimo global. Resumindo, os dois melhores algoritmos deste estudo foram a recristalização simulada e o algoritmo evolutivo com o trepa-colinas estando praticamente empatados nos resultados das fitnesses, a recristalização simulada apresenta ligeiramente melhores fitnesses do que o anterior em conjuntos com “n” elevado.

