E. J. Solteiro Pires*
J. A. Tenreiro Machado**

Geração de Trajectórias para Manipuladores Redundantes

Utilizando Algoritmos Genéticos

1. Introdução

O aparecimento dos algoritmos genéticos (AGs) deu-se no início da década de 50 quando vários investigadores simularam sistemas biológicos. Todavia, os AGs tal como são conhecidos nos dias de hoje devem-se ao trabalho realizado por John Holland (Universidade de Michigan) no fim da década de 60. Desde então o interesse pelos AGs não tem parado de crescer e estes algoritmos são actualmente aplicados num vasto número de áreas tais como em controlo, na identificação e estimação de parâmetros, em robótica, em reconhecimento de voz, em processamento de imagem, em reconhecimento de padrões e no planeamento e escalonamentos de tarefas.

A selecção natural, ou seja, a sobrevivência dos mais aptos, governa a adaptação evolutiva do mundo biológico, desde um "simples" vírus até ao mamífero mais complexo. A selecção natural opera em todo o organismo de modo a criar descendentes com as melhores características possíveis.

Os descendentes resultam, normalmente, do crescimento de uma célula que contém um exemplar de material genético dessa espécie. É através da criação desse material específico que os pais influenciam a estrutura e as funções hereditárias do descendente. Espécies diferentes criam material genético distinto para os respectivos descendentes, mas, em todas as espécies, os pais providenciam esse material. Em alguns casos, existem dois pais, sendo o seu material genético misturado para criar o descendente. Nos casos em que existe só um pai, o material genético hereditário sofre mutações (modificações aleatórias do material genético) garantindo-se, assim, que o descendente é distinto do seu progenitor.

A evolução é o principio de unificação principal da biologia moderna. A evolução clássica de Darwin juntamente com a selecção de Weismann e a genética de Mendel é a base para a teoria da evolução actual. Por outras palavras, a evolução é o resultado de processos estocásticos interactivos (reprodução, mutação, cruzamento e selecção) sobre populações ao longo de várias gerações. Todavia, a teoria da evolução estende-se para além dos sistemas biológicos, quando utilizada por máquinas computadorizadas com o fim de resolver problemas de optimização. A computação evolutiva é o termo que abrange uma serie de algoritmos entre os quais os algoritmos genéticos consiste num dos ramos principais [1-7].

Nesta perspectiva este artigo encontra-se organizado da seguinte forma. Na secção dois são introduzidos os conceitos que fundamentam o desenvolvimento dos AGs. De seguida, na secção três apresenta-se a formulação do problema investigado, nomeadamente o planeamento de trajectórias para manipuladores robóticos. Nas secções quatro e cinco analisa-se o desempenho do AG proposto no planeamento de movimentos em ambientes sem e com obstáculos, respectivamente. Por último, na secção seis traçam-se as principais conclusões que decorrem do trabalho desenvolvido.

2. Aspectos fundamentais dos AGs

Os AGs canónicos utilizam uma representação binária nas *strings* (que têm comprimento fixo) podendo cada *caracter* da *string* ter o valor "0" ou "1". Os AGs utilizam frequentemente funções de codificação e de descodificação de modo a facilitar a projecção entre as soluções do problema e as *strings* binárias. O sucesso dos AGs é muitas vezes dependente da função de codificação. No caso de problemas de optimização com parâmetros contínuos, os algoritmos genéticos representam normalmente o conjunto de parâmetros reais por uma *string*, sendo esta dividida em vários segmentos (um por cada parâmetro (gene)) do mesmo comprimento. Cada segmento contém um valor inteiro que é projectado linearmente no intervalo que o parâmetro real correspondente pode tomar.

Um algoritmo genético tem a seguinte estrutura:

```
Inicio
t = 0
inicializar aleatoriamente P(t)
avaliação P(t)
repetir
    seleccionar P(t + 1) a partir
     de P(t)
    cruzamento P(t + 1)
    mutação P(t + 1)
    avaliação P(t + 1)
    t = t + 1
até condição de conclusão verificada
fim do algoritmo
```

Algoritmo 1 Algoritmo genético simples.

O procedimento *inicializar aleatoriamente* cria *strings* a partir de um processo heurístico. A função de avaliação é responsável pela avaliação das *strings* através de uma função de aptidão, distingido, assim, as *strings* mais capazes. A função de selecção escolhe as *strings*, de acordo com o valor de aptidão, que vão dar origem a população seguinte. A população seguinte é criada através dos operadores de cruzamento e de mutação que permitem respectivamente a troca e reintrodução de material genético. Por seu lado, o operador de cruzamento é baseado na escolha aleatória do local de cruzamento (ponto de cruzamento). Desta forma, um descendente é criado a partir da parte esquerda do ponto de cruzamento de uma *string* com

a parte direita da outra *string*. As partes restantes serão utilizadas para formar o segundo descendente (figura 1). O operador de mutação consiste em modificar um determinado gene, aleatoriamente, para o seu valor complementar (figura 2).

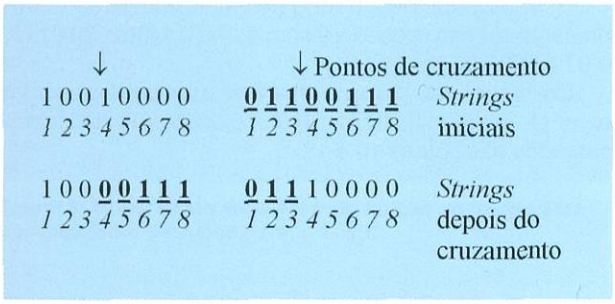

Figura 1: Cruzamento entre duas *strings*

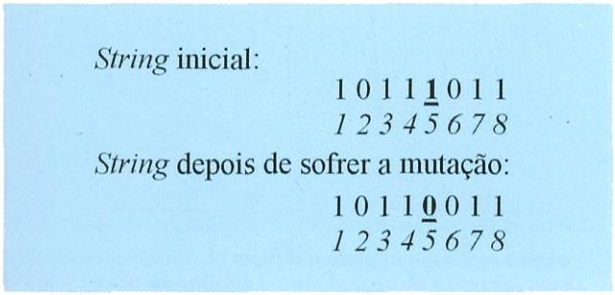

Figura 2: Mutação do *bit* número 5 de uma *string*.

O algoritmo termina quando converge ou quando é alcançado o número máximo de gerações.

3. Formulação do problema

Num sistema robótico de manipulação as trajectórias são compostas por uma sequência de pontos nos quais a "mão" deve passar. Como resultado da movimentação do braço, ao longo dos pontos discretos, obtém-se uma trajectória continua. Assim, a optimização da trajectória de um robô significa a identificação da combinação óptima e do número de posições intermédias.

No presente estudo pretende-se mover um manipulador robótico do ponto (1, 1) até ao ponto (0, 2). O robô é constituído por um conjunto de elos com um comprimento total de 2 [m]. Assim, o comprimento de cada elo do robô de 2, 3 e 4 elos é, respectivamente, de 1, 2/3 e 1/2 [m]. Considera-se que cada elo do robô pode rodar livremente em torno do seu eixo, pelo que o problema é resolvido através da cinemática directa.

A configuração angular inicial do robô de 2 elos é de (0º; 90º) para os eixos 1 e 2. A posição inicial do robô de 3 elos foi escolhida de modo a obter a configuração "próxima" da configuração do robô com dois elos. Assim, a posição inicial do robô é (-10,89º; 34,11º; 34,11º), respectivamente para os eixos 1, 2 e 3. De modo análogo, a posição inicial do robô de 4 elos é de (0º; 0º; 90º; 0º), respectivamente para os eixos 1, 2, 3 e 4.

De seguida são usados AGs onde a representação da solução é constituída por um cromossoma com $n = 15$ genes. Cada gene é composto pelos deslocamentos angulares das juntas do robô. Assim, para um robô com três elos (figura 3) o gene é composto por três valores (q_1, q_2, q_3) angulares (figura 4).

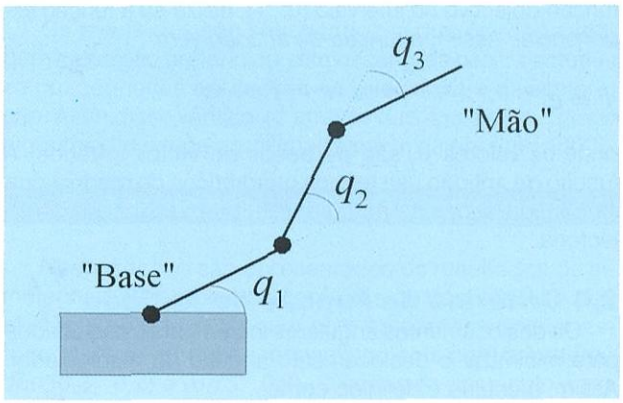

Figura 3: Robô com três elos.

$$[(q_{11},\ldots,q_{1k})\ldots(q_{i1},\ldots,q_{ik})\ldots(q_{n1},\ldots,q_{nk})]$$
$$[(\text{gene } 1)\ldots(\text{gene } i)\ldots(\text{gene } n)]$$

O parâmetro $q_{i,j}$ é a configuração i ($1 \le i \le n$) do ângulo j ($1 \le j \le k$) do robô

Figura 4: Representação do cromossoma para o robô de k elos.

Os genes são codificados através de valores reais e cada valor real pode ter um valor compreendido entre –360º e 360º. Este intervalo é utilizado para eliminar eventuais descontinuidades, que ocorrem quando um eixo passa de -180º para +180º. A configuração inicial não é codificada no cromossoma devido a esta se manter constante ao longo da pesquisa.

Por simplicidade, e sem perda de generalidade, o tempo entre duas configurações possíveis é considerado $T = 1$ seg, porque é sempre possível fazer um reescalonamento dos tempos.

No AG adoptou-se uma população de 100 cromossomas e 5000 gerações em cada simulação. Por outro lado, desenvolveram-se operadores de cruzamento e de mutação bem como uma função de aptidão e respectivos critérios de optimização, conforme descrito nas sub-secções seguintes.

2.1 Operador de Cruzamento

Depois de escolher as *strings* que vão acasalar é executado o operador de cruzamento simples. O ponto de cruzamento só pode ocorrer entre genes (*i.e.* o operador de cruzamento não pode dividir os genes).

2.2 Operador de Mutação

O valor de um gene é reposto aleatoriamente com uma certa probabilidade. O operador de mutação segue a seguinte equação:

$$q_{ij}(t+1) = q_{ij}(t) + 1.8\,x$$
$$x \sim U[-1;\ 1]$$

onde q_{ij} é o valor i do gene j na geração t e x é um número aleatório uniforme entre [-1, 1].

2.3 Função de Aptidão

Pretende-se obter uma trajectória que atinja o ponto desejado com um custo mínimo, ou seja, pretende-se que a actuação nas juntas do robô seja mínima e que não ocorram colisões com os obstáculos. Para transformar o problema de minimização em maximização e normalizar a

função objectivo no intervalo [0; 1], utiliza-se a função exponencial. Assim, a função de aptidão vem:

$$f = e^{-\alpha_1.\varepsilon-\alpha_2.\Delta q-\alpha_3.\ddot{q}-\alpha_4.\Delta p-\alpha_5.\ddot{p}-\alpha_6.nap}$$

onde os valores α_i são os pesos de vários critérios. A função de aptidão usa termos quadráticos de modo a que as configurações do robô se distribuam ao longo da trajectória.

2.4 Critérios da Função de Aptidão

Os deslocamentos angulares incrementais são usados para minimizar o deslocamento angular do manipulador. Assim, o critério é definido como:

$$\Delta q = \sum_{j=1}^{n}\sum_{i=1}^{k}(1+|\Delta q_{ij}|)^2$$

onde Δq_{ij} é o incremento angular intermédio j da junta i, na simulação do manipulador de k elos.

A aceleração angular (velocidade angular incremental, $T = 1$ segundo) é usada para minimizar a ondulação residual (*ripple*) nas juntas do robô. O critério da aceleração angular é calculado com a seguinte expressão:

$$\ddot{q} = \sum_{j=1}^{n}\sum_{i=1}^{k}(1+|\ddot{q}_{ij}|)^2$$

A distância incremental cartesiana é o critério usado para minimizar a trajectória percorrida pelo ponto terminal do robô, de acordo com a expressão:

$$\Delta p = \sum_{w=2}^{n}[1+d(p_w, p_{w-1})]^2$$

onde p_w é a posição Cartesiana do ponto terminal intermédio w e a função $d(\cdot,\cdot)$ fornece a distância entre os dois argumentos.

A aceleração cartesiana na função de aptidão é responsável pela redução da ondulação residual da resolução do ponto terminal do robô, vindo a função:

$$\ddot{p} = \sum_{w=3}^{n}[1+|d(p_w, p_{w-1}) - d(p_{w-1}, p_{w-2})|]^2$$

A distância final (ε) dá um indicação do erro entre o ponto após a simulação e o ponto que se pretende atingir. O peso α_1 deve ser superior ao peso α_2 já que, no caso contrario uma solução que se desloque no sentido do ponto desejado tem um desempenho inferior à solução onde o robô permanece no ponto de partida ($\Delta q_{ik} = 0$ para todo o i e k).

O valor dos pontos não admissíveis (nPNA) fornece uma medida do conflito existente entre o robô e os obstáculos. Este valor é calculado do seguinte modo: cada elo do robô é dividido em p partes iguais (p = (4, 3, 2) respectivamente para um robô com 2, 3 e 4 elos). De seguida, são atribuídos pesos a cada um dos pontos da forma: peso 1 para o ponto correspondente ao ponto terminal, peso 2 para o ponto adjacente ao ponto terminal, e assim sucessivamente até atingir o ponto da base. Por fim, o valor nPNA é calculado a partir da soma desses valores. Quando, não são considerados obstáculos no problema o valor de α_3 é zero como seria de esperar.

As gerações sucessivas de novas *strings* são reproduzidas com base na sua função de aptidão. Neste caso, o operador de reprodução usa o método de selecção baseado na roleta, geracional, elitista [1], para seleccionar entre as *strings* da população velha a nova geração.

4. Ambientes sem Obstáculos

Nesta secção efectuam-se experiências onde o manipulador é simulado num ambiente sem obstáculos. As probabilidades de cruzamento e de mutação usadas são, respectivamente, de p_c=0,8 e p_m=0,01.

Os resultados para o robô de 2 elos encontram-se representados nas figura 5 a 9 com α = (1; 0,01/2; 0,0015/2; 0,01;0,0005;0).

Os resultados para um manipulador de 3 elos com α = (1; 0,05/3; 0,0015/3; 0,05; 0,0005; 0) estão representados nas figuras 10 e 11.

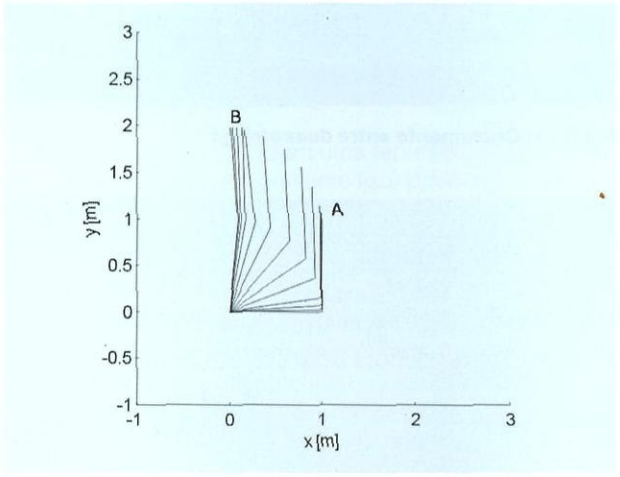

Figura 5: Configurações sucessivas para o robô de 2 elos para um ambiente sem obstáculos.
A – ponto inicial, B – ponto final.

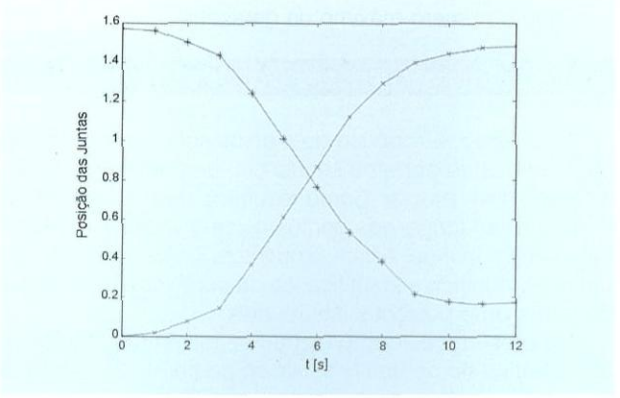

Figura 6: Posições angulares em função do tempo para um robô de 2 elos (-x-$\dot{q}_1$; -*- $\dot{q}_2$).

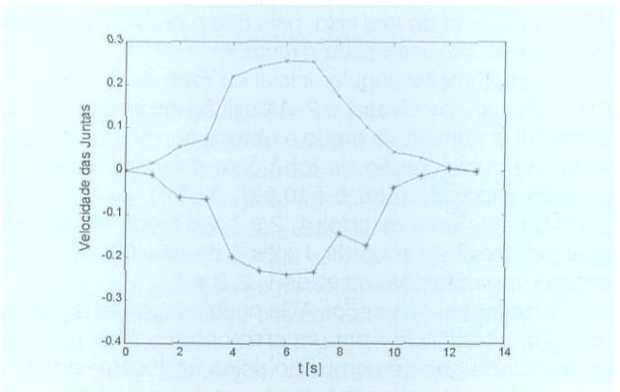

Figura 7: Velocidades angulares em função do tempo para o robô de 2 elos (-x-$\dot{q}_1$; -*- $\dot{q}_2$).

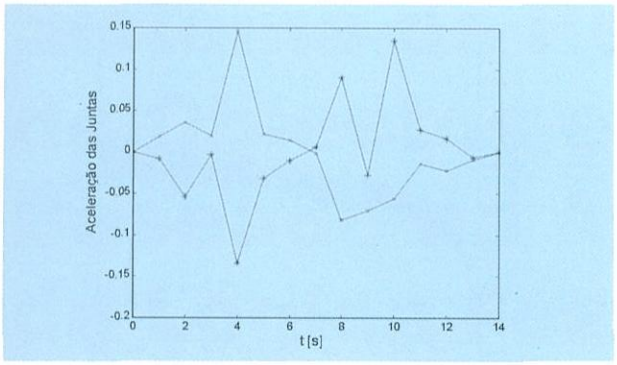

Figure 8: Acelerações angulares em função do tempo para o manipulador de 2 elos (-x- $\ddot{q}_1$; -*- $\ddot{q}_2$).

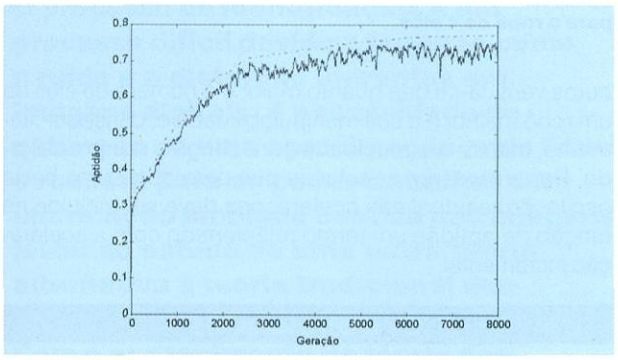

Figura 9: A evolução do melhor indivíduo (--) e a evolução do valor médio de aptidão (–) em função da geração, para o robô de 2 elos.

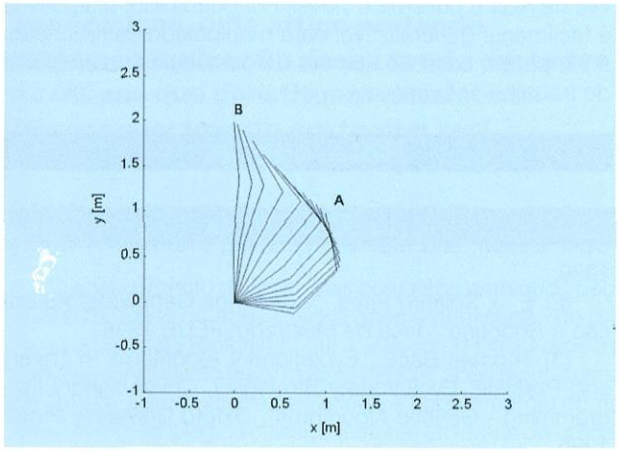

Figure 10: Configurações sucessivas para a trajectória do robô de 3 elos num ambiente sem obstáculos. A – ponto inicial, B – ponto final.

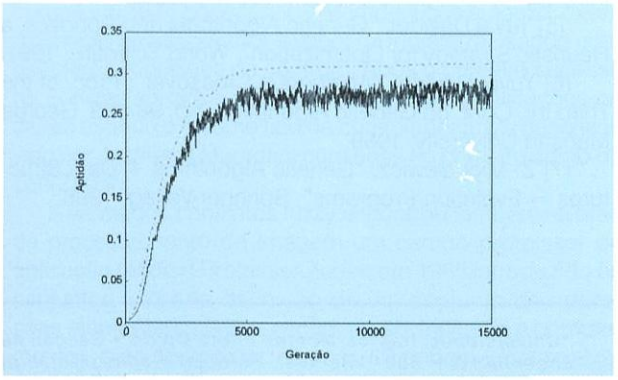

Figura 11: Evolução do melhor indivíduo (--) e evolução do valor médio de aptidão (–) em função da aptidão, para o robô de 3 elos.

Os resultados são satisfatórios já que os robôs atingem a posição pretendida sem oscilações nas trajectórias e com pequenas oscilações na velocidade e na aceleração. Além disso verifica-se através das figuras 5 e 7 que o algoritmo converge aproximadamente na geração 4000.

5. Ambientes com Obstáculos

Nesta secção são apresentados os resultados de simulações em ambientes com dois obstáculos. Um dos obstáculo é um rectângulo com os cantos superior esquerdo e inferior direito respectivamente com coordenadas (0,3; 0,2) e (0,75; 0,8). O segundo obstáculo é um circulo, com centro (-0,7; -0,7) e raio 0,8. As probabilidades de cruzamento e de mutação são p_c = 0,8 e p_m = 0,01.

Verifica-se que o robô de dois elos não atinge a meta porque não consegue passar fisicamente entre os obstáculos.

Os resultados para o manipulador de 3 elos são apresentados nas figuras 12 e 13 para α = (1; 0,01/3; 0,0025/3; 0,05; 0,0005; 100).

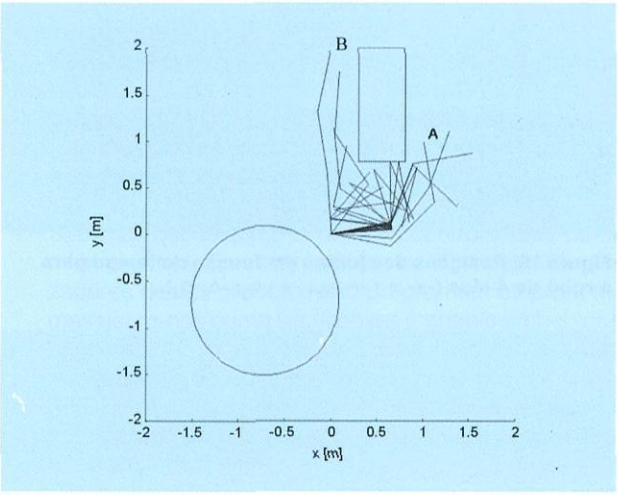

Figura 12: Configurações sucessivas para a trajectória do robô de 3 elos num ambiente com obstáculos. A - ponto inicial, B - ponto final.

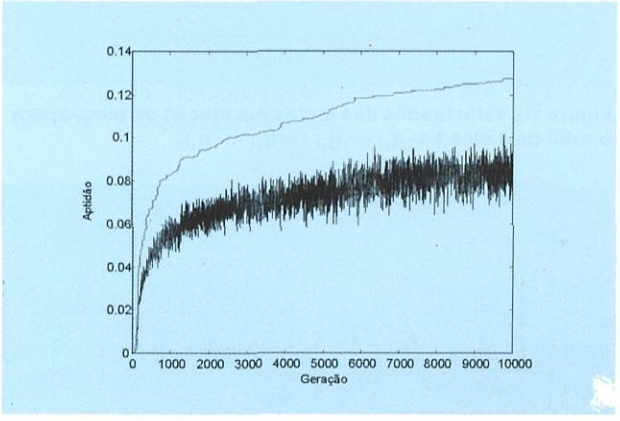

Figure 13: Evolução do melhor indivíduo (--) e o valor de aptidão médio (–) em função da geração, para o robô de 3 elos.

Os resultados para o manipulador de 4 elos encontram-se nas figuras 14 a 18 para α = (1; 0,01/4; 0,0025/4; 0,01; 0,0005; 100).

Como seria de esperar, num ambiente com obstá-

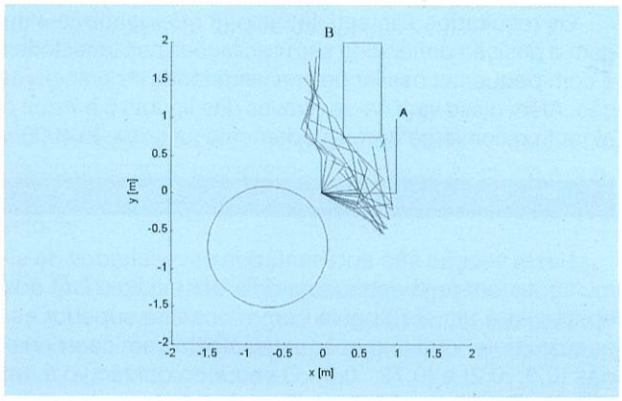

Figure 14: Configurações sucessivas para a trajectória de um robô de 4 elos e um ambiente com dois obstáculos. A - ponto inicial, B - ponto final.

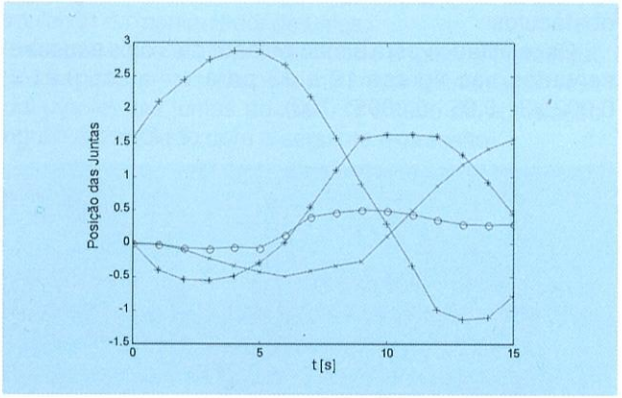

Figura 15: Posições das juntas em função do tempo para o robô de 4 elos (-x- q_1; -*- q_2; -+- q_3; -o- q_4).

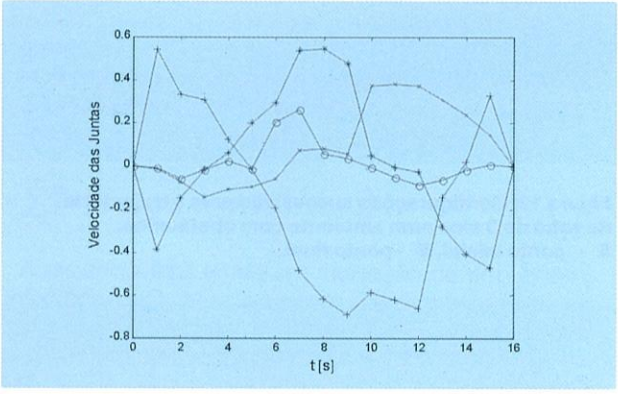

Figura 16: Velocidades das juntas em função do tempo para o robô de 4 elos (-x- $\dot{q}_1$; -*- $\dot{q}_2$; -+- $\dot{q}_3$; -o- $\dot{q}_4$).

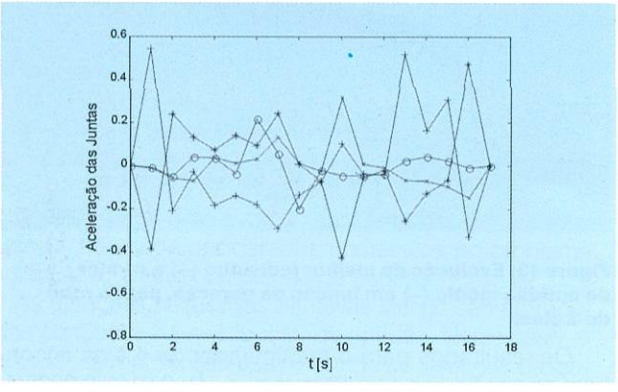

Figura 17: Aceleração das juntas em função do tempo para o robô de 4 elos (-x- $\ddot{q}_1$; -*- $\ddot{q}_2$; -+- $\ddot{q}_3$; -o- $\ddot{q}_4$).

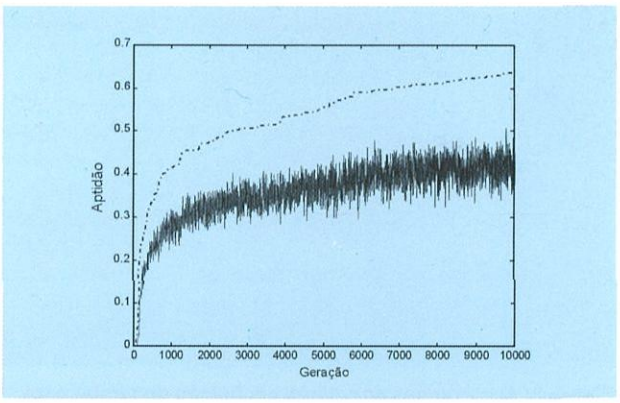

Figura 18: Evolução do melhor indivíduo (--) e a evolução do valor médio de aptidão (–) *versus* o número de gerações, para o robô de 4 elos.

culos verifica-se que quanto maior é o número de elos de um robô melhor é a sua manipulabilidade e, consequentemente, melhor a capacidade para atingir o ponto desejado. Por outro lado, conclui-se que para obter uma baixa oscilação residual nas acelerações deve ser incluído na função de aptidão um termo relacionado com a aceleração incremental.

6. Conclusões

Foi apresentado um método baseado em AGs para planear trajectórias de manipuladores robóticos. Os resultados demostram que o planeamento de trajectórias através de AGs é possível e viável. Por outro lado, o algoritmo é facilmente generalizável para manipuladores redundantes e para o caso do número de obstáculos no ambiente de trabalho ser variável.

Referências

[1] David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison – Wesley, 1989.

[2] E. J. Solteiro Pires, "Algoritmos Genéticos: Aplicação à Robótica", Tese de Mestrado, FEUP, 1998.

[3] Thomas Bäck, "Evolutionary Algorithms in Theory and Practice, Evolutionary Strategies - Evolutionary Programming - Genetic Algorithms", Oxford University Press, 1996.

[4] Thomas Bäck, Ulrich Hammel, Hans-Paul Schwefel, "Evolutionary Computation: Comments on the History and Current State", IEEE Trans. on Evolutionary Computation, Vol. 1, n. 1, pp. 3-17, April 1997.

[5] Yaval Davidor, "Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization", World Scientific, 1991.

[6] Yuval Davidor, "Analogous Crossover", Proc. of the Third Int. Conf. on Genetic Algorithms, pp. 98-103, George Manson University, 1989.

[7] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, 1996.

*Universidade Trás-os-Montes e Alto Douro • Secção de Engenharias • 5001 Vila Real, Portugal • epires@utad.pt

**Instituto Politécnico do Porto • Instituto Superior de Engenharia • Departamento de Engenharia Electrotécnica Rua de S. Tomé, 4200 Porto, Portugal • jtm@dee.sep.ipp.pt