ALGORITMOS GENÉTICOS APLICADOS À MODELAGEM ÓTIMA DE PROBLEMAS DE PLANEJAMENTO E UM ESTUDO DE CASO

Roger Vinícius Garcia
ITA - Instituto Tecnológico de Aeronáutica
Rua H8-B, nº224, Campus do CTA, São José dos Campos - SP, Brasil, CEP 12228-461
Bolsista PIBIC-CNPq
rogergarcia88@gmail.com

Paulo Marcelo Tasinaffo
ITA - Instituto Tecnológico de Aeronáutica – Divisão de Ciência da Computação
Pç. Marechal Eduardo Gomes, nº 50, São José dos Campos – SP, Brasil, CEP 12228-900
tasinaffo@ita.br

Resumo – *Este trabalho pretende aplicar algoritmos genéticos como técnica de otimização para selecionar a melhor escolha em problemas de planejamento que envolvem grandeza físicas, por exemplo, o tempo e o gasto de combustível, e que devem ser minimizadas para obtenção de uma solução ótima adequada para estes tipos de problemas. Algoritmos genéticos, com números de indivíduos da população constante em cada geração seguinte, são propostos para irem excluindo soluções inadequadas e propensas a mínimos locais. Os algoritmos genéticos serão então aplicados ao problema de consumo de combustível e planejamento de postos de gasolina temporários realizados por um caminhão que almeja cruzar um deserto, na situação em que atravessá-lo numa única viagem é impossível, por se tratar de um percurso demasiado extenso e acidentado para sua capacidade total de transporte de combustível.*

1. Introdução

A ciência surge dos muitos desejos humanos para compreender e controlar o mundo. Deste modo, ao longo da história foi-se construindo gradualmente um grande edifício de conhecimento capacitando-nos a predizer – entre muitas outras coisas – o tempo meteorológico, os movimentos dos planetas, o eclipse lunar e solar, a proliferação geográfica de doenças contagiosas, o crescimento e queda de sistemas econômicos, entre muitos outros fenômenos dentro do panorama social e econômico (Mitchell, 1996). Em tempos recentes foi também comprovado que existe um limite – mesmo em nível matemático - na habilidade de predizer algo. Deste modo alguns aspectos da natureza se demonstraram previsíveis, enquanto outros completamente imprevisíveis.

Com o advento do computador, nos últimos cinqüenta anos o desenvolvimento tecnológico humano demonstrou revolucionar os horizontes de alcance para predizer fenômenos naturais. A meta de se criar inteligência artificial pode ser traçada desde os primórdios da era computacional. Os cientistas que primeiro imaginaram isto foram Alan Turing, John von Neumann, Norbert Wiener, entre outros. A primeira grande revolução foi o surgimento do campo das redes neurais artificiais (Nascimento e Yoneyama, 2000), a segunda do aprendizado de máquinas e a terceira da computação evolutiva (Mitchell, 1996), em que os algoritmos genéticos são os exemplos mais conhecidos.

Nos anos 50 e 60 vários cientistas estudaram de forma independente sistemas evolucionários com a idéia de que a teoria da evolução de Darwin poderia ser utilizada como uma ferramenta de otimização para problemas de engenharia (Mitchell, 1996; Russell e Norvig, 2003). A idéia principal consiste em criar uma população de candidatos à solução para um dado problema e, utilizando operadores matemáticos inspirados na variação genética e seleção natural, encontrar uma solução ótima para um problema particular.

Durante os anos 60 e 70, Rechenberg em 1965 e 1973 introduziu estratégias evolucionárias, um método para ser utilizado na otimização de parâmetros de valores reais com aplicação em engenharia (Mitchell, 1996). Esta idéia foi mais tarde desenvolvida por Schwefel em 1975 e 1977. Fogel, Owens, e Walsh em 1966 desenvolveram a programação evolucionária, uma técnica em que candidatos à solução para uma dada tarefa representam uma máquina de estados finitos, que foi envolvida por mutações randômicas de seus diagramas de transição de estados e assim, elaborar uma seleção em direção à população dos mais adaptados. Conjuntamente, estratégias evolucionárias, programação evolutiva e algoritmos genéticos formam o leque de atuação do campo da computação evolutiva.

Entre outros pesquisadores que trabalharam na elaboração de algoritmos de otimização evolucionários e aprendizado de máquina citam-se: Box em 1957, Friedman em 1959, Bledsoe em 1961, Bremermann em 1962 e Baricelli em 1967. Assim, computação evolutiva está definitivamente presente desde os primórdios de criação do computador eletrônico (Mitchell, 1996). Algoritmos genéticos foram inventados por John Holland no início dos anos 60 e foi utilizado pelos seus estudantes da universidade de Michigan nos anos 60 e 70. Em 1975, Holland, em seu livro *Adaptation in Natural and Artificial Systems,* apresenta o algoritmo genético como uma abstração da evolução biológica dando uma estrutura teórica sobre o próprio algoritmo genético, como solução geral para muitos problemas de otimização do mundo real.

Neste trabalho pretende-se desenvolver um estudo sobre a aplicação de Algoritmos Genéticos a um problema-exemplo, analisando desde as heurísticas utilizadas para sua resolução até os resultados obtidos com a implementação dos algoritmos. Assim, na seção 2 é realizado um desenvolvimento descritivo e teórico da Computação Evolutiva e dos Algoritmos Genéticos; na seção 3 é formulado matematicamente o problema proposto e a heurística proposta para a construção da função *fitness* utilizada pelos Algoritmos Genéticos; na seção 4 são apresentados os resultados e, na seção 5, as conclusões.

2. Desenvolvimento Teórico da Computação Evolutiva e dos Algoritmos Genéticos

Algoritmo genético é uma designação para arquiteturas de algoritmos que atuam sobre problemas de otimização, inspiradas na Teoria da Evolução de Darwin, na qual os indivíduos mais adaptados a um meio tendem a sobreviver nele e perpetuar seu código genético através das gerações. Neste sentido, o algoritmo deve atuar sobre uma população de indivíduos, sendo cada um relacionado a uma solução (potencialmente ótima) para o problema proposto (Alves da Silva, 2003). Através de restrições feitas aos indivíduos, análogas à hostilidade do meio para os indivíduos biológicos, os melhores entre eles vão sendo naturalmente classificados para transmitir suas informações para as próximas gerações, onde se busca uma convergência para um "indivíduo perfeito", isto é, uma solução quase ótima para o problema.

As vantagens dos algoritmos genéticos sobre outras arquiteturas de busca de máximos (ou mínimos) são a sua capacidade "estocástica" de análise sobre os pontos do domínio, evitando assim máximos ou mínimos locais, e a sua capacidade de economizar processamento por selecionar apenas uma amostra dos pontos a serem analisados. Além disso, por ser um algoritmo de busca de ordem zero, não envolve derivadas na modelagem do problema, o que permite a ele ser aplicado sobre funções de seleção com descontinuidades de primeiro grau e aplicações discretas.

Nas próximas subseções serão definidas as técnicas canônicas utilizadas na criação de um algoritmo genético para um problema qualquer.

2.1 Codificação

A produção da população de um número fixo de candidatos a solução do problema proposto envolve a codificação das informações sobre o domínio da solução em linguagem de máquina, geralmente em nível binário. Deste modo, cada indivíduo é representado por um vetor de números binários, correspondente a um valor real de uma solução numérica ou associado a um item de uma tabela de características, dependendo da modelagem do problema. Os indivíduos podem ser gerados aleatoriamente, dentro do domínio considerado, ou através de heurísticas especiais para esse fim. O espalhamento inicial dos indivíduos sobre o espaço de procura permite ao algoritmo aproximar-se da solução ótima de forma multidirecional, evitando prender-se a máximos ou mínimos locais. A determinação do número de bits necessários para a representação do indivíduo depende da discretização e do tamanho do espaço de procura. Por exemplo, seja a função:

$$\begin{aligned} \mathrm{f} : [a,b] &\to \mathbb{R} \\ x &\mapsto \mathrm{f}(x) \end{aligned}$$

Se o domínio puder ser discretizado em um conjunto A de pontos, estando disponível um número m de bits na memória para representar cada um deles, pode-se definir uma distância mínima padrão $\delta = (b-a)/(2^m - 1)$, maior ou igual à precisão disponível na máquina. Assim, cria-se um novo domínio para a mesma função, $A = \{x \in \mathbb{R} / x = a + k\delta,\ k = 0,1,...,2^m\text{-}1\}$, e tem-se:

$$\begin{aligned} \mathrm{f} : A &\to \mathbb{R} \\ x &\mapsto \mathrm{f}(x) \end{aligned}$$

Neste caso, indivíduos vizinhos estão eqüidistantes entre si, e cada um deles pode ser representado por uma cadeia de m bits, abrangendo todo o domínio A da função: $x = (b_0, b_1, b_2, ..., b_{m-1})$, $b_i = \{0,1\}$, i = 0, 1, ..., m-1, cujo valor real é:

$$x = a + \sum_{i=0}^{m-1} 2^i b_i \delta \tag{1}$$

É claro que outras heurísticas de discretização de um intervalo poderiam ser usadas, não se valendo obrigatoriamente da condição de eqüidistância entre pontos. Para um caso mais geral, pode-se aplicar um algoritmo genético à solução de um problema com n variáveis, de conjuntos A_i discretos, caracterizando-o como um problema de *dimensão n.* A solução é, então, um vetor de números reais x, e a função de avaliação deve ser da forma:

$$\begin{aligned} \tilde{\mathrm{f}} : A_1 \times A_2 \times ... \times A_n &\to \mathbb{R} \\ \mathbf{x} &\mapsto \tilde{\mathrm{f}}(\mathbf{x}) \end{aligned}$$

Para a representação binária dos indivíduos em um problema de dimensão n, basta concatenar as cadeias de números binários referentes a cada dimensão, desde que se conheça o número de bits associado a cada uma delas.

2.2 Seleção: Função *Fitness*

Depois de representados em forma binária, os indivíduos devem ser selecionados segundo a sua proximidade a uma possível solução ótima. Para isso, deve-se utilizar uma função *fitness,* ou *função objetivo*, a qual se deseja maximizar, e que representa a heurística da resolução do problema. Essa função foi exemplificada como f e $\tilde{\mathrm{f}}$ nos exemplos anteriores. Cada um dos indivíduos da

primeira população deve ser fornecido como argumento da função *fitness*, retornando um número real. Quanto maior o valor obtido na avaliação, maior é a probabilidade de o ponto analisado estar próximo a um máximo local ou global.

Em uma analogia com a teoria da seleção natural, os indivíduos mais adequados ao problema proposto devem ter chance maior de repassar seu código genético a futuras gerações, ou seja, transmitir seu código binário à população da próxima iteração. Computacionalmente, atribui-se a cada indivíduo uma probabilidade de ser escolhido para a reprodução, baseada no desempenho deste na função avaliadora. Assim, seja uma população de p indivíduos, e $f(x_i)=d_i$, com i=1,2,...,p, o desempenho de cada um. A probabilidade P de escolha de um indivíduo é definida como sendo:

$$P_i = \frac{d_i}{\sum_{k=1}^{p} d_k} \tag{2.1}$$

Observa-se que a soma das probabilidades é unitária:

$$\sum_{k=1}^{p} P_k = 1 \tag{2.2}$$

Percebe-se que, quanto maior o valor da função *fitness* para um dado indivíduo, maior é a probabilidade de que este seja escolhido para a reprodução.

2.3 *Crossover* e reprodução

Para que a primeira população (de escolha estocástica) seja refinada segundo o critério do aumento da função *fitness*, selecionam-se os melhores indivíduos da população para se reproduzirem e darem origem a uma nova população que, na média, estará um pouco mais próxima de uma solução ótima. Para ocorrer a reprodução, devem ser selecionados dois pais da população anterior, sorteados em uma distribuição uniforme, e contemplados segundo sua probabilidade de escolha P. Atente-se ao fato de que um mesmo pai poderá ser selecionado diversas vezes, se seu valor *fitness* for alto, assim como um indivíduo de baixo *fitness* pode não ser contemplado no sorteio. Ocorrendo p seleções de progenitores, dar-se-á início a p/2 cruzamentos, visto que cada cruzamento dá origem a 2 novos indivíduos.

A cada par de pais selecionado, é escolhido estocasticamente um ponto entre o i-ésimo e o (i+1)-ésimo bit de sua representação binária. Então, executa-se o *crossover*, que consiste na operação de comutar blocos de bits entre os dois indivíduos, a partir do bit i+1. Matematicamente, se são escolhidos os pais x e y:

$$\begin{aligned} x &= (b_0, b_1, b_2, ..., b_{m-1}) \\ y &= (c_0, c_1, c_2, ..., c_{m-1}) \end{aligned} \tag{3.1}$$

Então, após o *crossover*, têm-se dois novos indivíduos:

$$\begin{aligned} \overline{x} &= (b_0, b_1, b_2, ..., b_i, c_{i+1}, c_{i+2}, ..., c_{m-1}) \\ \overline{y} &= (c_0, c_1, c_2, ..., c_i, b_{i+1}, b_{i+2}, ..., b_{m-1}) \end{aligned} \tag{3.2}$$

Esse processo tem como objetivo combinar características de soluções distintas, que possam se combinar e convergir para a solução ótima. Além disso, previne a convergência para um ponto de máximo local, dando a oportunidade de diferentes soluções serem testadas e miscigenadas entre as melhores, fazendo com que o algoritmo explore regiões diversas do espaço de busca. Assim, ao final do processo de reprodução, haverá uma nova população de p indivíduos, mais próxima do máximo global.

2.4 Mutação

Para evitar que o algoritmo convirja para um máximo local, aplica-se o processo de mutação à nova população. Com isso, o algoritmo é desviado a locais ligeiramente diferentes do original para a procura, dando oportunidade para que um caminho para a solução ótima seja encontrado. O análogo biológico prevê pequenas alterações do código genético, que levam a mudanças importantes na evolução das espécies.

Computacionalmente, define-se um fator de mutação, p_m, a ser aplicado sobre a nova população. O algoritmo percorre todos os bits de todos os indivíduos, executando uma operação NOT em cada bit a uma probabilidade p_m, geralmente definida entre 2% e 4%. Para populações maiores, é necessário um fator de mutação menor. Caso seja aplicado um fator demasiado grande, a procura pela solução degenerar-se-á, não convergindo para uma solução satisfatória.

2.5 Formação da nova população e critério de parada

Depois de executada a mutação, a nova população é testada na função *fitness*, e seus resultados são comparados com a população anterior. Destes 2p indivíduos, abrangendo a população inicial e a nova população, os p melhores classificados são selecionados, para compor a população que será utilizada na iteração seguinte. O critério de parada do algoritmo pode ser definido como uma alteração

muito pequena, de uma iteração para a outra, do valor *fitness* do indivíduo melhor colocado. Critérios diferentes também podem ser adotados.

3. Definição Matemática do Problema Proposto e Heurísticas para a Função de *Fitness* Utilizada no Algoritmo Genético

O problema que se pretende analisar e resolver é o do "*Caminhão no Deserto"* (Wood, 1986 e 1988) e que pode ser enunciado da seguinte forma:

"Um caminhão atinge a extremidade de um deserto de 400 milhas de extensão. O veículo faz apenas 1 milha com um galão de gasolina, e a capacidade total do caminhão, incluindo bujões adicionais, é de 180 galões; assim sendo, terão de ser providenciados depósitos provisórios no deserto. Existe gasolina à vontade na extremidade do deserto. Se fizermos um bom planejamento, qual é o menor consumo de gasolina necessário para que o veículo consiga atravessar o deserto?"

Por se tratar de um problema de otimização e planejamento não-linear, em geral, partir de um único ponto em busca de um mínimo, por exemplo, através de algoritmos de gradiente, Quasi-Newton, Levemberg-Marquardty, etc., (Fritzsche, 1978; Wismer e Chattergy, 1978) limitam bastante o espaço de busca, pois, em geral, estes problemas permitem a existência de vários mínimos locais. Geralmente, os algoritmos considerados convergem para o mínimo local mais próximo e que poderá não ser suficiente (Mitchell, 1996) para a solução do problema proposto. O que se pretende aqui, portanto, é utilizar uma heurística sobre os algoritmos genéticos que irá avaliar melhor o espaço de busca em direção ao mínimo global.

A principal diferença entre os algoritmos que utilizam derivadas dos que não utilizam está no fato dos primeiros partirem inicialmente de vários pontos distintos, ao mesmo tempo, em busca do mínimo, enquanto os últimos partem de um único ponto (Russell e Norvig, 2003). Em geral, os algoritmos genéticos avaliam vários pontos ao mesmo tempo, enquanto os demais *seguem uma direção*, por exemplo, da descida mais íngreme. Os primeiros devem antes processar muitos pontos até atingir um mínimo satisfatório, enquanto os últimos são *lerdos* na direção em que estão seguindo.

O problema proposto é n-dimensional nas variáveis $x_1,x_2,...,x_n$, onde $n \geq 3$ é o número total de postos provisórios e x_i para i = 1,2,...,n é a posição de cada posto no deserto em relação à partida. A simulação considerada aqui avaliará o mínimo para cada n e escolherá como melhor solução o mínimo dos mínimos. Observe que os postos provisórios não necessariamente estarão eqüidistantes um do outro e os algoritmos genéticos deverão estimar a melhor localização no deserto para cada posto, caracterizando o problema proposto como um de minimização do espaço contínuo. Deste modo, a discretização do problema – necessária para implementar os algoritmos genéticos – será de grande importância, pois uma discretização baixa poderá ser imprecisa e inadequada, enquanto uma alta discretização poderá tornar o problema insolúvel do ponto de vista computacional, devido ao elevado tempo de processamento para uma precisão muito elevada e desnecessária. Assim, matematicamente pode-se definir o problema como segue:

Sejam A_i conjuntos discretizados de pontos do intervalo $I = [0, L]$, uma constante $c \in \mathbb{R}$ e seja $f : A_1 \times A_2 \times ... \times A_n \times \mathbb{R} \rightarrow \mathbb{R}$ uma função fitness. Encontrar uma n-upla $(x_1, x_2, ..., x_n) \in A_1 \times A_2 \times ... \times A_n$ tal que $f(x_1, x_2, ..., x_n, c)$ seja mínimo.

No caso do problema apresentado, L = 400 milhas é o comprimento do deserto, n é o número de postos provisórios e c = 180 galões é a capacidade de transporte do caminhão. O primeiro passo para a resolução do problema é encontrar a heurística que defina a função *fitness* que se quer minimizar – neste caso, o consumo de combustível. Assim, sejam os n postos, suas posições x_i e distância d_i, representados na Fig 1:

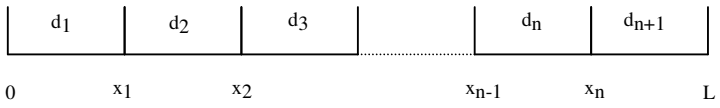

Figura 1. Representação do posicionamento dos postos no deserto.

Para calcular a quantidade de combustível necessária, deve-se dividir o problema em submetas: encontrar a quantidade de combustível necessária em cada posto, regressivamente de x_n até x_1, para que o caminhão possa efetuar a travessia. Desse modo, logo se percebe que o combustível necessário em x_n é de d_{n+1} galões, desde que o caminhão chegue a x_n de tanque vazio, ou que a soma do combustível do posto x_n com o combustível já estocado no caminhão seja exatamente d_{n+1}. Perceba que o caminhão deve chegar a L com o tanque vazio; caso contrário, haveria desperdício de combustível. Tem-se, portanto, uma condição para o problema:

$$Comb(x_n) = d_{n+1}, \tag{4}$$

em que $Comb(x_i)$ representa o combustível necessário no posto x_i. Caso o consumo do caminhão fosse diferente de 1 galão/milha, bastaria multiplicar um fator k adequado na equação. Sabendo que o armazenamento máximo do caminhão é de c galões, e estando ele no posto x_i, então se pode estocar no máximo $(c - 2d_{i+1})$ galões no posto x_{i+1} a cada viagem de ida e volta, uma vez que se devem gastar $2d_{i+1}$ galões no percurso. No entanto, se a quantidade d_{i+1} mais a necessária x_{i+1} de galões for menor ou igual à capacidade c do caminhão, então ele pode se abastecer em x_i e transportar todo o combustível necessário em uma só viagem, sem voltar a x_i. Matematicamente:

$$\left(Comb(x_{i+1}) + d_{i+1} \le c\right) \Rightarrow d_i = Comb(x_{i+1}) + d_{i+1} \tag{5}$$

Em um raciocínio análogo, se o caminhão já tiver feito algumas viagens de ida e volta até o posto seguinte, e restar uma quantidade de combustível para ser transportada tal que obedeça à restrição (5), ele pode fazer mais uma viagem apenas, e completar o transporte de x_i para x_{i+1}. Assim, o número de viagens completas (ida e volta) necessárias antes de o critério (5) ser satisfeito é N:

$$N = \frac{Comb(x_{i+1}) + d_{i+1} - c}{c - 2d_{i+1}} \tag{6}$$

Como N pode ser um número fracionário, deve-se considerar como o número de viagens completas o menor inteiro maior ou igual a N, ou seja, o teto de N, representado por $\lceil N \rceil$. Assim, depois de $\lceil N \rceil$ viagens, em cada uma sendo entregues ($c - 2d_{i+1}$) galões, o número R de galões necessários para se cumprir a cota de entrega de $Comb(x_{i+1})$ será:

$$Comb(x_{i+1}) = R + \lceil N \rceil (c - 2d_{i+1}) \quad \Rightarrow \quad R = Comb(x_{i+1}) - \lceil N \rceil (c - 2d_{i+1}) \tag{7}$$

E, portanto, o número de galões em x_i deverá ser $\lceil N \rceil$ vezes c (um tanque cheio com c galões de combustível a cada viagem completa para carregamento), somado à quantidade R de galões faltantes, que pode ser entregue em apenas uma viagem de ida, já que satisfaz à restrição (5), somados à distância d_{i+1} que se precisa percorrer nesta última viagem. Portanto:

$$Comb(x_i) = \lceil N \rceil c + R + d_{i+1} = \lceil N \rceil c + Comb(x_{i+1}) - \lceil N \rceil (c - 2d_{i+1}) + d_{i+1} \tag{8}$$

E, assim,

$$Comb(x_i) = Comb(x_{i+1}) + d_{i+1}(2\lceil N \rceil + 1) \tag{9}$$

Desta forma, partindo da quantidade já fixada de combustível para o posto x_{i+1} e calculando a quantidade de combustível dos postos antecedentes, chega-se ao total necessário na borda, ou seja, $Comb(x_0)$. Deve-se notar que, devido à limitação da capacidade c de combustível do caminhão, os postos x_i e x_{i+1} devem estar distantes entre si menos de ($c - 2d_{i+1}$), ou o caminhão não será capaz de estocar galões de combustível em x_{i+1}. O último posto, no entanto, pode estar a uma distância de até c do final do deserto, já que essa distância final deve ser percorrida apenas uma vez. Matematicamente, têm-se as condições:

$$\begin{cases} 0 < d_i < \dfrac{c}{2}, & i = 1, 2, ..., n \\ 0 < d_{n+1} \le c \end{cases} \tag{10}$$

Com a função *fitness* e as restrições necessárias em mãos, já se pode implementar um algoritmo genético para a resolução do problema.

4. Resultados e Discussão

Por se tratar de um problema de procura do mínimo de uma função, a heurística aplicada à determinação da probabilidade de escolha dos pais deve ser o inverso numérico do desempenho na função *fitness*. Em outras palavras, quanto menor o valor f(x) (*fitness*) obtido pelo indivíduo, maior a probabilidade de ele ser escolhido:

$$\mathrm{f}(x_i) = d_i \quad \Rightarrow \quad P_i = \frac{d_i^{\,m}}{\sum_{k=1}^{p} d_k^{\,m}}, \; m < 0 \tag{11}$$

Foram testados diferentes valores de m na resolução do problema, para observar o comportamento da convergência. Na Figura 2, são mostrados os exemplos para m=-0,66, m=-0,5, m=-0,33 e m=-0,1. Cada linha representa o comportamento da melhor solução encontrada para cada dimensão do problema (no caso, para dimensão n=4).

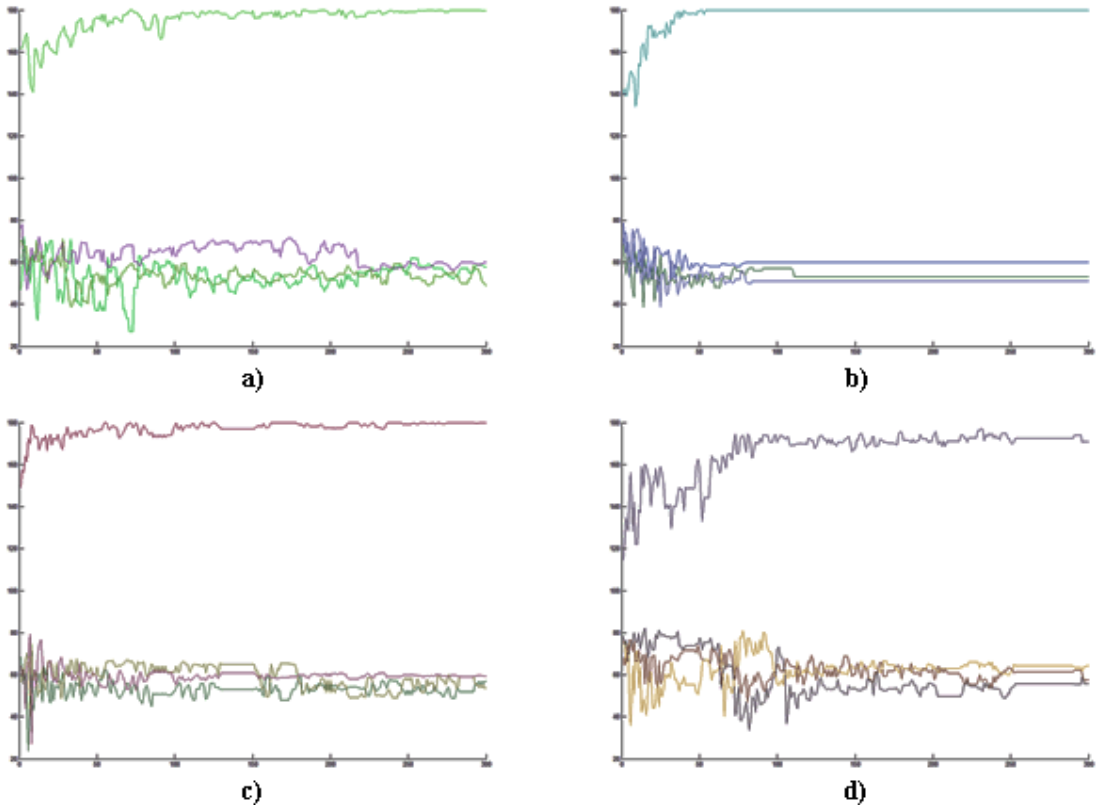

Figura 2. Gráfico (valor de distância de cada dimensão versus número de iterações) da convergência da solução para cada dimensão, em n=4. a) m=-0,66; valor fitness: 4496 galões. b) m=-0,5; valor fitness: 4138 galões. c) m=-0,33; valor fitness: 4424 galões. d) m=-0,1; valor fitness: 5400 galões.

Para valores mais próximos a -1, o algoritmo apresentou problemas de convergência. Para valores mais próximos a 0, o algoritmo tende a procurar o máximo global, e não o mínimo, como explicado na seção 2. Pelo gráfico, observa-se que valores de m diferentes de -0,5 tendem a ser turbulentos a uma mesma taxa de mutação, enquanto m=-0,5 converge antes de 150 iterações. Além disso, m=-0,5 atingiu o menor valor *fitness*. Por isso, no decorrer dos testes posteriores, foi adotado esse valor para m.

Assim, foram testados diferentes valores para o número n de postos. Observou-se que, quanto maior o número n, maior era a população de soluções necessária para que o algoritmo apresentasse soluções quase-ótimas. Conseqüentemente, o fator de mutação teve de decrescer com o aumento da população, para não gerar turbulências na convergência das soluções. Muitas vezes, foram necessários ajustes empíricos nesses valores, à medida que n aumentava, a partir da observação do gráfico de comportamento das soluções ao longo das iterações do algoritmo. O algoritmo foi executado para cada número n de postos, de 3 a 13, dez vezes. Os resultados obtidos são mostrados na Tabela 1.

Tabela 1. Gasto de combustível para as configurações ótimas de diferentes números de postos.

Número de postos	Mínimo gasto de combustível (galões)
3	14576
4	4138
5	3000
6	2626
7	2506
8	2456
9	2406
10	2304
11	2266
12	2274
13	2218

Para valores de n maiores que 13, o processamento sobre a heurística utilizada para o cálculo tornam inviável a execução do algoritmo, podendo esta ser otimizada em trabalhos futuros. Pela observação da Tabela 1, espera-se que, quanto maior o número de

postos provisórios instalados no deserto, maior seja a economia de combustível, desde que esses postos obedeçam a uma configuração de máxima performance. A Figura 3 mostra a convergência da solução ótima para n=13:

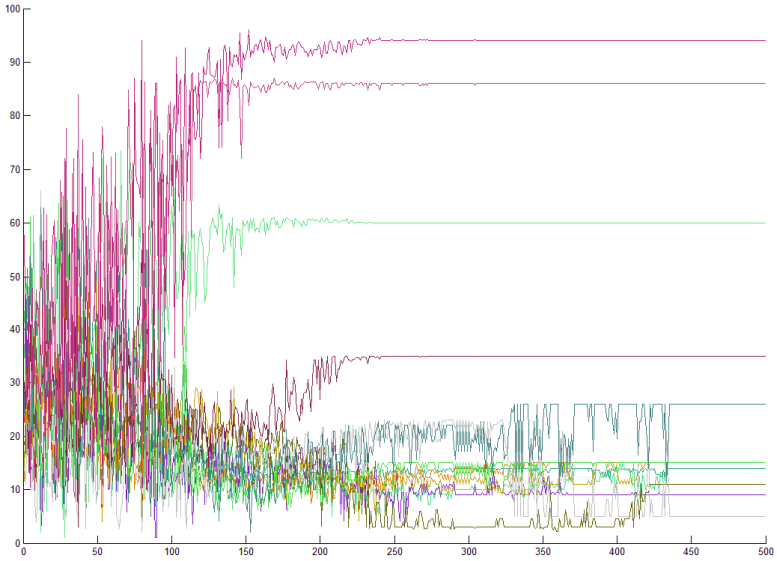

Figura 3. Gráfico (valor de distância de cada dimensão versus número de iterações) da convergência da solução para cada dimensão em n=13.

Como determinação da eficiência do algoritmo na diminuição do custo de combustível, pode-se analisar o gráfico do gasto total versus número de postos, na Figura 4. O ponto n=3 foi omitido por ter um valor *fitness* associado muito discrepante dos outros pontos considerados.

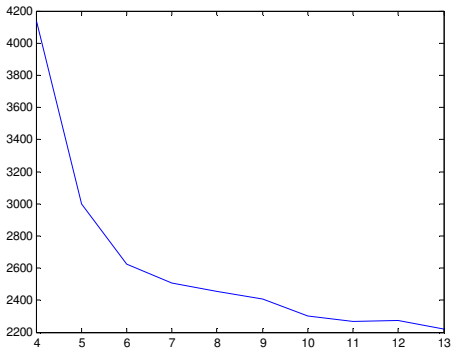

Figura 4. Gráfico (galões versus postos) do combustível total gasto na configuração quase-ótima para diferentes valores de n.

Pela análise do gráfico, espera-se que a economia seja cada vez menor ao longo das iterações subseqüentes, tendendo assintoticamente a um mínimo dos mínimos em um comportamento exponencial amortecido. A configuração ótima n=13 obtida para a resolução do problema, lançando mão dos recursos computacionais disponíveis, é mostrada na Tabela 2. Também é mostrado o número de viagens de ida e volta necessárias em cada trecho do percurso (notar que o trecho entre o posto 13 e o final do percurso é percorrido apenas uma vez, na ida).

Tabela 2. Valores encontrados para as posições ótimas dos postos.

Trecho	Distância (milhas)	Viagens ida e volta
Início a x_1	3	12
x_1 a x_2	11	12
x_2 a x_3	15	11
x_3 a x_4	9	10
x_4 a x_5	11	8
x_5 a x_6	15	7
x_6 a x_7	15	6
x_7 a x_8	5	5
x_8 a x_9	15	4
x_9 a x_{10}	26	4
x_{10} a x_{11}	35	3
x_{11} a x_{12}	60	2
x_{12} a x_{13}	86	1
x_{13} ao final	94	0,5 (somente ida)

5. Conclusões

Embora o resultado alcançado seja bastante eficiente e adequado para o planejamento e organização dos postos no deserto, ainda seria possível uma discretização maior do espaço de busca, nas vizinhanças do ponto ótimo encontrado. Assim, se o resultado tivesse uma precisão de meia milha, a economia de combustível seria maior, e daí em diante, até a precisão máxima alcançável do ponto de vista prático.

Foi de grande importância a determinação da taxa de mutação correta, dependendo da população utilizada para cada valor da dimensão do problema. Grandes mutações impedem o algoritmo de convergir, enquanto pequenas mutações viciam o algoritmo em máximos ou mínimos locais, impossibilitando a obtenção do ponto ótimo global. A determinação da mutação, neste problema, foi efetuada principalmente pela análise do gráfico e do comportamento das curvas ao longo das iterações, e pela constatação de que populações grandes exigem uma diminuição na taxa de mutação, ou o algoritmo degenerar-se-á.

A turbulência inicial é característica da procura pela solução, mas não pode persistir até o fim das iterações. Sabe-se que um algoritmo convergiu quando suas soluções ficam estagnadas a uma linha reta, ou seja, não há mudanças significativas na solução por um certo número de iterações. Por fim, para aplicações do algoritmo para dimensões maiores que 13, espera-se que a quantidade de combustível gasta decresça lentamente até uma assíntota. Neste caso, o custo computacional de processamento e de dificuldade da determinação dos parâmetros necessários para uma convergência quase-ótima da solução dificilmente justificam a pequena economia de combustível conseguida pelo algoritmo.

6.Agradecimentos

Agradeço ao Prof. Dr. Paulo Marcelo Tasinaffo pelo imenso apoio e ensinamentos, que tornaram possível não só a conclusão deste trabalho, como também um grande aprendizado na área de inteligência artificial. Agradeço também ao CNPq que, através do suporte financeiro, torna possível a execução deste e de outros trabalhos, contribuindo para a formação de jovens pesquisadores brasileiros.

7. Referências Bibliográficas

Alves da Silva, A. P. Tutorial Genetic Algorithms. *Learning and Nonlinear Models.* Vol. 1, No. 1, pp. 43-58, 2003.

Fritzsche, H. *Programação Não-Linear análise e métodos*. São Paulo: Edgard Blücher: Ed. da Universidade de São Paulo, 1978.

Mitchell, M. *An Introduction to Genetic Algorithms.* The MIT Press: Massachusetts Institute Technology, 1996.

Nascimento Jr., C. L.; Yoneyama, T. *Inteligência Artificial em Controle e Automação*. 1. ed., São Paulo: Edgard Blucher Ltda., Fapesp, 2000.

Russell, S.; Norvig, P. *Artificial Intelligence A Modern Approach*. Second Edition, New Jersey: Pearson Education, 2003.

Wismer, D. A.; Chattergy R. *Introduction to Nonlinear Optimization A Problem Solving Approach*. Second Printing, New York: Elsevier North Holland, Inc., 1978.

Wood, L. E. *Estratégias do Pensamento*: *Técnicas de Aptidão Mental*. São Paulo, SP, Brasil: Círculo do Livro, 1988.

Wood, L.E. *Thinking Strategies*: *Exercises for Mental Fitness*. Englewood Cliffs, NJ: Prentice-Hall, 1986.