ALGORITMOS GENÉTICOS PARA RESOLVER PROBLEMAS DE PROGRAMAÇÃO NÃO-LINEAR COM INCERTEZAS

Ricardo Coelho Silva
Departamento de Telemática
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas – UNICAMP
Caixa Postal 6101, 13081-970, Campinas – SP
rcoelhos@dt.fee.unicamp.br

Luiza Amalia Pinto Cantão
Departamento de Engenharia Ambiental
Universidade Estadual Paulista – UNESP
Av. Três de Março, 511, 18087-180, Sorocaba – SP
luiza@sorocaba.unesp.br

Akebo Yamakami
Departamento de Telemática
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas – UNICAMP
Caixa Postal 6101, 13081-970, Campinas – SP
akebo@dt.fee.unicamp.br

RESUMO

Otimização é um procedimento de encontrar e comparar soluções factíveis até nenhuma solução melhor pode ser encontrada. Ela tem aplicação em uma ampla variedade de problemas do mundo real, mas seus parâmetros, muitas vezes, não podem ser formulados precisamente. Assim, a teoria de conjuntos nebulosos faz total sentido aplicá-lo como uma maneira de descrever matematicamente essa incerteza. Dois algoritmos genéticos foram desenvolvidos neste trabalho para resolver uma classe de problemas de programação matemática com incertezas na função objetivo e no conjunto de restrições. Aqui foi introduzida uma nova abordagem para restrições de igualdade com incerteza. Alguns problemas foram selecionados da literatura para validar a eficiência dos algoritmos genéticos descritos neste trabalho.

PALAVRAS CHAVE. Teoria Fuzzy. Otimização Não-Linear. Programação Matemática Fuzzy.

ABSTRACT

Optimization is a procedure of finding and comparing feasible solutions until no better solution can be found. It has applications in a wide range of real-world problems, but many times their data cannot be formulated precisely. Hence it makes perfect sense apply the fuzzy set theory as a way to describe mathematically this vagueness. In this work we develop two meta-heuristic algorithms that solve a class of mathematical programming problems with uncertainties in the objective function and in the set of constraints. We introduce a novel approach to the equality constraints with uncertainties. Selected examples from the literature are presented to validate the efficiency of the addressed algorithms.

KEYWORDS. Fuzzy Theory. Nonlinear Optimization. Fuzzy Mathematics Programming.

1. Introdução

Programação matemática é usada para encontrar a melhor solução de um problema, teórico e prático, que minimiza (ou maximiza) uma função objetivo sujeita ou não a algumas restrições. O conjunto de restrições de um problema de programação matemática pode conter três maneiras de função: (i) restrições de igualdade; (ii) restrições de desigualdade; (iii) restrições mistas, que contém as duas maneiras anteriores no mesmo conjunto. Quando o objetivo e o conjunto de restrições são formados somente por funções lineares, esse problema pertence a um grupo chamado de Programação Linear. Caso contrário, esse problema pertence a outro grupo chamado Programação Não-Linear. A maioria dos problemas da vida real, que são modelados como programação matemática, pertencem a esse último grupo. Outro fato comum nesses problemas é a obtenção dos dados, que contêm algum tipo de incerteza. Nesse contexto, a teoria de conjuntos nebulosos ajuda no tratamento desses dados incertos. Assim, um problema de programação matemática com parâmetros nebulosos pode ser formulado como:

$$\begin{array}{lll} \min & f(\tilde{c};x) & \\ s.a. & g_k(\tilde{a};x) \tilde{\leq} \tilde{b}_k & k=1,\ldots,m \\ & h_l(\tilde{d};x) \tilde{=} \tilde{e}_l & l=1,\ldots,n \\ & x \in \Omega & \end{array} \qquad (1)$$

onde $\Omega \subseteq \mathrm{IR}^n$ e, $\tilde{a},\tilde{b},\tilde{c},\tilde{d}$ e $\tilde{e}$ representam os parâmetros nebulosos na função objetivo e conjunto de restrições do problema a ser otimizado, enquanto $\tilde{\leq}$ e $\tilde{=}$ representam a incerteza nas relações de ordem do conjunto de restrições. Entretanto, neste trabalho iremos focar somente em parâmetros nebulosos nos custos da função objetivo e nas relações de ordem do conjunto de restrições.

Em recentes anos, o uso da Computação Flexível (*Soft Computing*) mostrou um grande potencial para modelar sistemas que são não-lineares, complexos e mal-definidos. Ela é composta pela lógica nebulosa, argumentação probabilística, neuro-computação e metaheurísticas. A computação Flexível emerge como uma ferramenta apropriada para resolver problemas de alta complexidade para a computação clássica. Além do uso de lógica nebulosa, um algoritmo genético é usado também neste trabalho. Os algoritmos genéticos usam o princípio da evolução como um método de busca por soluções candidatas de problemas de otimização. Algumas estratégias para obter uma evolução satisfatória em algoritmos genéticos estão descritas em Goldberg(1989) e Michalewics(1996). Algoritmos genéticos têm obtido um considerável sucesso nas últimas décadas, que são aplicados a grandes grupos de problemas, tal como controle ótimo, transportes, problemas de escalonamento, problemas de otimização entre outros. A implementação dos algoritmos propostos neste trabalho são baseados em Liu e Iwamura(1998a) e Liu e Iwamura(1998b), com algumas modificações.

Esse trabalho está dividido da seguinte forma: na Seção 2 são detalhados os tipos de restrições, função de *fitness* e nível de satisfação; na Seção 3 são apresentadas duas versões de algoritmos genéticos para o problema proposto; As simulações numéricas dos problemas selecionados e uma análise dos resultados obtidos são apresentadas na Seção 4; na Seção 5, finalmente, são apresentadas as conclusões deste trabalho.

2. Critérios de decisão

Os algoritmos de otimização desenvolvidos neste trabalho necessitam estabelecer critérios de decisão para guiar a busca da solução ótima dos problemas. Esses critérios são apresentados em três formas: (i) satisfação do conjunto de restrições (factibilidade); (ii) valor de *fitness*; (iii) nível de satisfação.

Na sub-seção 2.1 são descritos os tipos de restrições aplicadas aos algoritmos genéticos da Seção 3. A medida de *fitness* para cada indivíduo da população é apresentada na sub-seção 2.2. Na sub-seção 2.3 é apresentado nível de satisfação de cada indivíduo como uma solução do problema.

2.1. Conjunto de restrições *fuzzy*

Problemas de programação matemática podem ter três tipos de restrições: (i) restrições de igualdade; (ii) restrições de desigualdade; (iii) restrições mistas.

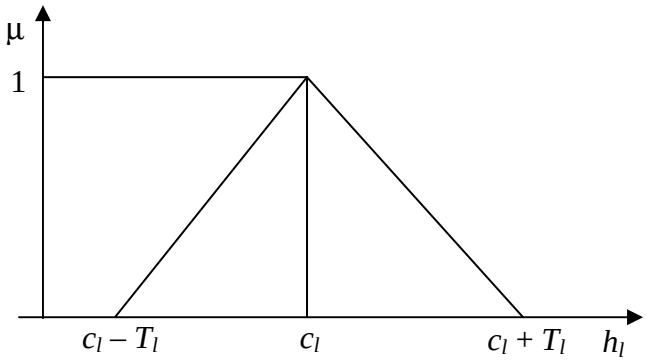

Figura 1: Função de pertinência da restrição de igualdade

As meta-heurísticas apresentam uma dificuldade em obter soluções factíveis para problemas de programação matemática com restrições de igualdade. Nesse trabalho foi definido um conjunto de restrições que permite aos algoritmos genéticos, introduzidos na Seção 3, resolverem problemas com qualquer tipo de restrição.

A Figura 1 representa a função de pertinência das restrições de igualdade com incertezas, enquanto as Figuras 2 e 3 representam as duas formas que representam as funções de pertinência das restrições de desigualdade com incertezas.

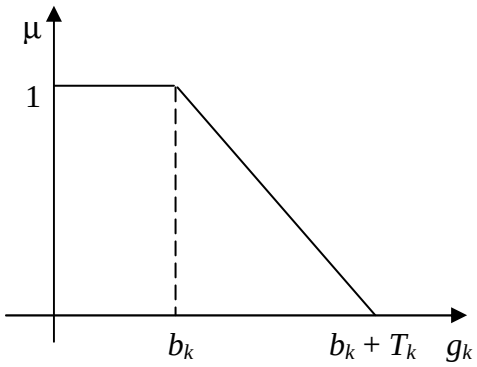

Figura 2: Função de pertinência da restrição de desigualdade decrescente.

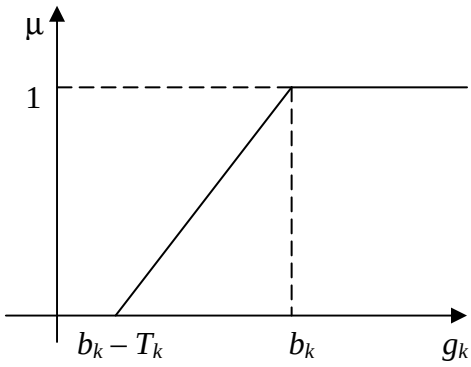

Figura 3: Função de pertinência da restrição de desigualdade crescente.

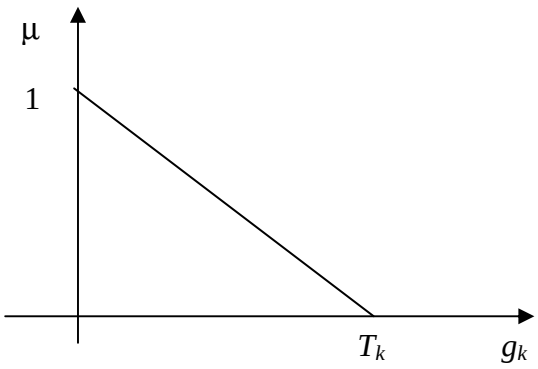

Figura 4: Função de pertinência da restrição de igualdade transformada.

Normalmente, cada restrição de igualdade é dividida em duas restrições de desigualdade, como apresentadas nas Figuras 2 e 3. Então, o esforço computacional cresce com o número de

restrições de igualdade. Cada restrição de igualdade neste trabalho é transformada em uma única restrição de desigualdade usando a função módulo. A Figura 4 ilustra a restrição transformada.

A formalização matemática desta transformação é da forma:

Seja $h(x) \cong c$, então $h(x) - c \cong 0$. Logo, usando a função módulo obtém-se $|h(x) - c| \cong 0$. Sem perda de generalidade, pode-se substituir a relação de igualdade pela de desigualdade (menor ou igual incerto) da seguinte maneira: $|h(x) - c| \lesssim 0$. Agora pode-se tratar a incerteza como segue: $|h(x) - c| \leq T(1-\alpha)$, sendo que T representa a violação máxima e α é o nível de satisfação.

Então, o Problema (1) pode ser reescrito como segue:

$$\begin{aligned} \min \quad & f(\tilde{a}; x) \\ s.a \quad & g_i(x) \lesssim b_i \quad i = 1, \ldots, m, m+1, \ldots, m+n \\ & x \in \Omega \end{aligned} \tag{2}$$

O esforço computacional permanece sem alteração porque a restrição transformada mantém as características de uma restrição de igualdade.

2.2. Valor de *fitness*

A função de *fitness* avalia a qualidade de cada indivíduo da população a cada geração e ela pode ser de várias maneiras. Entretanto, a função de *fitness* mais popular é a função objetivo do problema, que usamos neste trabalho. Em Goldberg(1989) e Michalewics(1996), pode-se encontrar formas alternativas de calcular o valor de *fitness*.

Sendo o valor de *fitness* um número nebuloso, foi aplicado o primeiro índice de Yager para encontrar o número clássico que melhor represente a distribuição do *fitness*. Em Cantão(2003), Klir e Yuan(1995), Liu e Iwamura(1998a) e Pedrics e Gomide(1998), outros métodos de defuzzificação são apresentados.

2.3. Nível de satisfação

O valor de *fitness* nebuloso fornece um conjunto de soluções com vários valores de pertinência. O valor de pertinência da solução apresenta dois pontos a serem avaliados: (i) valor de pertinência baixo com valor de *fitness* baixo, implica confiabilidade baixa; (ii) valor de pertinência alto com um *fitness* mais elevado, implica confiabilidade alta. Define-se um nível de satisfação para gerenciar estas situações, o qual pretende criar uma relação entre *fitness* e confiabilidade.

Logo, parametriza-se as restrições do Problema (2) usando os níveis de α-cut propostos na teoria nebulosa, (Zadeh (1965)), e obtemos

$$C_\alpha = \{x \mid x \in \Re^n, \mu_c(x) \geq \alpha\}, \forall \alpha \in [0,1] \tag{3}$$

sendo μ_C a função de pertinência que representa a intersecção das funções de pertinência do conjunto de restrições. Assim, pode-se re-escrever o Problema (2) como

$$\begin{aligned} \min \quad & f(\tilde{a}; x) \\ s.a \quad & g_i(x) \leq b_i + T_i(1-\alpha) \quad i = 1, \ldots, m+n \\ & x \in \Omega, \alpha \in [0,1] \end{aligned} \tag{4}$$

A solução do Problema (3) pertence ao intervalo limitado [0,1] que gera os valores inferior e superior da seguinte forma

$$\begin{aligned} \tilde{m} &= f(\tilde{a}; x^*(0)) = \min_{x \in C_0} f(\tilde{a}; x) \\ \tilde{M} &= f(\tilde{a}; x^*(1)) = \min_{x \in C_1} f(\tilde{a}; x), \end{aligned} \tag{5}$$

sendo C_0 e C_1 os níveis de corte para α igual a 0 e 1, respectivamente. Portanto, os limites inferior e superior da função objetivo são dados por

$$\begin{aligned} \mu_{\tilde{G}}^{u} &= 1 \\ \mu_{\tilde{G}}^{l} &= \mathrm{IDf}\left(\frac{\tilde{m}}{\tilde{M}}\right) \end{aligned} \tag{6}$$

sendo $\mathrm{IDf}(\cdot)$ o método de defuzzificação.

O nível mínimo de satisfação é determinado pela razão entre a solução do problema com restrições totalmente violadas e a solução com restrições totalmente satisfeitas, veja (Silva (2005)). Essa razão é um número nebuloso que é determinado por uma relação nebulosa. O método de defuzzificação escolhido foi o valor modal.

3. Algoritmos genéticos

Os algoritmos genéticos básicos descritos neste trabalho são desenvolvidos baseados em simulação nebulosa. Na sub-seção 2 introduz-se um algoritmo genético puro, enquanto na sub-seção 3 apresenta-se o mesmo algoritmo acrescido de uma busca local.

3.1. Algoritmo genético puro

O algoritmo genético puro usa somente estratégias evolucionárias básicas, isto é, processo de seleção, operação de recombinação e mutação.

3.1.1. Representação estrutural

Na definição básica de algoritmos genéticos, cada cromossomo representa uma solução factível para o problema e o tamanho do cromossomo depende da precisão desejada. Uma alternativa para representar a solução é a implementação de ponto flutuante, sendo que cada cromossomo tem o mesmo tamanho que o vetor solução. Aqui é usado o vetor $V^i = (x_1^i, x_2^i, \ldots, x_n^i)$, com $i = 1,2,\ldots, pop\_size$, para um cromossomo representar uma solução possível do problema de otimização, sendo que n é a dimensão e pop_size é o tamanho da população desse algoritmo evolutivo.

3.1.2. Processo de inicialização

Define-se um inteiro $pop\_size = 10 \cdot n$ como o número de cromossomos e inicializa os pop_size indivíduos aleatoriamente. A forma usada neste trabalho para inicializar a população é definir um ponto interior, denotado por V^0, e selecionar pop_size indivíduos aleatoriamente numa dada vizinhança desse ponto interior. Define-se um número positivo Γ que é o passo a ser caminhado na direção aleatória selecionada. Esse número Γ é um número positivo suficientemente grande. Esse número é usado no processo de inicialização e no operador de mutação. A direção $d^i \in \mathrm{IR}^n$ é escolhida de forma aleatória e define-se o indivíduo $V^i = V^0 + \Gamma \cdot d^i$. Esse indivíduo é adicionado na população inicial se ele representa uma solução factível; caso contrário, seleciona-se outro valor aleatoriamente no intervalo de $[0,\Gamma]$ até que $V^0 + \Gamma \cdot d^i$ seja factível. Repete-se este processo pop_size vezes formando uma população com pop_size soluções iniciais factíveis $V^1, V^2, \ldots, V^{pop\_size}$.

3.1.3. Função de avaliação

Uma função de avaliação, denotada por $eval(V^i)$, com $i = 1,2,\ldots, pop\_size$, é definida para determinar a probabilidade de recombinação de cada indivíduo da população, isto é, para cada cromossomo V^i. A possibilidade que um indivíduo seja selecionado é proporcional ao seu valor de *fitness* e os que possuírem os melhores *fitness* terão uma chance maior de serem selecionados a produzir filhos utilizando a operação de reprodução.

Sejam $V^1, V^2, \ldots, V^{pop\_size}$ os indivíduos da população na geração corrente. Um método interessante é aquele baseado na tentativa de reproduzir os indivíduos de acordo com a sua classificação, segundo o valor objetivo atual. Nesse processo os indivíduos são reorganizados do melhor indivíduo para o pior, de acordo com o valor de *fitness* da geração atual. Agora, seja um parâmetro $a \in (0,1)$ num sistema genético dado, então pode-se definir uma função de avaliação baseada na ordenação dos indivíduos de uma população, que pode ser escrita como:

$$eval(V^i) = a \cdot (1-a)^{i-1}, \quad i = 1,2,\ldots, pop\_size \tag{7}$$

Como mencionado acima, para $i = 1$ está armazenado o melhor indivíduo, para $i = pop\_size$ está o pior indivíduo, e

$$\sum_{i=1}^{pop\_size} eval(V^i) \approx 1.$$

3.1.4. Processo de seleção

O processo de seleção é baseado em girar a roleta *pop_size* vezes, sendo que a cada giro é selecionado um único indivíduo para uma população auxiliar da seguinte forma:

(i) Calcular a probabilidade acumulada q_i para cada indivíduo V^i da população,

$$\begin{cases} q_0 = 0 \\ q_i = \sum_{j=1}^{i} eval(V^j), \ i = 1,2,\ldots, pop\_size; \end{cases}$$

(ii) Gerar um número real aleatório r entre $[0, q_{pop\_size}]$;

(iii) Selecionar o i-ésimo indivíduo V^i $(1 \le i \le pop\_size)$, tal que, $q_{i-1} < r \le q_i$;

(iv) Repetir os itens (ii) e (iii) acima *pop_size* vezes e obtendo *pop_size* cópias dos indivíduos.

3.1.5. Operador de reprodução

Define-se um parâmetro P_r para determinar a probabilidade de recombinação. Essa probabilidade determina o número inteiro positivo par esperado de indivíduos que participam dessa operação, isto é, $\text{int}(P_r \cdot pop\_size) + \text{mod}(\text{int}(P_r \cdot pop\_size), 2)$ é o número esperado. Os indivíduos selecionados são denotados por $P^1, P^2, P^3, P^4 \ldots$, e agrupados em pares para aplicar o operador de recombinação, por exemplo, (P^1, P^2). Primeiramente, gera-se um número aleatório c no intervalo aberto (0,1), então produz-se dois filhos F^1 e F^2 usando o operador da seguinte forma:

$$\begin{aligned} F^1 &= c \cdot P^1 + (1-c) \cdot P^2 \\ F^2 &= (1-c) \cdot P^1 + c \cdot P^2 \end{aligned} \tag{8}$$

Se a região factível é um conjunto convexo, então ambos os indivíduos filhos são factíveis se os indivíduos pais também o forem. Contudo, na maioria dos problemas de programação matemática, a região factível não é necessariamente convexa ou é difícil de verificar a sua convexidade. A solução para este conflito é checar a factibilidade de cada indivíduo filho. Se ambos os indivíduos filhos forem factíveis, então os mesmos substituirão os indivíduos pais. Caso contrário, tenta-se factibilizar o indivíduo filho infactível, pois se refaz o operador de reprodução com uma nova geração de um número aleatório *c* até encontrar um cromossomo factível ou termina-se com um número pré-definido de ciclos. Neste caso, os indivíduos pai são mudados somente se os indivíduos forem factíveis.

3.1.6. Operador de Mutação

Um parâmetro P_m é definido como a probabilidade de mutação. Essa probabilidade determina o número inteiro positivo, $\text{int}(P_m \cdot pop\_size)$, esperado de indivíduos que participam dessa operação. Para cada indivíduo selecionado V^i, a operação é realizada, em primeiro lugar, escolhendo uma direção $d^i \in \text{IR}^n$ por meio aleatório. Se $V^i + \Gamma \cdot d^i$ não for factível para as restrições do problema, então o valor Γ é atualizado por um número randômico entre 0 e Γ, até que a operação anterior seja factível, sendo que Γ é um número positivo suficientemente grande definido no item 3.1.2. Caso o processo não encontre uma solução factível para valor de $\Gamma \leq 10^{-4}$, então defini-se $\Gamma = 0$.

3.1.7. Procedimento do algoritmo genético puro

Depois da seleção, reprodução e mutação, a nova população é modificada para a próxima geração. O algoritmo genético termina quando atingir um número pré-definido de repetições destes passos. Pode-se descrever cada passo deste algoritmo genético idealizado para resolver problemas de programação matemática com parâmetros *fuzzy* na função objetivo e no conjunto de restrições da seguinte forma:

Algoritmo 1: Algoritmo genético puro

Entrada: Introduzir parâmetros *pop_size*, P_r, P_m.
Saída: Retornar o melhor indivíduo como solução do problema.
PASSO 1: Inicializar *pop_size* indivíduos;
PASSO 2: Checar a factibilidade dos indivíduos e calcular os seus *fitness*;
PASSO 3: Ordenar a população, calcular a função de avaliação;
PASSO 4: Realizar o processo de seleção;
PASSO 5: Aplicar a operação de reprodução e checar a factibilidade dos filhos;
PASSO 6: Utilizar a operação de mutação e checar a factibilidade dos mutantes;
PASSO 7: Selecionar os indivíduos que passaram para a próxima geração;
PASSO 8: Calcular os *fitness* de cada indivíduo;
PASSO 9: Repetir os passos 3 a 7 por um número pré-definido de gerações;

3.2. Algoritmo genético com busca local

O algoritmo genético descrito abaixo apresenta uma estratégia chamada de "operação de busca local". A implementação de um algoritmo genético com a estratégia da operação de busca local é chamada, por alguns autores, de algoritmo memético. A operação de busca local é baseada no método de máxima descida. Ao final desta seção estão sumarizados os passos necessários no algoritmo genético com busca local desenvolvido neste trabalho.

3.2.1. Operação de busca local

Esta operação de busca local tem por finalidade realizar uma busca na vizinhança e tentar encontrar solução melhor que a atual. Assim, usa-se essa estratégia em alguns indivíduos com a intenção de evoluir a população em número de gerações menor. Alguns cuidados devem ser

tomados com a estratégia de busca local, pois quando aplicada em toda a população, gera-se dois problemas:

- esforço computacional mais elevado;
- convergência prematura para um mínimo local (principalmente).

A operação de busca local baseia-se no método de máxima descida, usado para otimizar problemas de programação matemática irrestritos. Algumas mudanças foram feitas para atender as necessidades deste algoritmo genético. A principal mudança realizada no método de máxima descida, descrito em Luiza(2003), está na retirada do critério de parada. Assim, o cálculo do passo pode ser feito de duas formas:

$$\lambda_k = \arg\min_{\lambda>0} f(\tilde{a}; x+\lambda d) + pen \cdot \sum_{i=0}^{R} \max[0, g(x+\lambda d)]$$
$$\lambda_k = \arg\min_{\lambda>0} f(\tilde{a}; x+\lambda d) - bar \cdot \sum_{i=0}^{R} \ln[-g(x+\lambda d)] \quad (9)$$

sendo que *pen* e *bar* são os fatores multiplicadores dos métodos de Penalidade e de Barreira, respectivamente.

Esta estratégia define somente uma direção de descida, pois, com o decorrer das gerações, essa direção tende a convergir para a solução ótima do problema a ser otimizado. A garantia de fornecer o melhor passo na direção definida está em avaliar a função objetivo acrescida do somatório do valor de todas as restrições infactíveis. Este procedimento foi idealizado com base nos métodos de penalidade e de barreira, porém não é resolvido nenhum destes métodos na estratégia de busca local.

3.2.2. Procedimento do algoritmo genético com busca local

Depois da seleção, reprodução, mutação e busca local, a nova população é modificada para a próxima geração. O algoritmo genético termina quando atingir um número pré-definido de repetições destes passos. Pode-se descrever cada passo deste algoritmo genético da seguinte forma:

Algoritmo 2: Algoritmo genético com busca local

Entrada: Introduzir parâmetros *pop_size*, P_r, P_m.
Saída: Retornar o melhor indivíduo como solução do problema.
PASSO 1: Inicializar *pop_size* indivíduos;
PASSO 2: Checar a factibilidade dos indivíduos e calcular os seus *fitness*;
PASSO 3: Ordenar a população, calcular a função de avaliação;
PASSO 4: Realizar o processo de seleção;
PASSO 5: Aplicar a operação de reprodução e checar a factibilidade dos filhos;
PASSO 6: Utilizar a operação de mutação e checar a factibilidade dos mutantes;
PASSO 7: Usar a operação de busca local e checar a factibilidade dos modificados;
PASSO 8: Selecionar os indivíduos que passaram para a próxima geração;
PASSO 9: Calcular os *fitness* de cada indivíduo;
PASSO 10: Repetir os passos 3 a 7 por um número pré-definido de gerações;

4. Resultados Computacionais

Aqui estão os problemas selecionados para verificar a eficiência das meta-heurísticas demonstradas na seção anterior. Esta seção está dividida em duas categorias: (a) uma formulação matemática dos problemas selecionados; e (b) os resultados computacionais, junto com uma breve análise. A análise dos resultados confronta a solução determinada por métodos clássicos com a solução encontrada pelos algoritmos genéticos que solucionam problemas com incertezas.

A implementação da metodologia apresentada na Seção 3 foi desenvolvida em C++. A máquina que usamos para simular todos os casos foi um Pentium 4, 2.53GHz, com 512Mb de memória RAM.

4.1. Formulação dos problemas

Uma formulação matemática tenta representar, através de funções, as relações existentes em um problema específico, o qual pode representar um caso real ou hipotético. As formulações apresentadas neste trabalho são casos hipotéticos. Nesta sub-seção apresentamos seis problemas de programação matemática, dos quais três pertencem ao conjunto dos problemas com restrições de desigualdade, um ao conjunto de problemas com restrição de igualdade e dois ao conjunto de problemas com restrições mistas.

4.1.1. Problemas com restrições de desigualdades

Prob.	$f(\tilde{a};x)$	Variação *fuzzy*	$x_{inicial}$	Restrições	Violação	Soluções clássicas	
						$\bar{x}^T$	$f(\bar{x}^T)$
PG1	$(x_1-\tilde{2})^2+(x_2-\tilde{1})^2$	10%	[0.5; 0.5]	$g_1(x)=x_1^2-x_2\lesssim 0$ $g_2(x)=x_2^2-x_1\lesssim 0$	T_1=0.35 T_2=0.35	[1.0; 1.0]	1.0
PG2	$x_1^2+x_2^2+\tilde{4}x_1-\tilde{4}$	10%	[1.0; 2.5]	$g_1(x)=x_2-x_1-2\lesssim 0$ $g_2(x)=x_1^2-x_2+1\lesssim 0$ $g_3(x)=-x_1\leq 0$ $g_4(x)=-x_2\leq 0$	T_1=1.0 T_2=0.5 T_3=0.0 T_4=0.0	[0.5536; 1.306]	3.79894
PG3	$\tilde{9}x_1^2+x_2^2+\tilde{9}x_3^2$	10%	[1.0; 1.0; 1.0]	$g_1(x)=1-x_1x_2\lesssim 0$ $g_2(x)=-x_2\leq 0$ $g_3(x)=-x_3\leq 0$	T_1=0.5 T_2=0.0 T_3=0.35	[0.5774; 1.732; $-0.2\,e^{-5}$]	6.0

Tabela 1: Problemas com restrições de desigualdade

Os problemas com restrições de desigualdade, descritos na Tabela 1, foram retirados de Schittkowski(1987).

4.1.2. Problemas com restrições de igualdades

Prob.	$f(\tilde{a};x)$	Variação *fuzzy*	$x_{inicial}$	Restrições	Violação	Soluções clássicas	
						$\bar{x}^T$	$f(\bar{x}^T)$
PH1	$(x_1-\tilde{2})^2+(x_2-\tilde{2}x_2)^2$	10%	[0.0; 0.0]	$h_1(x)=x_1^2-x_1\cong 0$	T_1=0.5	[0.94611; 0.89344]	1.9433

Tabela 2: Problemas com restrições de igualdade

O problema com restrições de igualdade, descritos na Tabela 2, foi retirado de Bazaraa, Sherali & Shetty(1993).

4.1.3. Problemas com restrições mistas

Prob.	$f(\tilde{a};x)$	Variação *fuzzy*	$x_{inicial}$	Restrições	Violação	Soluções clássicas	
						$\bar{x}^T$	$f(\bar{x}^T)$
PM1	$\tilde{1000} - x_1^2 - \tilde{2}x_2^2 - x_3^2 - x_1x_2 - x_1x_3$	10%	[3.0; 0.3; 4.0]	$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 \cong 0$ $h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 \cong 0$ $g_1(x) = -x_1 \leq 0$ $g_2(x) = -x_2 \leq 0$ $g_3(x) = -x_3 \leq 0$	T_1 =2.5 T_2 =5.6 T_3 =0.0 T_4 =0.0 T_5 =0.0	[3.512; 0.217; 3.552]	961.715
PM2	$(x_1 - \tilde{2})^2 + (x_2 - \tilde{1})^2$	10%	[0.0; 0.5]	$h_1(x) = x_1 - 2x_2 + 1 \cong 0$ $g_1(x) = x_1^2/4 - x_2^2 - 1 \lesssim 0$	T_1 =0.5 T_2 =0.5	[0.823; 0.911]	1.393

Tabela 3: Problemas com restrições mistas

Os problemas com restrições mistas, descritos na Tabela 3, foram retirados de Schittkowski(1987).

Nestes problemas as incertezas nos dados foram inseridas nas constantes da função objetivo com uma variação de 10% do valor modal, por exemplo, o número $\tilde{2}$ pode variar em 0.2 unidades para mais ou para menos. A solução ótima dos problemas, expostos aqui, sem inserir incertezas em seus dados estão apresentados nas colunas $\bar{x}^T$ e $f(\bar{x}^T)$.

4.2. Resultados e análises

Nesta sub-seção mostramos os resultados obtidos dos problemas apresentados na subseção acima, os quais foram solucionados pelos métodos adaptados apresentados na Seção 3. Nas Tabelas 1-3 são apresentadas as formulações dos problemas e suas soluções clássicas. Aqui são omitidos os cálculos dos níveis de satisfação; tais cálculos podem ser encontrados em Silva(2005). Mediante os resultados apresentados nessa referência pode-se calcular o nível de satisfação mínimo para cada um dos problemas de programação matemática *fuzzy* com incertezas na função objetivo e no conjunto de restrições.

A principal análise a ser feita diante de todos os resultados aqui apresentados está na escolha da relação entre o valor da função objetivo e o nível de satisfação calculado. Esta escolha depende do **decisor**, pois o conhecimento prévio do objetivo principal a ser alcançado lhe guiará a uma escolha apropriada.

Métodos	Mínimo de $f(\tilde{a};x^*)$				Tempo
	x^*	$f(\tilde{a};x^*)$	$\Im(f(\tilde{a};x^*))$	μ	
Genético Puro	[1.0608; 1.0454]	[0.53847, 0.88415, 1.3189]	0.90706	0.759117	12s
Memético	[1.0329; 1.0021]	[0.57862, 0.93551, 1.3728]	0.95563	0.717547	79s
Zimmermann adaptado	[1.0535;1.0534]	[0.71657, 0.89871, 1.1059]	0.90494	0.77643	39s
Xu adaptado	[1.058;1.0578]	[0.70907, 0.89076, 1.0974]	0.89698	0.75749	4s
Penalizado	[1.0005;1.0004]	[0.80655, 0.99895, 1.2114]	1.004	1.0000	3s

Tabela 4: Resultados para o problema PG1

A Tabela 4 mostra que o algoritmo genético com busca local foi o mais demorado, seguido pelo método adaptado de Zimmermann. A quinta linha da Tabela 4 referente-se ao método de penalização descrito em Silva, Cantão e Yamakami(2005), que apresenta o maior valor defuzzyficado da função objetivo, porém o nível de satisfação é máximo, isto é, não permite violação nas restrições, e conseguiu resolver o problema PG1 de forma mais rápida. O método adapta-

do de Xu fornece o menor valor defuzzyficado de função objetivo e tanto o nível de satisfação quanto a velocidade de processamento são admissíveis.

Métodos	Mínimo de $f(\tilde{a}; x^*)$				Tempo
	x^*	$f(\tilde{a}; x^*)$	$\Im(f(\tilde{a}; x^*))$	μ	
Genético Puro	[0.52202; 1.231]	[2.9113, 3.6988, 4.4863]	3.6988	0.958039	11s
Memético	[0.5499; 1.2931]	[2.9584, 3.7757, 4.5931]	3.7757	0.938688	14s
Zimmermann adaptado	[0.55094;1.2965]	[3.1603, 3.7807, 4.4011]	3.7807	0.93768	2s
Xu adaptado	[0.55473;1.3015]	[3.1607, 3.7826, 4.4045]	3.7826	0.9361	11s
Penalizado	[0.55177;1.3038]	[3.1765, 3.7972, 4.4179]	3.7972	1.0000	5s

Tabela 5: Resultados para o problema PG2

Na Tabela 5, pode-se constatar que os resultados do valor defuzzyficado da função objetivo e do nível de satisfação são muito parecidos entre si. A Penalização ainda continua sendo o método que prioriza estritamente o nível de satisfação, porém o valor do objetivo sofre pouca modificação entre todos.

Métodos	Mínimo de $f(\tilde{a}; x^*)$				Tempo
	x^*	$f(\tilde{a}; x^*)$	$\Im(f(\tilde{a}; x^*))$	μ	
Genético Puro	[0.5559;1.7345;0.08268]	[5.2659, 5.851, 6.4361]	5.8663	0.919717	8s
Memético	[1.0;1.0;1.0]	[17.1, 19, 20.9]	5.851	0.92212	16s
Zim. adaptado	$[0.57749;1.6907;0.9\,e^{-11}]$	[5.5598, 5.8599, 6.1601]	5.8599	0.92252	3s
Xu adaptado	$[0.5764;1.7191;\text{-}0.2\,e^{-18}]$	[5.6464, 5.9454, 6.2444]	5.9454	0.91687	5s
Penalizado	$[0.57935;1.7247;\text{-}0.2\,e^{-24}]$	[5.6932, 5.9953, 6.2974]	5.9953	1.0000	9s

Tabela 6: Resultados para o problema PG3

A Tabela 6 mostra que o nível de satisfação de todos os métodos estão acima de 90%. O tempo de processamento do genético com busca local foi o mais demorado, porém este forneceu o menor valor defuzzyficado da função objetivo. Observando os tempos, os níveis de satisfação e o valor de objetivo, Zimmermann adaptado foi o que forneceu melhores resultados.

Prob.	Mínimo de $f(\tilde{a}; x^*)$				Tempo
	x^*	$f(\tilde{a}; x^*)$	$\Im(f(\tilde{a}; x^*))$	μ	
PH1	[0.946748; 0.828629]	[0.82683, 1.7355, 3.2347]	1.8829	0.82863	12s
PM1	[3.0582;0.321026;3.99534]	[861.26, 961.28, 1061.3]	961.28	0.832225	18s
PM2	[0.782109; 0.893693]	[0.83529, 1.2714, 1.8075]	1.2963	0.90452	12s

Tabela 7: Resultado usando o algoritmo genético purto

A Tabela 7 apresenta os resultados obtidos dos problemas com restrição somente de igualdade e os com restrições de igualdade e desigualdade usando o algoritmo genético puro. Os resultados encontrados dos problemas PH1, PM1 e PM2 obtiveram um nível de satisfação acima de 80%. Os valores defuzzyficados da função objetivo de cada problema foi inferior ao apresentado em Silva(2005).

5. Conclusões

Os resultados obtidos pelos algoritmos genéticos para esses problemas hipotéticos foram bons, pois eles são melhores que os resultados encontrados na literatura. Entretanto, os resultados apresentam níveis de satisfação menores que 100%, isto é, a solução encontrada, para cada problema, viola uma ou mais restrições.

Uma grande dificuldade dos algoritmos evolutivos desenvolvidos para otimização é tratar as restrições de igualdade, mas a abordagem de usar a função módulo foi satisfatória. Uma das vantagens da função módulo é que ela transforma uma restrição de igualdade em somente uma restrição de desigualdade, enquanto que a abordagem tradicional transforma em duas restrições de desigualdade. Logo, a abordagem usada neste trabalho herda todas as características da restrição de igualdade mantendo equivalente o esforço computacional necessário. Contudo, esta abordagem não pode ser usada nos métodos clássicos porque a função módulo não é diferenciável.

Agradecimentos

Os autores agradecem a CAPES pelo suporte financeiro.

Referências

Bazaraa, M. S., Sherali, H. D. e Shetty, C. M., *Nonlinear programming – Theory and Algorithms*, 2ª ed., New York: Jonh Wiley & Sons, 1993.

Cantão, L. A. P., Programação Não-Linear com Parâmetros Fuzzy. Tese de Doutorado, FEEC – UNICAMP, Campinas, 2003.

Himmelblau, D. M., *Applied nonlinear programming.* McGraw–Hill Book Company, Reading, MA, 1972.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addson-Wesley, Reading, MA, 1989.

Klir, G. J. e Yuan, B., *Fuzzy sets and fuzzy logic: theory and applications*, Upper Saddle River, Prentice Hall, New Jersey, 1995.

Liu, B. e Iwamura, K. (1998a), Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems* **94**: 227 – 237.

Liu, B. e Iwamura, K. (1998b), A note on chance constrained programming with fuzzy coefficients, *Fuzzy Sets and Systems* **100**: 229 – 233.

Michalewics, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3ª ed., Springer, New York, 1996.

Pedrycs, W. e Gomide, F., *An Introduction of Fuzzy Sets: Analysis and Design*, A Bardford Book, 1998.

Schittkowski, K., *More test examples for nonlinear programming codes.* Spring-Verlag, 1987.

Silva, R. C., Contribuições ao estudo de programação não-linear incertezas. Dissertação de mestrado, FEEC – UNICAMP, Campinas, 2005.

Silva, R. C., Cantão, L. A. P. e Yamakami, A. (2005). Métodos iterativos para problemas de programação matemática com incertezas, *XXXVII Simpósio Brasileiro de Pesquisa Operacional*, Gramado, RS, 1937 – 1946.

Zadeh, L. A. (1965). Fuzzy sets, *Information and Control* **8**: 338 – 353.