Tendências em Matemática Aplicada e Computacional, **1**, No. 2 (2000), 525-536.
Sociedade Brasileira de Matemática Aplicada e Computacional.

Um Algoritmo Genético Associado a Métodos Determinísticos para Resolver Problemas de Otimização Não-lineares Mistos

A. de VICENTE[1], Centro de Ciências Exatas e Tecnológicas, Universidade Estadual do Oeste do Paraná (UNIOESTE), 85814-110 Cascavel, PR, Brasil

M.B. GONÇALVES[2], Departamento de Matemática, Universidade Federal de Santa Catarina (UFSC), 88010-970 Florianópolis, SC, Brasil.

Resumo. O trabalho proposto tem por objetivo apresentar um procedimento que integra Algoritmos Genéticos a métodos de otimização determinísticos, como Método Simplex, Método do Gradiente, etc., para resolver problemas de programação matemática não-lineares mistos com ou sem restrições. A metodologia proposta consiste em gerar indivíduos a partir do conjunto de variáveis binárias $Y = \{y_1, y_2, \ldots, y_n\}$ do modelo em questão, fixando seus elementos em 0 ou 1. Os modelos resultantes da substituição destes indivíduos no modelo original, os quais deixam de conter variáveis binárias, são resolvidos em seguida através de um método apropriado. O papel do Algoritmo Genético, então, é manipular uma população destes indivíduos através de cruzamentos e mutações, a fim de que, após um certo número de gerações se obtenha uma boa solução para o problema.

1. Introdução

A representação minuciosa de situações do mundo real freqüentemente conduz a modelos de programação matemática não-lineares. Em muitos casos as variáveis do problema, ou algumas delas, não admitem valores fracionários, o que faz surgir os modelos de programação matemática inteira ou mista. Estes modelos têm larga aplicação em problemas de controle de estoques, planejamento de produção, seqüenciamento, finanças [2], agricultura [9, 10] e muitas outras áreas. A resolução deste tipo de problema geralmente não é uma tarefa fácil, sendo que, em alguns casos, eles podem se tornar computacionalmente intratáveis [8]. Há muito tempo os pesquisadores vêm tentando desenvolver métodos eficazes para resolver problemas desta natureza. Pode-se citar por exemplo o trabalho de Sarathy [6], onde foi desenvolvido um algoritmo para resolver um problema de programação não-linear

[1] amarildo@unioeste.br
[2] mirian@mtm.ufsc.br

binária para alocação de dados numa rede de comunicação. A estratégia de resolução, que é específica para este problema, está baseada na linearização de suas equações, o que leva à resolução de uma série de subproblemas de programação linear. Vassilev [8] apresenta um algoritmo para obter aproximações de soluções para problemas de programação não-linear inteira. Tal algoritmo tem por princípio obter "direções viáveis inteiras". A pesquisa para encontrar tais direções é feita com base em uma aproximação linear para a função objetivo nos pontos inteiros que satisfazem às restrições do problema. Djerdjour [2] desenvolveu um algoritmo, que foi denominado Branch and Search, para resolver problemas de programação não-linear inteira genéricos que sejam linearmente restritos. Neste algoritmo, que é baseado no Branch and Bound, seleciona-se o nó mais recentemente criado para que sejam feitas as ramificações. Uma vez que um nó é gerado, pesquisa-se o mesmo completamente até que outro nó do mesmo nível seja criado.

Atualmente os estudos relativos à programação não-linear têm se concentrado bastante em métodos probabilísticos, os quais podem ser aplicados para resolver os mais diversos tipos de problemas de otimização. Pode-se citar, por exemplo, o trabalho desenvolvido por Tian [7], onde foi criado um algoritmo estocástico iterativo para resolver problemas de programação não-linear inteira. Tal algoritmo está fundamentado na reprodução de populações, tendo como base as estratégias de evolução de Darwin (auto-reprodução de boas espécies e processos de mutação que mudam a características das espécies) e as estratégias de têmpera de Boltzmann (movimento em direção ao gradiente, mudanças térmicas estocásticas para evitar mínimos locais e decréscimo de temperatura para melhorar a precisão da pesquisa).

Muitos outros trabalhos seguindo a linha de processos estocásticos podem ser encontrados. Dentre os processos existentes podem ser citados: Simulated Annealing (Liga Simulada), Neural Networks (Redes Neurais), Ant System, Tabu Search (Pesquisa Tabu), Genetic Algorithms (Algoritmos Genéticos), entre outros.

O procedimento que está sendo proposto neste trabalho tem por finalidade auxiliar na resolução de alguns problemas não-lineares e que possuem variáveis binárias, especialmente aqueles em que a não-linearidade ocorre apenas por conta das variáveis binárias. Trata-se de uma forma híbrida entre Algoritmos Genéticos com métodos determinísticos (Simplex, Newton, etc.), cuja escolha pode depender do problema em questão e do interesse do programador.

2. Algoritmos Genéticos

São algoritmos de pesquisa baseados na genética e no processo de seleção natural. Eles tornaram possível explorar um espaço mais amplo de soluções potenciais para um problema do que programas convencionais.

Muitos organismos evoluem por meio de dois processos primários: seleção natural e reprodução sexual. O primeiro determina quais membros da população devem sobreviver para reproduzir e o segundo assegura diversificação e recombinação dos genes de seus descendentes [4].

Os algoritmos genéticos são fundamentados no fato de que, na Natureza, ape-

nas os melhores indivíduos de uma espécie conseguem se adaptar ao meio em que vivem, reproduzir e formar novas gerações. Com o passar do tempo, estes indivíduos melhor adaptados tendem a predominar sobre os indivíduos mais fracos até eliminá-los. Os Algoritmos Genéticos simulam este processo de reprodução natural e, para tal, necessitam de uma representação artificial (codificação) para as criaturas. Dependendo do problema em questão, esta representação pode ser feita por meio de cadeia de caracteres (palavras), vetores, matrizes, etc. Além da representação dos indivíduos, é necessário, também, processos que imitem a troca de informações genéticas que ocorre durante o cruzamento de duas espécies, bem como a evolução ocorrida por causa de mutações. Tomando-se então uma população inicial de indivíduos, deve-se escolher alguns destes para que sejam feitos cruzamentos e mutações, a fim de que uma nova população seja obtida. É preciso lembrar, no entanto, que os indivíduos que vão gerar esta nova população devem ser aqueles melhor adaptados ao meio. A escolha destes indivíduos é feita de modo aleatório, mas de forma que os indivíduos melhor adaptados tenham mais chances de serem contemplados. Portanto, é necessário também criar um processo de seleção para tal fim.

2.1. Codificação

Dado um problema cuja solução pretende-se buscar por meio de Algoritmos Genéticos, torna-se necessário primeiramente criar as estruturas que vão representar os indivíduos que, por sua vez, vão formar a população. Estes indivíduos nada mais são que um conjunto de elementos da mesma espécie da solução do problema como, por exemplo, vetores de números reais para um problema de otimização de uma função real, caminhos de mínimo custo para a solução de um problema de roteirização, etc. A forma como os indivíduos são representados varia de problema para problema e também de acordo com a criatividade do pesquisador. Para o caso de uma população de números reais, a forma mais tradicionalmente utilizada é a representação binária. Cada indivíduo é representado por uma cadeia de caracteres, que por analogia à genética, recebe o nome de cromossomo, formada por 0 e 1 (zeros e uns). Assim, o número 9, por exemplo, seria representado pelo cromossomo 001001. O número de bits da cadeia (seis no exemplo dado) varia de acordo com a conveniência ou com o interesse do programador.

2.2. Cruzamento

Dados dois cromossomos, o cruzamento é o processo pelo qual um novo ser é gerado através destes. A idéia é que este novo indivíduo herde as características genéticas de seus genitores (que geralmente são os melhor adaptados ao meio) para constituir um indivíduo à altura ou melhor. Com repetidos processos de cruzamento espera-se que o espaço de pesquisa seja devidamente explorado, de forma que as futuras populações convirjam para a solução do problema. Existem inúmeras formas de se fazer o cruzamento entre dois cromossomos (ver [5]). Uma forma simples, para o caso da cadeia de caracteres, é escolher um ponto aleatório para os dois cromos-

somos genitores e permutar entre si os caracteres situados após o ponto escolhido. Por exemplo, cruzando-se os cromossomos 110100 e 101111 a partir da 4^a posição, resulta nos cromossomos 110111 e 101100, respectivamente.

Outro modo também simples é escolher dois pontos dos cromossomos, ao invés de um único ponto, e permutar entre ambos a parte compreendida entre tais pontos. Por exemplo, cruzando-se os cromossomos 11010010 e 01101001 entre a 2^a e a 5^a posições, resulta nos cromossomos 11101010 e 01010001, respectivamente.

Pode-se, alternativamente, escolher apenas um ou ambos os cromossomos gerados para compor a nova população.

2.3. Mutação

A mutação é um processo pelo qual um indivíduo é alterado para se adaptar a possíveis mudanças do meio em que vive. Por ser um processo não muito freqüente na natureza, geralmente ela deve ser feita poucas vezes quando uma população é gerada. Todavia, em alguns problemas o processo de mutação pode ser aplicado com freqüência, sempre que houver necessidade. Normalmente esta necessidade surge quando é gerado um indivíduo (solução) inviável para o problema e este precisa ser modificado para que se torne viável.

Para o caso dos cromossomos do exemplo anterior, uma mutação consiste apenas em mudar o valor de um carácter (bit) de 0 para 1 ou de 1 para 0. Por exemplo, fazendo-se uma mutação do cromossomo 111010 na 3^a posição, resulta no cromossomo 110010.

2.4. Seleção

A seleção é o processo pelo qual dois indivíduos são escolhidos para gerarem um novo ser. Os indivíduos são escolhidos aleatoriamente na população e, a rigor, todos têm oportunidade de serem escolhidos. Todavia, o processo deve dar privilégio àqueles melhor adaptados, qualidade dos indivíduos que é medida pelo seu *fitness*, que num problema de otimização representa o nível da função objetivo para o indivíduo considerado. Uma forma comumente utilizada para fazer a escolha é a regra da roleta. Consiste em tomar um círculo e atribuir um setor do mesmo a cada um dos indivíduos. Estes setores correspondem numericamente aos percentuais que os *fitness* dos indivíduos representam sobre o todo (soma dos *fitness* de todos os indivíduos da população). Assim, numa população de cinco indivíduos i_1, i_2, ..., i_5, com *fitness* 12, 15, 18, 30 e 45, na mesma ordem, o setor da roleta correspondente a cada um deles seria de 10.0%, 12.5%, 15.0%, 25.0% e 37.5%, respectivamente. A idéia é girar então a roleta e tomar como indivíduo sorteado aquele ao qual pertencer o setor da mesma que parar sobre um ponto previamente marcado.

Existem muitas outras maneiras de se fazer a seleção [5]. De acordo com o problema deve-se adotar aquela que for mais conveniente.

Seja f a função que representa o *fitness* (adaptabilidade) dos indivíduos de uma população. Se o objetivo do problema é procurar o máximo valor de f, então, pela regra da roleta, a probabilidade P de um indivíduo ser escolhido para reprodução

pode ser calculada por

$$P(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)},$$

onde i é um indivíduo da população e N é o tamanho desta (número de indivíduos). Para que esta expressão faça sentido, é necessário que f(i) ≥ 0, $\forall$ i, e que f(i) > 0 para algum i.

O funcionamento da roleta para a seleção pode ser simulado pelo seguinte algoritmo:

Algoritmo A
Soma := 0 e i := 0
R:= random (um número randômico)
Repita
    i:= i + 1
    Soma:= Soma + P(i)
Até que (Soma $\geq$ R) ou (i = N)

Após a execução deste algoritmo, i será o indivíduo escolhido.

De forma resumida, um Algoritmo Genético bastante simples funciona de acordo com as seguintes etapas:

Algoritmo B
Dados iniciais:
    K: Número de indivíduos da população.
    N: Número de descendentes a serem gerados em cada geração.
    M: Número máximo de gerações.
Gere uma população inicial com K indivíduos.
Calcule o *fitness* de cada indivíduo da população.
Para m:= 1 até M faça
    Para n:= 1 até N faça
        Selecione dois indivíduos da população (pais).
        Faça o cruzamento entre eles gerando um novo indivíduo e aplique a ele uma mutação se for necessário.
        Calcule o *fitness* deste novo indivíduo.
    Fim n.
    Substitua alguns ou todos os elementos da população atual pelos indivíduos gerados, formando uma nova população.
Fim m.

Os parâmetros K, N e M são tomados de forma arbitrária, e o desempenho devido a eles depende do problema em questão.

3. Processo de Resolução Proposto

Para exemplificar o funcionamento do processo, consideremos o seguinte problema:

maximizar $Z = -x_1^2 + 3x_2 - x_3^2 + x_4 + x_5 - 15y_1 - 20y_2 - 30y_3 - 12y_4 - 21y_5$
sujeito a:

$$\begin{array}{l} x_1 + 4x_2 + x_3 - 100y_1 - 20y_4 \leq 50, \\ 5x_1 + x_2 + 6x_4 - 120y_2 - 30y_5 \leq 60, \\ 6x_2 + 4x_4 + 3x_5 - 200y_3 \leq 80, \\ y_1 + y_2 \leq 1, \\ x_i \geq 0 \quad \text{e} \quad y_i \quad \text{binárias, i = 1, 2, ..., 5.} \end{array} \tag{P.1}$$

Este problema, além de possuir a função objetivo não-linear, o que traz dificuldades para a resolução, apresenta variáveis binárias em sua estrutura, aumentando ainda mais tais dificuldades. Poder-se-ia pensar então em fazer todas as combinações possíveis para as variáveis binárias y_i e resolver os modelos resultantes destas combinações por meio de um algoritmo destinado à resolução de programação não-linear. Fixando-se, por exemplo, $y_1 = 1$, $y_2 = 0$, $y_3 = 1$, $y_4 = 1$ e $y_5 = 0$, obtém-se o problema a seguir:

maximizar $Z = -x_1^2 + 3x_2 - x_3^2 + x_4 + x_5 - 57$
sujeito a:

$$\begin{array}{l} x_1 + 4x_2 + x_3 \leq 170, \\ 5x_1 + x_2 + 6x_4 \leq 60, \\ 6x_2 + 4x_4 + 3x_5 \leq 280, \\ x_i \geq 0, \text{ i = 1, 2, ..., 5.} \end{array} \tag{P.2}$$

Para o número de varáveis binárias consideradas neste problema, cinco no total, este processo levaria à resolução de 24 problemas distintos (sem a quarta restrição de (P.1) este número seria 32), o que é perfeitamente possível. Todavia, aumentando-se o número de variáveis binárias, o total de combinações cresce rapidamente. Para N destas variáveis ter-se-ia 2^N problemas distintos, menos alguns casos que devem ser excluídos em função de restrições lógicas semelhantes à quarta restrição do problema (P.1). Apesar disto, a resolução de (P.1) caso a caso por meio da fixação de variáveis, como foi feito para obter (P.2), é uma saída que merece atenção. Pela natureza deste procedimento, parece razoável utilizar um Algoritmo Genético a fim de tentar reduzir o número de combinações necessárias para obter uma boa solução.

Para o problema (P.1) isto pode ser feito seguindo-se os seguintes passos:

Algoritmo C
1 - Gere uma população de N indivíduos do tipo $Y^k = (y_1, y_2, y_3, y_4, y_5)$. Um exemplo de uma família com três indivíduos seria $Y^1 = (1, 0, 1, 0, 1)$, $Y^2 = (0, 0, 1, 0, 1)$ e $Y^3 = (1, 1, 1, 0, 1)$.
2 - Viabilize a população aplicando uma mutação para os indivíduos que não satisfizerem à quarta restrição. Por exemplo, $Y^3 = (1, 1, 1, 0, 1)$ não satisfaz a esta

restrição, devendo portanto ter sua estrutura alterada para (1, 0, 1, 0, 1) ou (0, 1, 1, 0, 1).
3 - Crie um modelo novo para cada indivíduo da população, substituindo tais i ndivíduos no modelo original.
4 - Resolva os novos modelos criados e interprete o valor da função objetivo em cada caso como sendo o *fitness* do indivíduo correspondente.
5 - Com base nestes *fitness* faça os sorteios para a escolha dos indivíduos que vão gerar a nova população. A probabilidade P de escolha de um indivíduo i pode ser calculada da seguinte forma:

$$P(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)},$$

onde i representa os indivíduos da população e Z é a função objetivo. Conhecidos os valores P(i), i = 1, 2, ..., N, faça o sorteio pela regra da roleta.
6 - Tendo sido selecionados dois indivíduos i1 e i2, faça os cruzamentos entre eles. Repita os cruzamentos até que N indivíduos tenham sido gerados.
7 - Aplique um processo de mutação aleatória, com uma probabilidade q estipulada, para auxiliar na diversificação da população.
8 - Viabilize a população resultante conforme descrito no passo 2 e repita o ciclo a partir do passo 3. O processo é encerrado quando um certo número M de gerações tiver se completado.

Para um problema de programação não-linear genérico misto o raciocínio é análogo. De um modo geral, um modelo de programação não-linear misto pode ser representado da seguinte forma:

maximizar (Minimizar) Z(X,Y)
$X \in S \subset I\!R^n$, (P.3)
$Y \in I\!R^m$, com $y_i \in \{0, 1\}$,
onde X é um vetor que contém as variáveis contínuas, S é o conjunto dos possíveis valores para X, e Y é um vetor de 0 e 1 (zeros e uns) que representa as variáveis binárias do modelo.

O papel do Algoritmo Genético é gerar populações de indivíduos, que são possíveis configurações para Y. Cada indivíduo dá origem a um novo modelo livre de variáveis binárias, os quais são resolvidos um a um. Os respectivos valores obtidos para a função objetivo representam os *fitness* destes indivíduos. Com base nestes *fitness* são realizados os processos de seleção, cruzamento e mutação, para que seja gerada uma nova população. O processo então se repete até que um critério de parada (número máximo de gerações, por exemplo) seja atendido.

Os passos a serem seguidos podem ser sumarizados pelo algoritmo a seguir.

Algoritmo D
Dados iniciais:
M: número de gerações.
K: número de indivíduos.

1. n:= 0; r:= 0; S:= 0.
2. Gerar uma população inicial viável P_m contendo k indivíduos da espécie Y.
3. Gerar um modelo novo para cada indivíduo de P_m, através da substituição destes indivíduos no modelo original.
4. Resolver os modelos criados para cada indivíduo Y^k, k = 1, 2, ..., K, e tomar o valor conferido à função objetivo em cada caso como o *fitness* de Y^k.
5. Atribuir a r o máximo destes *fitness*. Se r > S então mude S para r e guarde o indivíduo Y^* que o gerou.
6. m:= m + 1.
7. Selecionar dois a dois, pela regra da roleta, os indivíduos que vão gerar descendentes e fazer os cruzamentos e mutações para formar uma nova população P_m com K indivíduos.
8. Viabilizar P_m por meio de um processo de mutação.
9. Executar novamente os passos 3 a 5.
10. Se m = M então pare e tome S como a melhor solução, senão, forme uma nova população P_{nova} pelos K melhores indivíduos das populações P_m e P_{m-1}.
11. P_m:= P_{nova}.
12. Voltar ao passo 6.

4. Emprego do Algoritmo

Apesar de o alvo deste algoritmo ser os modelos não-lineares que possuem variáveis binárias, ele pode também ser aplicado de forma idêntica a modelos de programação linear mista e também a modelos de programação inteira.

Caso o problema em questão possua apenas variáveis binárias, então o passo 4 deste algoritmo é eliminado e o procedimento se resume puramente à aplicação do Algoritmo Genético.

4.1. Problemas de Programação Inteira

Consideremos o seguinte problema:

maximizar $Z = \sum_{j=1}^{n} c_j x_j$
sujeito a:

$$\begin{aligned} & a_j x_j \leq b, \quad j = 1, 2, ..., n, \\ & x_j = 0 \quad \text{ou} \quad 1, \quad a_j \geq 0, \quad b \geq 0 \quad \text{e} \quad c_j \geq 0. \end{aligned} \tag{P.4}$$

Este problema, conhecido como o problema da mochila, possui apenas variáveis binárias. Ao se tomar um indivíduo viável qualquer, obtém-se diretamente o valor correspondente da função objetivo Z, dispensando portanto a execução do passo 4 do algoritmo apresentado anteriormente (Algoritmo D). Além disso, se o problema apresentar variáveis inteiras (não puramente binárias), pode-se representá-las como uma combinação linear de variáveis binárias e utilizar o mesmo algoritmo para

resolvê-lo. Por exemplo, se x é uma variável inteira tal que $0 \leq$ x $\leq$ m , então esta representação requer a criação de k variáveis binárias, onde k é escolhido de modo que $2^k - 1 \leq m < 2^k$. Daí, a variável inteira x pode ser representada por:

$$x = y_1 + 2y_2 + 4y_3 + \ldots + 2^{k-1}y_k,$$

onde cada y_j é uma variável binária.

4.2. Problemas com Não-Linearidades apenas nas Variáveis Binárias

Consideramos o seguinte problema:

maximizar $Z = \sum_{j=1}^{n} y_j(x_j - k_j)$
sujeito a:
$\sum_{j=1}^{n} a_j x_j \leq b,$ (P.5)
$\sum_{j=1}^{n} y_j \leq m,$
$x_j \geq 0$, $y_j = 0$ ou 1, $m \in$ N, a_j, k_j *e* b constantes arbitrárias positivas.

Este é um tipo de problema em que as não-lineraridades são devidas exclusivamente às variáveis binárias, razão pela qual o modelo se torna linear quando tais variáveis são eliminadas.

Observemos ainda o modelo a seguir:

maximizar $Z = \sum_{j=1}^{n}(x_j - y_j m_j + w_j k_j)$
sujeito a:
$x_j \leq y_j m_j, \quad j = 1, 2, \ldots, n,$ (P.6)
$w_j = \begin{cases} 1, & \text{se } \sum_{t=1}^{j} y_t = 1, \quad j = 1, \ldots, n, \\ 0, & \text{nos demais casos} \end{cases}$
$x_j \geq 0$, $y_j = 0$ ou 1, $w_j = 0$ ou 1, sendo k_j e m_j constantes arbitrárias positivas.

Este problema também é não-linear, mas o que caracteriza tal fato é a descontinuidade das funções que definem w_j. Da mesma forma que no problema prévio, se as variáveis binárias forem eliminadas então o problema passa a ser linear.

Muito embora o algoritmo proposto tenha um caráter geral com relação à natureza das variáveis contínuas do problema (lineares ou não), deve-se salientar que, para os casos em que os subproblemas gerados pela eliminação das variáveis binárias são não-lineares, não há garantia da obtenção de uma solução, já que a resolução destes é feita por métodos convencionais (passo 4 do Algoritmo D). Na verdade a essência deste procedimento são os problemas de programação não-lineares mistos como os desta seção, nos quais a não-linearidade é devida apenas a variáveis binárias, ou que possuem funções descontínuas envolvendo tais variáveis.

5. Testes Realizados

No que diz respeito ao tempo de processamento, é necessário frisar que o objetivo do algoritmo proposto não é competir com outros algoritmos tradicionais na redução do tempo de processamento, mas sim, possibilitar a obtenção de uma boa solução para problemas que, pelo fato de possuírem variáveis binárias e apresentarem não-linearidades em sua estrutura, oferecem dificuldades na resolução. Desta forma, a avaliação de desempenho deve ficar por conta da eficiência em se obter uma boa solução e não pelo tempo de processamento.

Em relação à convergência ou não para a solução ótima, pode-se dizer que vai depender do modelo em questão. Tendo em vista que o processo está vinculado a um método determinístico destinado a resolver os problemas resultantes da escolha de indivíduos arbitrários, então é claro que o desempenho do sistema depende do sucesso ou fracasso deste método. Em função disto, pode-se afirmar que o procedimento proposto tem sucesso garantido apenas para os modelos em que ocorrem não-linearidades somente por conta das variáveis binarias, já que os modelos resultantes após a eliminação destas são lineares. Ao contrário, se o modelo original apresentar não-linearidade nas variáveis contínuas, então o modelo resultante da escolha de um indivíduo qualquer será não-linear, o que quer dizer que o sistema só poderá alcançar a solução ótima do modelo original se o método determinístico empregado for capaz de resolver os subproblemas gerados após a tomada dos indivíduos da população.

Para testar o desempenho do sistema foram tomados diversos problemas, dentre os quais o problema (P.7) dado a seguir.

maximizar $Z = \sum_{i=1}^{n}(c_i x_i - m_i y_i - k_i w_i)$
sujeito a:

$$x_i \leq b_i y_i, \quad i = 1, 2, \ldots, n, \tag{P.7}$$

$$w_i = \begin{cases} 1, & \text{se} \quad \sum_{j=1}^{i} 2y_j - y_i = 1, \quad i = 1, 2 \ldots, n, \\ 0, & \text{em caso contrário} \end{cases}$$

$x_i \geq 0$, y_i e $w_i \in \{0, 1\}$, n $\in \mathbb{N}$, e k_i, m_i b_i e c_i constantes arbitrárias positivas.

Este modelo é parte integrante de um modelo maior aplicado à agricultura, desenvolvido pelos autores (ver [9, 10]). Trata-se de um problema onde se observam funções descontínuas envolvendo apenas variáveis binárias. Por este motivo, todos os subproblemas gerados são lineares. A baixa complexidade deste modelo não interfere no funcionamento do Algoritmo Genético, tendo em vista que seu papel é somente selecionar os indivíduos de acordo com o *fitness* de cada um. Na verdade a complexidade só interfere no tempo de resolução dos subproblemas, que é consumido principalmente pela execução do método empregado para resolvê-los (o método Simplex neste exemplo). Para medir alguns fatores relativos a eficiência do método, foram considerados diferentes dimensões (valores de n) para o problema (P.7) e foram feitas 10 execuções do algoritmo para cada caso. Para isto, considerou-se uma população de 16 indivíduos e um número máximo de 40 gerações

por tentativa. A população inicial em cada execução foi gerada por meio do gerador pseudo-aleatório da linguagem Pascal. O processamento foi feito num microcomputador Pentium 266 Mhz, sendo que os passos relativos ao Algoritmo Genético ficou a cargo de um programa feito na linguagem Pascal e a execução do método Simplex foi feita pelo aplicativo GAMS. Os resultados obtidos estão no Quadro 1 a seguir.

Quadro 1 - Resultados obtidos da resolução do problema (P.7) para diferentes valores de n

n	Variáveis binárias	Variáveis contínuas	Restrições	Solução ótima Z^*	Freqüência de Z^* em 10 execuções	Tempo médio por execução (min.)
05	10	05	20	35	10	0,18
07	14	07	28	50	10	0,22
10	20	10	40	74	10	0,62
12	24	12	48	74	10	0,70
15	30	15	60	74	10	0,91
17	34	17	68	81	10	1,34
24	40	20	80	96	10	1,52
22	44	22	88	111	9	2,05
25	50	25	100	117	9	2,59

Como se pode observar, apenas em dois casos a solução ótima não foi encontrada, uma para n = 22 e outra para n = 25. Isto mostra que o algoritmo possui um bom desempenho no que diz respeito à obtenção da solução.

Além de (P.7), diversos problemas foram experimentados, dentre os quais o problema (P.1) apresentado na Seção 3, o qual possui cinco variáveis contínuas, cinco variáveis binárias e quatro restrições. Para aquele modelo foi obtido o valor ótimo Z* = 85,333 em todas as 10 tentativas realizadas. O método determinístico empregado foi o Método dos Gradientes Reduzidos e o tempo médio por tentativa foi de 10,2 segundos.

6. Conclusões

O procedimento proposto mostrou ser uma boa ferramenta para resolver problemas de programação não-lineares mistos. Em testes realizados para diferentes configurações de um modelo específico, no qual as variáveis contínuas são lineares, chegou-se à solução ótima em pelo menos 90% de um total de 10 tentativas feitas para cada configuração.

Para modelos em que há variáveis contínuas não-lineares também se observou um bom desempenho do algoritmo. Todavia, em virtude das dificuldades inerentes à programação não-linear, sua eficiência está vinculada à performance do método determinístico empregado na resolução dos subproblemas gerados durante o processamento.

Apesar de o alvo do algoritmo apresentado ser modelos não-lineares mistos, ele

pode também ser empregado na resolução de problemas lineares mistos. Isto inclui também os problemas que possuem apenas variáveis binárias.

Referências

[1] P.S. Bradley, C.A. Hax e T.L. Magnanti, "Applied Mathematical Programming", Addison-Wesley Publishing Company, 1977.

[2] M. Djerdjour, An Enumerative Algorithm Framework for a Class of Nonlinear Integer Programming Problems, *European Journal of Operational Research*, **101** (1997), 104-121.

[3] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989.

[4] J.H. Holland, Genetic Algorithms, *Scientific American*, **July** (1992), 45-50.

[5] M. Mitchel, "An Introduction to Genetic Algorithm", Bradford Book, 1996.

[6] R. Sarathy, B. Shetty and A. Sen, A Constrained Nonlinear 0-1 Program for Data Allocation, *European Journal of Operational Research*, **102** (1997), 626-647.

[7] P. Tian, J. Ma and D. Zhang, Non-linear Integer Programming by Darwin and Boltzmann Mixed Strategy, *European Journal of Operational Research*, **105** (1998), 224-235.

[8] V. Vassilev and K. Genova, An Approximate Algorithm for Nonlinear Integer Programming, *European Journal of Operational Research*, **74** (1994), 170-178.

[9] A. de Vicente, Um modelo Matemático para a Integração das Atividades de um Sistema de Produção Agrícola. "Tese de Doutorado - Universidade Federal de Santa Catarina", (1999), 118p.

[10] A. de Vicente e M.B. Gonçalves, Modelagem Matemática para a Integração das Atividades de um Sistema de Produccão Agrícola, *Anais do XXXI Simpósio Brasileiro de Pesquisa Operacional da Sociedade Brasileira de Pesquisa Operacional*, (1999), 340-354.