ISSN 2175-6295 RIO DE JANEIRO- BRASIL, 08 E 09 NOVEMBRO DE 2007.
SPOLM 2007

EXTRAÇÃO DE ENERGIA EM UM REATOR NUCLEAR UTILIZANDO *SIMULATED ANNEALING* E ALGORITMOS GENÉTICOS

Rodolfo Ranck Junior
Instituto Nacional de Pesquisas Espaciais – (INPE)
Caixa Postal 15.064 – 91.501-970 – São José dos Campos – SP – Brazil
rodolforanck@yahoo.com.br

José Carlos Becceneri
Laboratório Associado de Computação e Matemática Aplicada Instituto – Nacional de Pesquisas Espaciais – (INPE)
Caixa Postal 15.064 – 91.501-970 – São José dos Campos – SP – Brazil
becce@lac.inpe.br

José Demísio Simões da Silva
Laboratório Associado de Computação e Matemática Aplicada Instituto – Nacional de Pesquisas Espaciais – (INPE)
Caixa Postal 15.064 – 91.501-970 – São José dos Campos – SP – Brazil
demisio@lac.inpe.br

Resumo. A extração de energia em um reator nuclear é um problema complexo de natureza combinatória, no qual é preciso estabelecer uma política de extração para cada uma das n seções de um trocador de calor. Neste trabalho procura-se por uma política de extração que incorra no menor custo possível utilizando as metas-heurísticas *Simulated Annealing* (SA) e Algoritmos Genéticos (AG). O objetivo é obter uma solução inicial através da meta-heurística SA e tentar melhorá-la com meta-heurística AG. A modelagem matemática do problema é apresentada e diversos testes são realizados com o objetivo de verificar a aplicabilidade do modelo heurístico proposto.

Palavras-chave. Algoritmos Genéticos, Problema de Extração, Trocador de Calor, Normalização de Cromossomo.

Abstract. The extraction of energy in a nuclear reactor is a complex problem of combinatorial nature, in which it is necessary to establish one extraction politic for each n sections of a heat exchanger. In this work it is looked for one extraction politic with the lesser possible cost using the heuristics Simulated Annealing (SA) and Genetic Algorithms (GA). The objective is to get an initial solution through heuristic SA and try to improve it with heuristic GA. The mathematical modeling of the problem is presented and diverse tests are made in reason to verify the applicability of the considered heuristic model.

Keywords. Genetic Algorithms, Extraction Problem, Heat Exchanger, Chromosome Normalization.

1. Introdução

Definir as quantidades a serem extraídas de cada seção de um trocador de calor é um problema computacionalmente complexo. Existe uma infinidade de maneiras de se extrair a energia ao longo dessas seções. Uma solução exata para este problema torna-se rapidamente inviável quando a complexidade da função objetivo é aumentada. Assim, métodos inteligentes de busca são estudados para se encontrar uma boa solução para este problema.

Inicialmente o *Simulated Annealing* (SA) é utilizado para se encontrar uma boa solução. Encontrada a solução, torna-se interessante vasculhar a vizinhança deste ponto, em busca de uma ainda melhor. Para isso é proposto uma aplicação da Computação Evolutiva através da meta-heurística Algoritmos Genéticos (AG). O objetivo é fazer com que o AG utilize a solução obtida no algoritmo SA proposto em Becenneri (2002), para gerar um conjunto inicial de soluções, e que a partir daí, evolua para uma solução melhor.

2. Fundamentação Teórica

2.1. O *Simulated Annealing*

O *Simulated Annealing* é uma meta-heurística utilizada como técnica de otimização. Seu nome, e também seu funcionamento, formam uma metáfora entre problemas de otimização com um processo de recozimento. Tal processo é muito utilizado em metalurgias para se encontrar estados de alta estabilidade (baixa energia). O procedimento consiste em submeter certos materiais a altas temperaturas e reduzi-las gradualmente. Durante esse período, o material passa por reduções e aumentos no seu estado de energia até que seja atingido um equilíbrio térmico. [Viana, 1998].

Na analogia proposta por [Metropolis et al 1953], o recozimento simulado trata a função objetivo a ser minimizada como a energia dos estados do sólido. Durante o processo de otimização as soluções são geradas aleatoriamente e o algoritmo aceita todas as soluções melhores que a atual. Quando a solução é pior, o algoritmo pode ou não aceitá-la de acordo com a função de probabilidade definida por: $e^{\frac{-\Delta E}{T}}$, onde: ΔE é a variação da solução atual para a melhor encontrada até o momento; T é a temperatura atual. [Viana, 1998].

2.2. Computação Evolutiva

O modelo Evolutivo da Computação é derivado da Teoria da Seleção Natural proposta inicialmente por Darwin. Tal modelo é freqüentemente utilizado na otimização de problemas combinatórios como uma meta-heurística. [Viana, 1998].

As observações de Darwin em 1838 incluíam registros de diversas espécies com características ideais para a sobrevivência no seu *habitat* natural. Tais observações possibilitaram Darwin formular a Teoria da Seleção Natural, onde as espécies se adaptavam ao meio que vivem, através de um processo evolutivo natural. Aqueles indivíduos com características que permitissem uma vida mais longa e favorável possuiriam maior probabilidade de procriar, e levar seus genes adiante. [Viana, 1998].

Algumas décadas se passaram até que a tecnologia propiciasse ferramentas que pudessem ser usadas para entender o mecanismo genético. A descoberta dos cromossomos foi a chave para o entendimento do processo natural evolutivo proposto por Darwin. [Griffiths et al, 2002].

Sabe-se hoje que os cromossomos são formados por uma seqüência de ácidos nucléicos que determinam características (fenótipos) de um indivíduo. Tais seqüências formam a unidade mínima da hereditariedade e são chamadas de genes [Griffiths et al, 2002].

Dois outros mecanismos completam o conhecimento fundamental do processo evolutivo: A recombinação de genes (*crossing over*), mecanismo onde acontece a troca de material genético (genes) entre cromossomos e o processo de mutação, mecanismo que garante variedade aos indivíduos em uma população. [Viana, 1998].

2.3. Algoritmos Genéticos (AG)

Criados por Holland (1975), os Algoritmos Genéticos pertencem a classe das Metas-heurísticas e por isso, tem como principal característica, a de serem destinados a encontrar uma boa solução, ao invés de serem destinados à busca pela solução ótima.

Um Algoritmo Genético é uma técnica de busca computacional utilizada para se obter soluções boas, eventualmente uma solução ótima, para um problema. Formam uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação (*crossover*). São utilizados freqüentemente para otimizar funções oriundas de problemas combinatórios. [Goldberg,1989]

Os Algoritmos Genéticos são implementados como uma simulação computacional na qual indivíduos de uma população (conjunto de soluções) trocam seus genes com o objetivo de formar soluções melhores a cada iteração. [Goldberg,1989]

3. O problema da extração de energia de um reator nuclear

Em um reator de uma usina termonuclear, um fluido é aquecido sob alta pressão e passa do circuito primário para um trocador de calor. No trocador de calor esse fluido aquece a água do circuito secundário que então, vira vapor. O vapor, por fim, vai movimentar uma turbina que acionará um gerador elétrico.

O modo que o calor é absorvido do trocador de calor pelo circuito secundário, isto é, a maneira na qual as quantidades extraídas de calor são distribuídas, irá definir um percentual importante dos custos do processo. É interessante que o calor total seja retirado de modo a gerar a menor quantidade de custos inerentes.

Procura-se então, por uma política de extração de calor que minimize os custos do processo. Neste âmbito, uma política de extração é a quantidade de energia a ser retirada por cada uma das n seção do trocador de calor.

O Problema de Extração apresentado é um problema combinatório e foi estudado inicialmente em [Becceneri e Zinober, 2001]. O uso de um método inteligente para se encontrar a solução torna-se necessário.

3.1. Modelagem Matemática do Problema

É preciso extrair do problema as variáveis a serem consideradas. O objetivo é minimizar o custo das quantidades de calor que fluem pelas seções, ou seja, a quantidade de calor que sai da seção (x_i) e a quantidade de calor transportada de uma seção para outra (g_i).

Deste modo, a troca de calor no sistema pode ser feita através de *n* seções onde o fluído é removido e o calor é retirado. Na Figura 1, verifica-se essa representação:

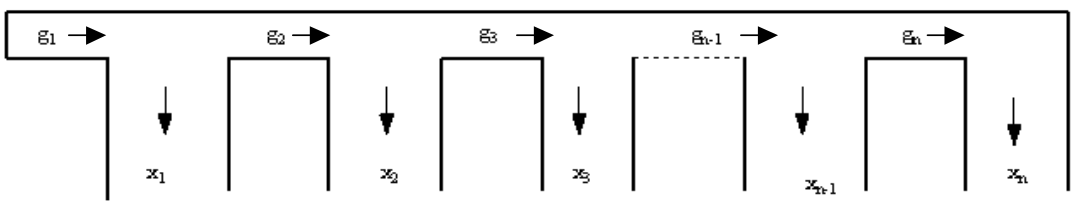

Figura 1. Esquema de um trocador de calor de um reator nuclear

Onde na representação;

$\mathbf{g_i}$ é a quantidade de calor disponível para ser retirada da seção *i* e $\mathbf{x_i}$ é a quantidade de calor extraída da seção *i;*

Dado que o objetivo é minimizar as quantidades, g_i e x_i representadas na Figura 1 observam-se a interação dessas variáveis no sistema. Trecho baseado na modelagem realizada em [Becceneri e Zinober, 2001]:

I-) A quantidade g_1 é a quantidade de calor inicial que entra no sistema. Este fluxo de calor inicial é igual a um para que haja uma relação direta com a representação percentual. As quantidades $g_2, g_3, ..., g_n$ são as quantidades restantes após a retirada em suas respectivas seções anteriores.

II-) As quantidades $x_1, x_2, ..., x_n$ representam todo calor que deixa o sistema. Considera-se que todo fluxo de calor é retirado do sistema.

III-) A quantidade g_i será um limitante superior para $x_i \ \forall i = 1,2,3,4$. Ou seja, não se pode tirar mais calor do que é disponível. Já o limitante inferior para $x_i \ \forall i = 1,2,3,4$... n será zero desde que respeite a condição estabelecida em (1) e (2) .

$\sum_{i=1}^{n} x_i = 1$ (1).

ou ainda; $\sum_{i=1}^{n} x_i = g_1$ (2).

IV-) A quantidade de fluxo de calor que passa pelo sistema entre uma e outra seção é a diferença entre a quantidade de fluxo que passou e a que saiu na seção anterior, ou seja;

$g_{i+1} = g_i - x_i$ (3)

Por fim, tendo x_n como a ultima seção do sistema, tem-se: $x_n = g_n$ (4).

V-) O custo utilizado como exemplo nesse trabalho será dado pela função:

$c(g_i, x_i) = g_i^3 + \lambda^2 x_i^3, \lambda > 0.$ (5)

A função representada na equação (5) é utilizada para verificar os resultados da meta-heurística apresentada. A partir de uma aplicação de sucesso utilizando esta função custo, poder-se-á validar a aplicação da meta-heurística proposta em funções custo mais complexas.

Agora, a partir da função (5) apresentada, é possível encontrar as quantidades x_i que minimizam a função objetivo:

$f_i(g_i)$ = min{ custo de remover x_i da seção i + custo mínimo de remover g_{i+1} das demais seções}

Verifica-se que o modo que o calor passa pelo sistema é dependente da quantidade que entra na seção i, que depende do fluxo da seção i-1, que por sua vez, depende da diferença de entrada e saída de calor anterior da seção i-2 e assim por diante. Tendo em vista essa característica do sistema, é interessante representá-lo através de uma relação de recorrência. Tal relação é apresentada a seguir;

$f_i(g_i) = \min\{ c(g_i, x_i) + f_{i+1}(g_i - x_i) \}, 0 \leq x_i \leq g_i, i = n-1, n-2, \ldots, 3, 2, 1.$ (6)

A solução discreta para este problema é dada a seguir.

Seja G a quantidade de calor inicial. Dividimos G em *p* partes iguais: $G = p\Delta g$, sendo:

$g = q\Delta g$, a quantidade que entra e $x = r\Delta g$, a quantidade recebida. Onde *q* and *r* são inteiros.

Considera-se: $Fi(g) = f_i(g)/G^3 = f_i(g)/p^3(\Delta g)^3$. Então:

$f_i(g) = \min\{g^3 + \lambda^2 x^3 + f_{i+1}(g-x)\}, 0 \le x \le g$

$F_i(q) = \min\{(q/p)^3 + \lambda^2(r/p)^3 + F_{i+1}(q-r)\}, 0 \le r \le q$ (7)

Sabe-se que $f_n = g_n^3$ logo:

$F_n(q) = (q/p)^3$, para cada q

Usa-se a relação recursiva (6) para $F_i(q)$, $i = n-1, n-2,..., 3, 2, 1$,

A solução analítica para este problema é mostrada em Zinober (2001).

É importante ressaltar que o método analítico não funcionará com facilidade para outras funções custo. A função de custo (5) utilizada nesse trabalho tem por finalidade validar o uso da meta-heurística proposta para solucionar desse tipo de problema.

4. O Algoritmo Genético na melhora da solução do problema

Aqui são descritos os princípios básicos da meta-heurística AG, implementada para melhorar as soluções geradas pela meta-heurística SA proposta em Becceneri (2002).

4.1. Cromossomo.

Dado que a solução buscada é a quantidade de calor extraído x_i em cada uma das n saídas, defini-se cromossomo como um conjunto formado por $x_1, x_2, x_3, ..., x_n$ e, portanto, os genes serão representados por cada quantidade individualmente retirada x_i. Visualiza-se essa representação na Figura 2.

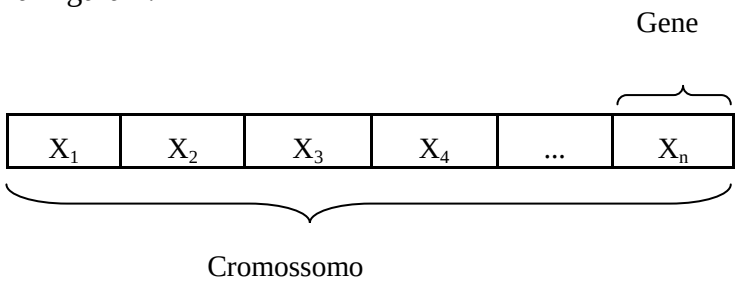

Figura 2. Representação de um cromossomo para o problema

Com essa estrutura, um exemplo de cromossomo para o problema com $i = 1,2,3,4$ seria: $x_1 = 0,2$; $x_2 = 0,5$; $x_3 = 0,1$; $x_4 = 0,2$. Observar que tal solução satisfaz (1) e, por conseguinte, é um cromossomo válido.

4.2. *Crossover*.

O *crossover* no modelo é a troca do material genético entre dois cromossomos, ou seja, entre duas soluções. O intuito deste cruzamento é gerar novas soluções baseadas nas melhores obtidas.

Para um trocador de calor com n sessões, o ponto de cruzamento, ponto que define os genes que irão ser cedidos para formar um novo cromossomo, é gerado aleatoriamente entre qualquer um dos P_i pontos possíveis de corte, onde i = 1,2,3,...,*n-1*. Os pontos possíveis de corte são aqueles que permitem separar um cromossomo em dois, assim como é apresentado no exemplo da Figura 3.

Considerando, por exemplo, que o problema a ser resolvido é de um trocador de calor com 4 seções, um possível *crossover* poderia acontecer no ponto P_2. Tal esquema pode ser visualizado a seguir na Figura 3:

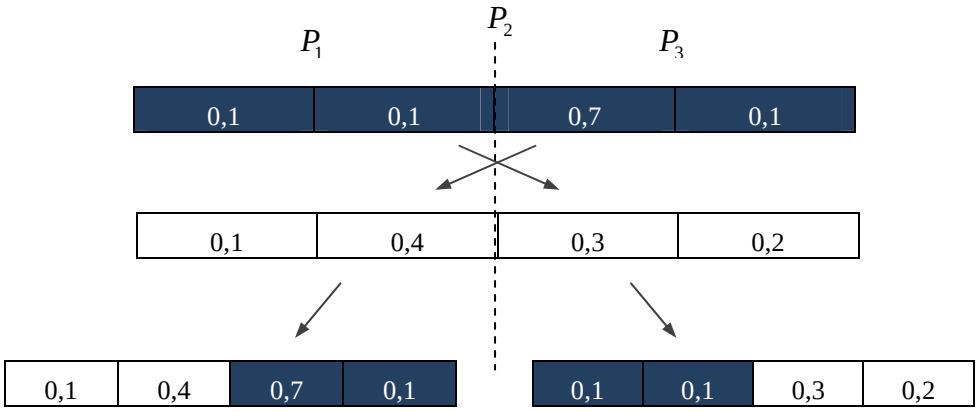

Figura 3. Representação de um *crossover* padrão realizado pelo algoritmo.

Porém, observar na Figura 4, que com esse procedimento, os novos filhos gerados, não são soluções válidas, pois não satisfazem (1):

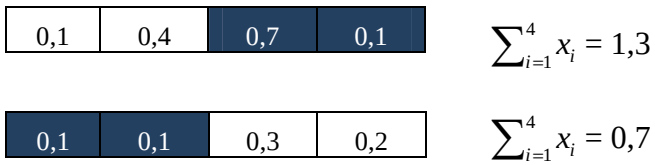

Figura 4. Representação de dois cromossomos não normalizados (resultantes de um cruzamento).

A solução é ajustar esses cromossomos para que satisfaçam a condição fundamental. Entretanto, tal ajuste, pode causar perda de informações passadas pelos pais, já que a codificação genética terá que ser modificada. Sem essas informações o cruzamento pode estar gerando filhos com características diferentes dos pais, fazendo com que a solução não evolua como desejado.

A fim de evitar tal problema, foram desenvolvidas duas políticas para normalizar os cromossomos:

A-) Política I

Será aplicada quando o corte for realizado em qualquer ponto com exceção do primeiro.

A solução é retirar o excesso ou completar o faltante de cada um dos vetores-solução gerados, para que satisfaça a condição proposta em (1).

O critério escolhido foi de manter inalterados os genes que sucedem o ponto de corte, e distribuir a diferença numérica por aqueles que antecedem o ponto de corte. A distribuição é proporcional ao valor de cada gene em questão.

O novo valor para os genes normalizados (x_j) pela política 1, para um trocador de calor com n seções, é dado pela fórmula a seguir:

$$x_j = \left[\left(1 - \sum_{k=1}^{n} x_k\right) \cdot \frac{x_j}{\sum_{k=1}^{j} x_k} \right] + x_j \quad \forall \le P_i \text{ onde } \{P_i = i \mid i = 1,2,3,...,n-1\} \qquad (8)$$

B-) Política II

Será utilizada quando o corte é realizado no primeiro ponto.

Neste caso, utilizando a política 1 de normalização, os cromossomos filhos gerados seriam cópias idênticas aos pais. Tal acontecimento não é interessante, visto a importância que as variedades de cromossomos proporcionam à busca. Para contornar este problema foi criada uma segunda política a ser utilizada pelo algoritmo apenas nessa ocasião.

A normalização é idêntica àquela aplicada pela política 1, entretanto, é realizada em todos os genes, fazendo com que o resíduo (diferença para satisfazer $\sum_{i=1}^{n} x_i = 1$) seja distribuído proporcionalmente por todo cromossomo.

O novo valor para os genes normalizados (x_j) pela política 2, para um trocador de calor com *n* seções, é dado pela fórmula a seguir:

$$x_j = \left[\left(1 - \sum_{k=1}^{n} x_k\right) \cdot \frac{x_j}{\sum_{k=1}^{n} x_k} \right] + x_j \quad \forall\, j \le P_i \qquad (9)$$

Essas políticas foram as que propiciaram os melhores resultados entre todas as outras avaliadas.

O processo de normalização do cromossomo é avaliado através do exemplo que segue (Figuras 5, 6, 7 e 8):

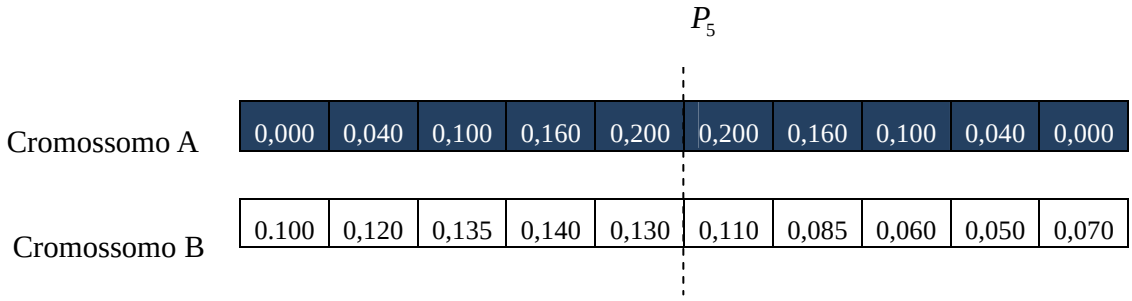

Figura 5. Cromossomos A e B utilizados em um exemplo de normalização.

Abstraindo o procedimento básico de recombinação genética apresentado na Figura 3. Tem-se, a partir da política de normalização implementada, os cromossomos apresentados na Figura 6:

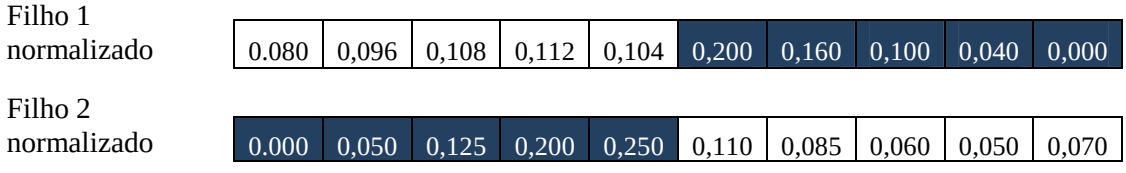

Figura 6. Filhos dos cromossomos A e B normalizados.

Nas Figuras 7 e 8 visualizam-se a distribuição do material genético dos cromossomos A e B e dos Filhos normalizados 1 e 2.

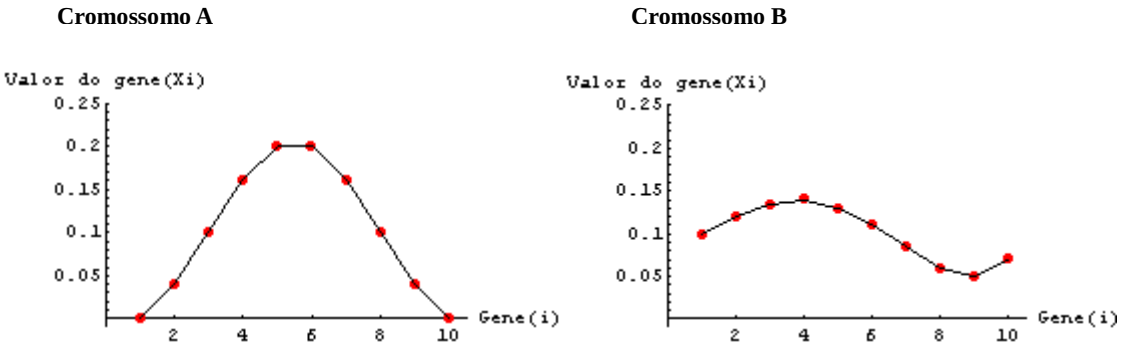

Figura 7. Curva de distribuição genética dos cromossomos A e B.

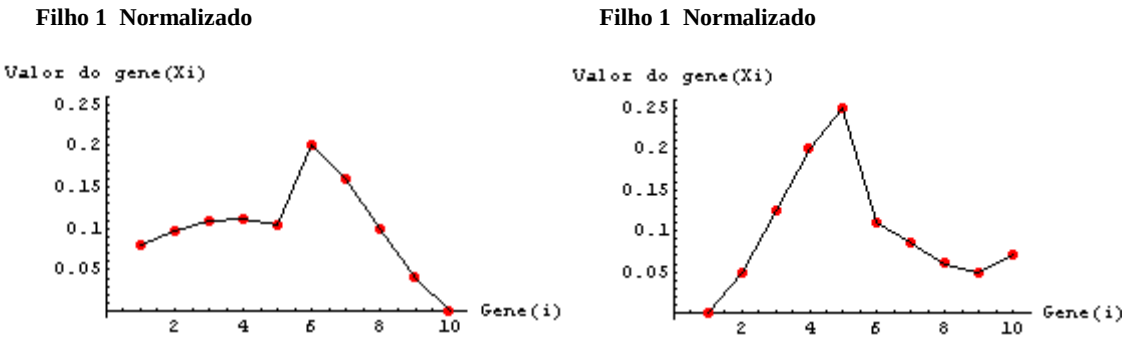

Figura 8. Curva de distribuição genética dos filhos dos cromossomos A e B normalizados.

Observar na Figura 8, que através dessa política, os cromossomos filhos mantêm algumas características de cada um dos pais mesmo após a normalização. Para cada cromossomo filho gerado, observa-se um pedaço do cromossomo mantido inalterado proveniente de um dos pais mais um pedaço proveniente do outro pai, porém normalizado.

Caso esses cromossomos representem uma boa solução e cheguem à fase de cruzamento, estes irão querer passar aos seus descendentes, suas características que os tornam boas soluções, ou seja, que minimizam a função objetivo. Deste modo, é importante que no processo de normalização dos cromossomos filhos algumas características originais dos pais sejam mantidas. Esse foi o motivo pelo qual essa normalização foi adotada.

4.3. Seleção

A função de seleção ordena todas as soluções de uma população de acordo com seu *fitness*. Após a ordenação uma determinada percentagem, previamente definida no algoritmo, é eliminada de modo a permitir apenas o cruzamento apenas entre as melhores soluções. O processo de Seleção e o processo de Elitismo são fundamentais para que as soluções evoluam corretamente no AG.

4.4. Mutação

As políticas de normalização do cromossomo (8) e (9) propostas em 4.2, ao mesmo tempo em que mantém características das gerações, produzem uma pequena modificação nas

informações genéticas originais. Tal modificação pode ser considerada como uma pequena mutação que é realizada suavemente e que ocorre naturalmente no processo sucessivo de permutação genética (*crossover*). Esse processo garante a variabilidade das soluções, por gerar características diferentes das originalmente trazidas pelas populações anteriores. Tal variedade é fundamental para evitar pontos de mínimos locais.

4.6. Esquema de funcionamento do modelo algorítmico.

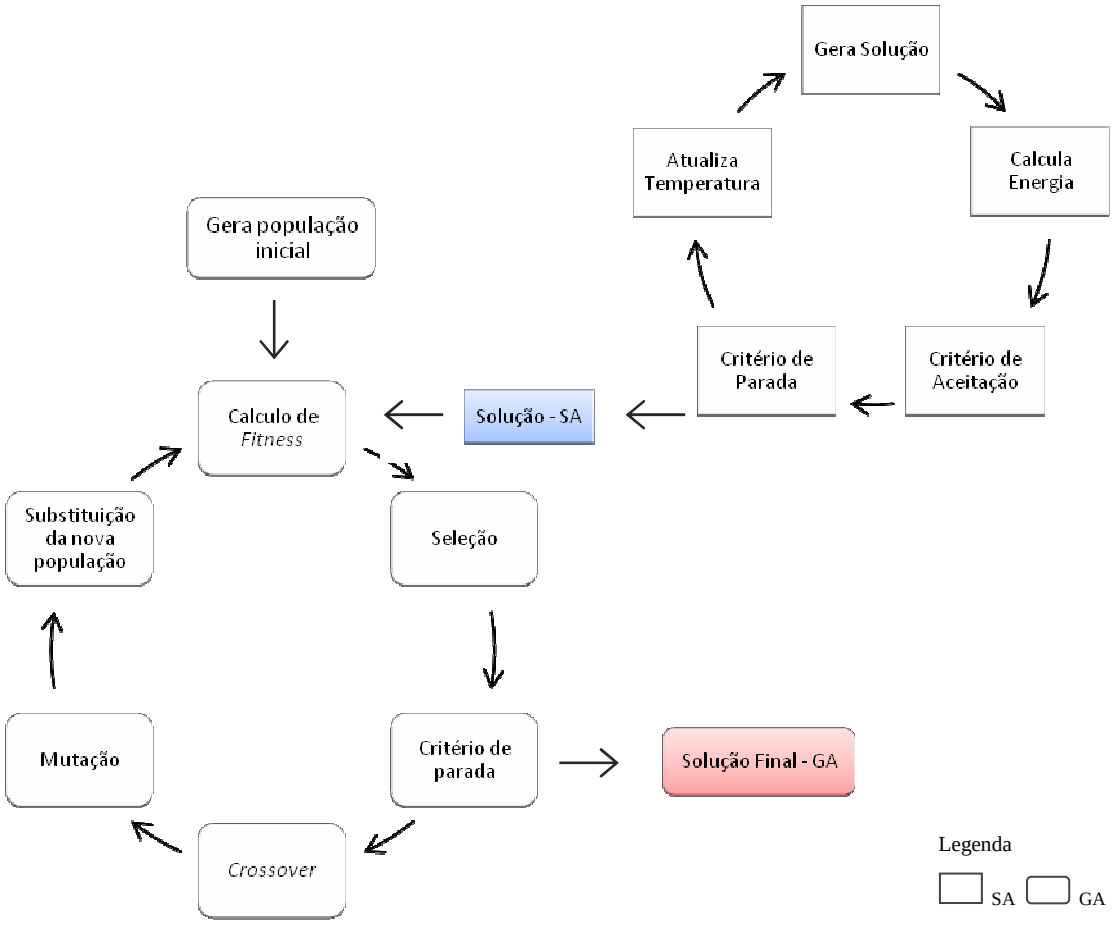

Figura 9. Fluxograma da meta-heurística proposta.

Conforme o fluxograma apresentado na Figura 9, segue a descrição dos componentes do esquema proposto.

Gera Solução: Gera aleatoriamente uma solução.

Calcula Energia: Realiza e associa à solução gerada, o cálculo da função objetivo.

Critério de Aceitação: Testa as soluções e aceita de acordo com o critério Metrópolis apresentado em 3.1.

Atualiza Temperatura: Atualiza a variável Temperatura que modifica a cada iteração do algoritmo a probabilidade de aceitar soluções aparentemente piores que a atual.

Solução – SA: É a solução obtida com o algoritmo SA e será utilizada para integrar a população inicial do AG. Tal solução passará suas características para a população do AG, que tentará melhorá-la ao longo das gerações.

Gera População Inicial: Corresponde a primeira etapa do algoritmo AG. Gera as soluções iniciais aleatoriamente satisfazendo (2).

Calcula *Fitness*: Realiza para cada solução encontrada, o cálculo da função objetivo.

Seleção: Seleciona apenas as melhores soluções para continuar na população como descrito em 4.3.

Critério de Parada: Ponto que determina se a melhor solução encontrada é a ótima, ou se haverá mais populações a serem geradas em busca de uma solução melhor.

***Crossover*:** Gera duas novas soluções a partir da interação entre outras duas, os pais dessas soluções. Estes pais são escolhidos aleatoriamente

Mutação: A mutação acontece implicitamente no processo de normalização, após a operação de *crossover*, conforme explicado em 4.4.

Substituição da nova população: A nova população é associada aos filhos provenientes do cruzamento da população anterior e a uma quantidade previamente definida de melhores soluções obtidas ao longo das gerações anteriores (Elitismo).

5. Resultados

O algoritmo utilizado na resolução do problema foi implementado na linguagem C++.

Os resultados foram gerados utilizando um computador com processador Intel Centrino Duo 1.83 GHz com 512MB de memória RAM.

Foram testadas separadamente, as heurísticas: SA (*Simulated Annealing*) e SAGA (*Simulated Annealing* seguido de Algoritmo Genético).

O objetivo dos testes foi verificar a melhora das soluções obtidas com SA quando se aplica AG para aperfeiçoar a solução.

Tabela 1. Parâmetros utilizados pelo SA

	Taxa de Redução (Temperatura)	Precisão da Solução	Total de Vizinhos Gerados	Número Máximo de Vizinhos
SA1	0,85000	0,0110	500000	50

Tabela 2. Parâmetros utilizados pelo AG

	Número de Cromossomos	Taxa de *Crossover*	Número de Populações	Taxa de Elitismo (%)
GA1	25	50	10	40%
GA2	25	180	28	40%

Abreviações utilizadas:

SAGA1 = SA1 e GA1. **SAGA2** = SA1 e GA2;

(%)Dif. Sol. Ótima = (%) Diferença da Solução Exata.

Tabela 3. Custo das soluções exatas para 5, 9 e 20 seções

Número de Seções	Custo
5	2,32987
9	2,39526
20	2,39795

Tabela 4. Resultados para um trocador de calor com 5 seções

	Solução SA	Solução SAGA
x_1	0,25	0,23094
x_2	0,15	0,16930
x_3	0,15	0,12239
x_4	0,05625	0,05966
x_5	0,39375	0,41768
Custo	**2,35387**	**2,33264**
(%)Dif. Sol. Ótima	1,0301	0,11889
Tempo (*s*)	0,547	0,578

Tabela 5. Resultados para um trocador de calor com 9 seções

	Solução SA	Solução SAGA
x_1	0,25	0,244662
x_2	0,1875	0,1834
x_3	0,1125	0,129314
x_4	0,15	0,0846967
x_5	0,00833333	0,0710613
x_6	0,00883838	0,0975455
x_7	0,0040404	0,0380638
x_8	0,00316804	0,0116672

x_9	0,27562	0,13959
Custo	**2,48345**	**2,41038**
(%)Dif. Sol. Ótima	3,68185	0,63125
Tempo (*s*)	0,875	0,922

Tabela 6. Resultados para um trocador de calor com 20 seções

	Solução SA	**Solução SAGA**	**Solução SAGA2**
x_1	0,25	0,250143	0,239944
x_2	0,1875	0,132455	0,184288
x_3	0,1875	0,0964227	0,133957
x_4	0,0340909	0,0882439	0,105064
x_5	0,00448565	0,0959922	0,0880078
x_6	0,0305839	0,105893	0,0568354
x_7	0,00501376	0,0116224	0,0456962
x_8	0,0231404	0,0258411	0,0337754
x_9	0,00555371	0,00620186	0,00755943
x_{10}	0,00971899	0,0108532	0,0217669
x_{11}	0,0041002	0,00457871	0,0156437
x_{12}	0,0369018	0,0412084	0,00331504
x_{13}	0,00311846	0,0034824	0,0035564
x_{14}	0,036382	0,040628	0,00290235
x_{15}	0,00395457	0,00787209	0,00384004
x_{16}	0,0026963	0,0361309	0,00289132
x_{17}	0,00730247	0,0261997	0,0156988
x_{18}	0,00230078	0,0042343	0,00387346
x_{19}	0,00460156	0,00767466	0,0134081
x_{20}	0,161054	0,00432251	0,0179768
Custo	**2,63948**	**2,51664**	**2,40088**
(%)Dif. Sol. Ótima	10,072	4,9490	0,12218
Tempo (*s*)	1,828	1,922	2,249

Conforme apresentado na Tabela 6, observa-se uma grande melhora entre a solução SA e SAGA. Entretanto, devido a maior complexidade do problema (20 seções), as configurações

de SAGA1 fizeram com que o algoritmo gerasse uma solução consideravelmente distante da ótima. A aplicação de uma configuração de parâmetros mais exigentes descrita por SAGA2 tornou-se necessária para este caso.

6. Conclusão

Conforme observado nas Tabelas 4, 5 e 6, a aplicação de um Algoritmo Genético para melhorar a solução do *Simulated Annealing* foi um sucesso. Com um pequeno custo computacional a meta heurística apresentada gerou bons resultados para o problema, e permitiu uma notória melhora na solução encontrada pelo SA.

Conforme verificado na Tabela 6, a aplicação dos parâmetros de SAGA2 propiciou resultados ainda melhores. Essa configuração mostrou o poder da heurística para resolver o problema mantendo um baixo custo computacional. Espera-se que esta heurística possa ser útil também para funções-objetivo mais complexas que a validada neste modelo.

7-Referências

Becceneri, J. C., Zinober, A. “Extraction of Energy in a Nuclear Reactor by Ants”. In: Simpósio Brasileiro de Pesquisa Operacional, 33, 2001, Campos do Jordão, SP. Anais. Campos do Jordão, SP: [s.n.], 2001.

Becceneri, J.C., Notas de aulas. INPE, 2002.

Goldberg, David E.. “Genetic Algorithms in Search, Optimization, and Machine Learning”. EUA: Addison-Wesley, 1989.

Metropolis N.; Rosenbluth A.W.; Rosenbluth M.N; Teller, A,H.; Teller, E. “Equation of State Calculation by Fast Computing Machines”, Journal of Chemical Physics, vol. 21, no. 6, pp. 1087-1091, 1953.

Viana, Valdisio. “Meta-Heurísticas e Programação Paralela em Otimização Combinatória”. Edições UFC, Fortaleza, 1998.

Griffiths, A. J. F.; Miller, J. H.; Suzuki, D. T.; Lewontin, R. C.; Gelbart, W. M. “Introdução à genetica”. 7. ed. Rio de Janeiro: Guanabara Koogan, 2002.