UNIVERSIDADE FEDERAL DO OESTE DO PARÁ – UFOPA
INSTITUTO DE ENGENHARIA E GEOCIÊNCIAS – IEG
BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

FERNANDO NOGUEIRA NAST

APRENDIZAGEM INCREMENTAL BASEADA EM POPULAÇÕES APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE COM VISTAS A RECARGA DE COMBUSTÍVEIS EM REATORES NUCLEARES

SANTARÉM
2016

FERNANDO NOGUEIRA NAST

APRENDIZAGEM INCREMENTAL BASEADA EM POPULAÇÕES APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE COM VISTAS A RECARGA DE COMBUSTÍVEIS EM REATORES NUCLEARES

Trabalho de Conclusão de Curso apresentado ao Bacharelado Interdisciplinar em Ciência e Tecnologia para obtenção do grau de Bacharel em Ciência e Tecnologia na Universidade Federal do Oeste do Pará, Instituto de Engenharia e Geociências.

Orientador: Anderson Alvarenga de Moura Meneses

SANTARÉM

2016

XVIII ENMC VI ECTM
XVIII ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL
VI ENCONTRO DE CIÊNCIA E TECNOLOGIA DE MATERIAIS
13 a 16 de outubro de 2015
SENAI CIMATEC - Salvador - BA

APRENDIZAGEM INCREMENTAL BASEADA EM POPULAÇÕES APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE COM VISTAS À RECARGA DE COMBUSTÍVEIS EM REATORES NUCLEARES

Fernando Nogueira Nast – fernandonnast@gmail.com
Patrick Vasconcelos da Silva – patrickvs@hotmail.com
Marcel Antonionni de Andrade Romano – antonionni@gmail.com
Anderson Alvarenga de Moura Meneses – anderson.meneses@pq.cnpq.br

Universidade Federal do Oeste do Pará, Instituto de Engenharia e Geociências, Laboratório de Inteligência Computacional – Santarém, PA, Brasil

Resumo. *O problema de Otimização do Gerenciamento de Combustível Intra-Núcleo (*In-Core Fuel Management Optimization*; OGCIN), ou otimização do projeto de Padrões de Carregamento (PCs) são denominações para o problema de otimização associado à operação de recarga de combustível em um reator nuclear. A OCGIN é considerada um problema de difícil resolução, pois considera aspectos da otimização combinatória e cálculos de análise e física de reatores. A fim de validarmos algoritmos para a solução da OGCIN, são utilizados problemas* benchmark *tais como o Problema do Caixeiro Viajante (PCV). No presente trabalho, com o intuito de fazer uma iniciação às técnicas de Inteligência Artificial na Engenharia Nuclear, implementamos o algoritmo Aprendizagem Incremental Baseada em População (*Population-Based Incremental Learning - *PBIL) e o aplicamos à solução do PCV Oliver30. O PBIL foi apresentado inicialmente por Baluja (1994) e aplicado à OGCIN por Machado (1999, 2005) Schirru et. al. (2006), Caldas e Schirru, (2008). O PBIL baseia-se no algoritmo genético e aprendizado competitivo, utilizando um vetor probabilidade que sofre alteração através de um operador evolucionário. Apresentamos os resultados preliminares do algoritmo PBIL na resolução do PCV Oliver30, para posterior aplicação a outros PCVs e em seguida à OGCIN.*

Palavras chave: *Otimização, Recarga de Combustível Nuclear, Aprendizado Incremental Baseada em Populações, Problema do Caixeiro Viajante.*

1. INTRODUÇÃO

O problema de Otimização do Gerenciamento de Combustível Intra-Núcleo (*In-Core Fuel Management Optimization*; OGCIN), ou otimização do projeto de Padrões de Carregamento (PCs) são denominações para o problema de otimização associado à operação de recarga de combustível em um reator nuclear, que é um problema clássico em Engenharia Nuclear. A OCGIN é considerada um problema de difícil resolução tanto por ser classificado como NP-difícil levando em conta aspectos da otimização combinatória, quanto pela própria complexidade dos cálculos de análise e física de reatores. O núcleo do reator de Angra 1, por exemplo, que contem 121 Elementos Combustíveis (ECs), possui um número de permutações

no núcleo de aproximadamente 10^{273} PCs possíveis. Porém, existem simetrias no núcleo e regras de colocação de ECs, o que faz com que este número caia para aproximadamente 10^{25}, que é um número ainda muito alto para resolver o problema por enumeração. Com esse número de combinações levaríamos 10^{19} anos para que pudesse avaliar todas as possibilidades, considerando um tempo de execução de código de reatores de aproximadamente 2 minutos, para cada avaliação de PC.

Tendo em vista a complexidade da OGCIN, no início da década de 90, foram desenvolvidos códigos computacionais heurísticos com o objetivo de reduzir o espaço de busca, mantendo uma solução adequada do ponto de vista prático. Um destes códigos é o FORMOSA (*Fuel Optimization for Reloads: Multiple Objectives by Simulated Annealing),* que utiliza o Simulated Annealing para otimizar a recarga (Kropackzek e Turinsky, 1991). É conveniente que esses métodos antes que sejam aplicados ao problema da recarga, sejam validados e para isso existem os problemas *benchmark*, tais como o Problema do Caixeiro Viajante (PCV) (Papadimitrious e Steiglitz, 1982). O PCV é um problema combinatório que envolve a visita a *n* cidades uma única vez até chegar à primeira cidade novamente, fazendo-se o menor percurso possível. O PCV será abordado na seção 4 deste trabalho.

A OCGIN é diferente do PCV em três aspectos fundamentais: na dimensão do problema; na grande dificuldade em obter heurística local para a OCGIN; e na analogia entre cidades com ECs e suas posições, e rotas com configuração de núcleo (De Lima, 2005).

Entretanto, são problemas similares no que se refere à representação das soluções por meio das permutações, onde os elementos de um conjunto devem ser ordenados de modo que cada elemento ocupe uma única posição na ordenação (Meneses et. al, 2009).

Partindo desse preceito, o presente trabalho visa a validação do algoritmo evolucionário PBIL, utilizando um problema *benchmark*, apresentando resultados preliminares da aplicação do PBIL ao PCV Oliver30, para posterior aplicação à OGCIN.

O restante do trabalho está organizado da seguinte forma: Na seção 2 apresentamos os aspectos básicos da OGCIN e as particularidades do problema do padrão de carregamento em uma Usina Nuclear. Na seção 3 a descrição do Algoritmo PBIL. Na seção 4 breve apresentação do PCV e os resultados do PBIL aplicado ao Oliver30. Na seção 5 são expostas as conclusões e futuros passos da pesquisa.

2. PROBLEMA DE OTIMIZAÇÃO DE COMBUSTÍVEL INTRA-NÚCLEO

A recarga de um reator nuclear faz-se necessária quando a queima de Elementos Combustíveis (EC) no núcleo do reator não é suficiente para manter o reator funcionando a potência nominal ou após um período de operação chamado de ciclo.

Devido ao fato de a queima do EC depender do tempo de exposição e de sua posição no núcleo do reator, é comum que após cada ciclo os ECs apresentem características neutrônicas diferentes, de modo que, no processo de recarga de combustível, cerca de 1/3 dos ECs são substituídos por novos. Sendo assim, o núcleo do próximo ciclo é formado por ECs novos e por ECs reutilizados, e a combinação formada por eles gera um novo PC.

A OGCIN consiste em determinar o PC de forma que satisfaça as regras de simetria do núcleo do reator (Machado, 2005), e prolongue a duração do ciclo de recarga ou minimize o fator de pico de potência radial do núcleo, atendendo a restrições de segurança.

A determinação de um PC é um problema classificado como NP-difícil, o que significa que sua complexidade cresce com o número de ECs de forma não-polinomial (De Lima, 2005). Por exemplo, o núcleo reator de Angra 1 há 121 ECs, gerando um número de permutações no núcleo de aproximadamente 10^{273} PCs possíveis. Porém, existem as regras de simetria no núcleo e as de colocação de EC, o que faz com que este número caia para aproximadamente 10^{25}, um número ainda muito alto para resolver o problema por enumeração. Com esse número de combinações levaríamos 10^{19} anos para testar todas as possibilidades, considerando um tempo de execução de código de reatores que leve aproximadamente 2 minutos.

A busca por PCs que otimizem a queima de EC é de suma importância, pois com isso, consegue-se um número maior de Dias Efetivos a Plena Potência (DEPP), que por outro lado equivale a um ganho de algumas centenas de milhares de dólares (De Lima, 2005).

Essas características de análise combinatória, segurança e economia, associadas tornam a OGCIN um problema de otimização desafiador, pois instiga o desenvolvimento de novas técnicas computacionais de otimização.

3. APRENDIZAGEM INCREMENTAL BASEADA EM POPULAÇÕES

O algoritmo PBIL (*Population-Based Incremental Learning*) (Baluja, 1994; Machado, 2005) é um algoritmo evolutivo baseado nos princípios do algoritmo genético (AG) e da aprendizagem competitiva simples.

O objetivo do algoritmo é associar um vetor probabilidade com valores reais entre 0,00 e 1,00, a cada posição, que ao ser decodificado gere indivíduos que apresentem as melhores soluções para o problema a ser otimizado. Por exemplo, o vetor probabilidade especifica a probabilidade de cada posição obter o valor "1", desta forma, um bom vetor probabilidade, para um problema com oito bits, a ser obtido ao final do processo de busca é apresentado na Fig. 2.

0,02	0,95	0,01	0,98	0,99	0,10	0,03	0,95

Figura 1 - Representação do vetor probabilidade

Tal vetor, ao ser decodificado, irá gerar, com alta probabilidade, o cromossomo apresentado na Fig. 2.

0	1	0	1	1	0	0	1

Figura 2 - Representação de um indivíduo gerado a partir do vetor probabilidade da Fig. 1

Como no algoritmo PBIL toda a população de indivíduos é definida a partir do vetor probabilidade, os operadores empregados para a evolução desta população não são definidos sobre a população, como acontece no AG, mas sim no vetor probabilidade.

A fim de se obter uma maior diversidade da população no início do processo de busca, cada posição do vetor probabilidade é definida com valor inicial de 0,50. Isto determina que a probabilidade de geração dos valores "0" e "1" em cada posição da cadeia de bits seja igual. Esta igualdade na geração de valores faz com que a população inicial gerada pelo PBIL seja aleatória. É muito importante para os algoritmos evolucionários que sua população inicial seja gerada aleatoriamente, fazendo com que não haja favorecimento em nenhuma parte do espaço de busca.

Os valores do vetor probabilidade vão sendo gradativamente alterados – com o passar das gerações – do valor inicial de 0,50 para valores próximos a 0,00 ou a 1,00, com a finalidade de representar os melhores indivíduos encontrados na população, para cada geração.

Para que aconteça essa variação nos valores de cada posição do vetor probabilidade é necessária a atualização deste a cada geração de uma população. Esta atualização é feita utilizando dois vetores: o vetor que possui a melhor avaliação da *fitness* e o vetor que possui a pior avaliação da *fitness*. O melhor vetor é aquele que apresenta maior valor da *fitness* e altera o vetor probabilidade de forma que este se aproxime de sua representação. O pior vetor é aquele com menor valor da *fitness* e que altera o vetor probabilidade de forma que este se afaste de sua representação.

Existindo assim, funções para a atualização do vetor probabilidade.

- Para o melhor vetor:

$$\vec{P}(i) = \vec{P}(i) * (1 - Lr\_pos) + \vec{v}(i) * Lr\_pos \quad (1)$$

- Para o pior vetor:

$$\vec{P}(i) = \vec{P}(i) * (1 - Lr\_neg) + \vec{v}(i) * Lr\_neg \quad (2)$$

Onde:

Lr_pos: taxa de aprendizagem positiva;
Lr_neg: taxa de aprendizagem negativa;
$\vec{P}(i)$: valor do vetor probabilidade na posição i;
$\vec{v}(i)$: valor do vetor solução escolhido na posição i.

Com isso, pode-se dizer que o algoritmo PBIL segue os seguintes passos. Inicialmente, é atribuído o valor inicial 0,50, a cada uma das posições do vetor probabilidade. Na etapa seguinte, são formados os indivíduos binários, como mostrado nas Fig. 2 e Fig. 3. Estes indivíduos binários são então decodificados usando o processo descrito por Machado (2005), utilizando uma implementação baseada em *Random Keys* (Bean, 1994) e avaliados pelo seu valor de *fitness*, para que em seguida sejam identificados os dois vetores (o melhor e o pior)

que serão utilizados no processo de atualização do vetor $\vec{P}$. A atualização deste é um processo iterativo onde parte do valor original da posição é mantida enquanto um incremento é atribuído à mesma. O resultado são variações nos valores das posições de $\vec{P}$, fazendo com que os indivíduos gerados sejam, ao mesmo tempo, cada vez mais similares ao melhor indivíduo avaliado até então e menos similares ao pior deles. Nesta etapa dois parâmetros desempenham um papel de fundamental importância: as taxas de aprendizagem positiva (Lr_pos) e negativa (Lr_neg). A Lr_pos é utilizada com o objetivo de determinar quanto os novos indivíduos gerados irão se aproximar do melhor indivíduo conhecido até então. A Lr_neg, por outro lado, determina quanto os novos indivíduos gerados serão afastados daquele que é o indivíduo com pior nível de *fitness*. Esta é a descrição do algoritmo PBIL padrão. Existem outras variantes do PBIL disponíveis na literatura que utilizam alguns parâmetros e etapas adicionais aos descritos aqui. O pseudocódigo do PBIL é exibido na Fig. 4, ressaltando que não foi utilizada a Eq. 2 em nossa implementação.

#INICIALIZAÇÃO DO VETOR PROBABILIDADE ($\vec{P}$)

Para j = 0 até j = ao tamanho do vetor

$\vec{P}$[j] = { 0,5 }

#LOOP DE GERAÇÕES

* Enquanto (ITERAÇÃO não for igual ao Número Máximo de Gerações)

Para i = 0 até i = o número de cromossomos

Para j = 0 até j = ao tamanho do vetor

Vetor solução ($\vec{S}$) [j] = gerado de acordo com as probabilidades de $\vec{P}$

* Avaliação $\vec{S}$ [j] = valor da *fitness*

* Encontrar o melhor vetor $\vec{S}$

* Atualização do $\vec{P}$ utilizando o melhor vetor solução

Para j = 0 até j = ao tamanho do vetor

$$\vec{P}(j) = \vec{P}(j) * (1 - Lr\_pos) + melhor\ vetor\ \vec{S}(j) * Lr\_pos$$

Figura 3 - Pseudocódigo do PBIL (Machado, 2005)

4. PBIL APLICADO AO PROBLEMA DO CAIXEIRO VIAJANTE

4.1. O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um famoso problema de análise, e referência, em otimização combinatória, classificado como NP-difícil (Papadimitrious e Steiglitz, 1982). Existem dois principais tipos de PCV: os simétricos e os assimétricos. O tipo aqui analisado é o PCV simétrico, onde dadas duas cidades i e j diferentes, a distância d_{ij} para ir de uma a outra é igual ao caminho inverso, isto é, $d_{ij} = d_{ji}$. O objetivo do PCV é visitar todas as cidades uma única vez, retornando para a cidade inicial, e assim, determinar o trajeto de menor distância a ser percorrida. Consequentemente, o PCV simétrico tem $[(n-1)!/2]$ soluções possíveis (onde n corresponde à dimensão do problema), e apesar da simples formulação, o PCV é um problema de difícil solução. Podemos citar, por exemplo, um problema com 30 cidades, como o Oliver30, onde teremos 29!/2 permutações, o que nos fornece um campo de aproximadamente $4{,}42\ x\ 10^{30}$ possíveis soluções e, levando em consideração o tempo de avaliação de cada solução (*fitness*), no código apresentado, de $9{,}266\ x\ 10^{-06}\ s$, levaríamos um total de aproximadamente $1{,}3\ x\ 10^{18}$ anos para avaliar todas as soluções por enumeração. É mais do que o tempo estimado de existência do universo em anos (mais de 13 bilhões de anos).

4.2. Decodificação de um PCV no PBIL

Para um problema com *N* cidades, a codificação requer uma cadeia de bits igual o valor *C*, resultado da Equação 3. Esta cadeia é subdividida em subcadeias, as quais são atribuídas uma à cada cidade. Uma subcadeia possui um comprimento *M*, onde *M* é igual à Equação 4. Cada subcadeia é decodificada em um número inteiro no intervalo de [0, *N*-1]. Depois de definidos o tamanho da cadeia e da subcadeia, é utilizado o processo usado por Machado (2005) baseado no *Random Keys* (Bean, 1994). Neste processo a subcadeia de bits é decodificada em um valor inteiro e relacionado à uma cidade. A cidade com o menor valor inteiro vem primeiro na ordenação final, a cidade com a segunda menor vem em segundo lugar, e assim por diante. No caso de empate no valor inteiro da cidade, a decodificação, cujo a subcadeia vem primeiro na sequência de bits, vem primeiro na ordenação final. A Figura 4 apresenta um exemplo com quatro cidades do processo descrito anteriormente.

$$C = N * \log_2 N \tag{3}$$

$$S = \log_2 N \tag{4}$$

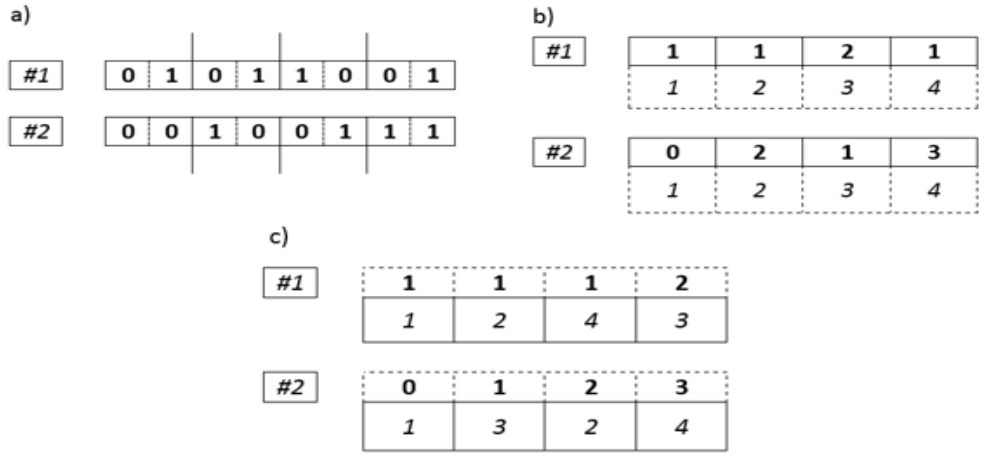

Figura 4 – Representação de dois indivíduos e codificação de soluções para um PCV com 4 cidades: (a) codificação binária obtida a partir do vetor probabilidade; (b) decodificação dos binários e os inteiros associados abaixo; (c) resultado após a aplicação do *Random Keys*, representando duas soluções válidas para o PCV.

Na Figura 4 está representado dois possíveis casos para um problema com 4 cidades. Este problema requer uma cadeia com um total de 8 bits, os quais são resultados da Eq. 3, onde teremos $4*log_2 4$ e subcadeias com 2 bits (Eq. 4 - $log_2 4$). Para o indivíduo 1 (*#1*), possuímos valores repetidos quando decodificamos todas as subcadeias, já no caso do indivíduo 2 (*#2*) não há (Fig. 4b). Na ordenação final (Fig. 4c), os valores decodificados são organizados de forma crescente e seguindo as regras do *Random Keys*, fazendo com que os inteiros associados nos forneçam uma configuração de rota de cidades.

4.3. Resultados da Aplicação do PBIL ao *benchmark* PCV Oliver30

A fim de analisar o comportamento do código PBIL utilizando o PCV Oliver30 (Dorigo e Gambardella,1997; Meneses et al., 2009), o qual possui um mínimo global com avaliação igual a 423,74, foram executadas quatro variações na Lr_pos e em cada variação foram realizados dez testes independentes (com inicialização aleatória). Em todos os testes o tamanho das populações foi o mesmo (100 indivíduos). A implementação do código PBIL, apresentado neste trabalho, foi desenvolvida em linguagem C/C++, no ambiente de programação *Visual Studio Express 2013 for Desktop* (fornecido gratuitamente na web pela Microsoft Corporation). Há de se ressaltar também, que no código, usou-se apenas o vetor com a melhor *fitness* para a atualização do vetor probabilidade, para melhor estudo do algoritmo, como apresentado por Machado (2005). O número máximo de iterações do algoritmo foi definido em 10.000.

Os resultados dos dez testes, usando a Lr_pos com melhor desempenho, foram usados para plotar o gráfico na Fig. 6, onde encontram-se através da média das *fitnesses* a cada 100 iterações, e o teste com o melhor desempenho entre eles na Fig. 5. (A Lr_pos usada foi 0,01 e a população com um total de 100 cromossomos).

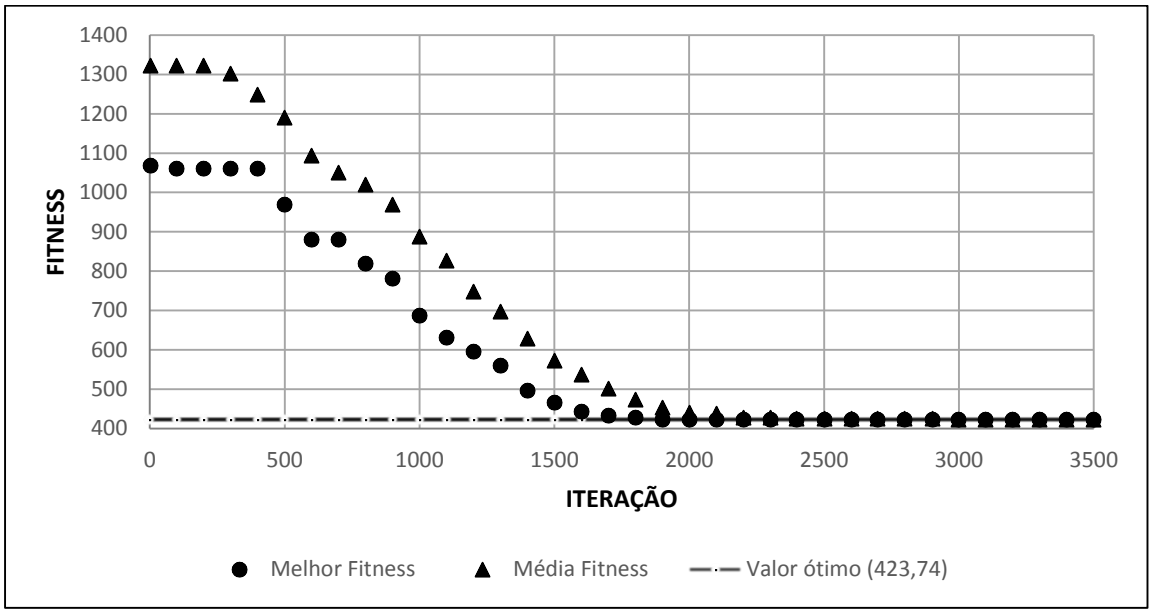

Figura 5 - O teste com melhor desempenho dentre os demais com a Lr_pos igual a 0,01, observando que dentro das 10.000 iterações, houve convergência após a iteração 2.500.

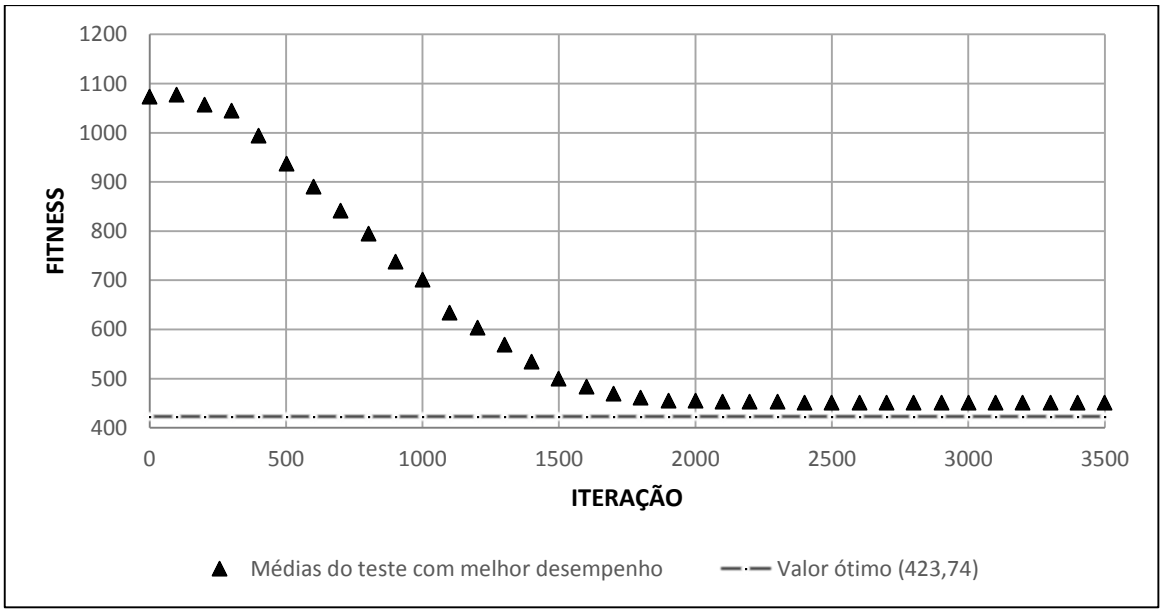

Figura 6 - As médias dos 10 testes com Lr_pos igual a 0,01, observando que dentro das 10.000 iterações, houve convergência, em média, após a iteração 2.000.

Os resultados obtidos demonstram que usando apenas a Lr_pos, o código foi eficaz, no que se refere a encontrar o valor mínimo conhecido do problema. No entanto, utilizando apenas o melhor vetor para a atualização do vetor probabilidade, foi notório que o código encontrou grande dificuldade para tal feito, fato também verificado em outros três testes, onde todos utilizaram os mesmos valores de parâmetro, sendo modificado apenas o valor da Lr_pos (teste 1 - 0,003; teste 2 - 0,005; teste 3 - 0,009), e nenhum deles alcançando o resultado desejado. Foi possível perceber também que o PBIL em pouco mais de 2.000 iterações (Fig.5) já obteve um vetor probabilidade convergido, ou seja, os valores nas posições do vetor já são iguais a 1,00 ou 0,00, ou muito próximo disso, pois a média é capaz de nos fornecer tal comportamento.

5. CONCLUSÃO

Neste trabalho apresentamos o desempenho do PBIL utilizando apenas o melhor vetor para a atualização do vetor probabilidade aplicado ao *benchmark* PCV Oliver30, para futura aplicação à OGCIN. Foram apresentados os principais aspectos do algoritmo e os resultados do teste com melhor desempenho. Considerando que a população inicial não possui conhecimento prévio do problema e que o algoritmo apresentado não utiliza o pior vetor para a atualização, que é um passo importante para o código, ou seja, que o potencial do PBIL não foi utilizado na sua totalidade, pode se dizer que o algoritmo atingiu resultados satisfatórios, pois encontrou o mínimo do problema apresentado.

O próximo passo da pesquisa é a continuidade da aplicação do PBIL a outros *benchmarks* PCV, para futuramente aplicá-lo à OGCIN.

Agradecimentos

A.A.M.M. agradece ao apoio financeiro do CNPq (Projeto nº. 472912/2013-5). M.A.A.R. agradece ao apoio da UFOPA. P.V.S. agradece ao apoio da UFOPA e financeiro da FAPESPA. F.N.N. agradece ao apoio financeiro da UFOPA.

REFERÊNCIAS

Baluja, Shumeet, (1994) "*Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning"*, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Baluja, Shumeet, (1995) "*An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics"*, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Bean, J.C. (1994), *"Genetic Algorithms and Random Keys for Sequencing and Optimization,* ORSA Journal on Computing, vol. 6, nº 2, Spring.

Caldas, G.H.F., Schirru, R., "*Parameterless evolutionary algorithm applied to the nuclear reload problem*". Annals of Nuclear Energy 2008, 35, 583–590.

De Lima, Alan M.M. (2005), *"Recarga de Reatores Nucleares Utilizando Redes Conectivas de Colônias Artificiais*", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Dorigo, M., Gambardella, L.M., "*Ant colony system: a cooperative learning approach to the traveling salesman problem*". IEEE Transactions on Evolutionary Computation, 1997, 1 (1), 53–66.

Kropackzek, D.J., Turinsky, P.J., (1991), "*In-Core Nuclear Fuel Management Optimization for Pressurized Water Reactors Utilizing Simulated Annealing",* Nuclear Technology, v. 95, nº 9, pp. 9-31.

Machado, Marcelo D. (2005), *"Algoritmo Evolucionário PBIL Multi_Objetivo Aplicado ao Problema da Recarga de Reatores Nucleares",* Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro.

Machado, M.D., "*Um Novo Algoritmo Evolucionário com Aprendizagem LVQ para Otimização de Problemas Combinatórios como a Recarga de Reatores Nucleares*", M. Sc. dissertação, 1999, COPPE/UFRJ, Brasil.

Meneses, A. A. M., Gambardella, Luca Maria, Schirru, Roberto . "*A new approach for heuristics-guided search in the In-Core Fuel Management Optimization.*", Progress in Nuclear Energy, v. 52, p. 339-351, 2009.

Papadimitriou, C.H., Steiglitz, K., "*Combinatorial Optimization"*, EUA, Prentice-Hall, 1982.

Schirru, R., De Lima, A.M.M., Machado, M.D., "*Parallel evolutionary methods applied to a PWR core reload pattern optimization. In: Proceedings of the Seventh International Conference on Applied Artificial Intelligence*", 2006, FLINS, Italy

APÊNDICE A

POPULATION-BASED INCREMENTAL LEARNING APPLIED TO TRAVELING SALESMAN PROBLEM WITH VIEWS TO FUEL CHARGING IN NUCLEAR REACTORS

The In-Core Fuel Management Optimization (ICFMO) problem, or optimizing Loading Standards project (LSs) are designations for the optimization problem associated with fuel refilling operation in a reactor nuclear. The ICFMO is considered a problem difficult to solve, considering aspects of combinatorial optimization and complex calculations analysis and reactor physics. In order to validate algorithms for the solution of ICFMO, benchmark problems are used such as the Traveling Salesman Problem (TSP). In this study, in order to make a start on Artificial Intelligence techniques in Nuclear Engineering, we implemented the Population-Based Incremental Learning algorithm (PBIL) and applied to the TSP Oliver30 solution, analyzing the results with a view to their application to ICFMO. The PBIL was initially presented by Baluja (1994) and applied to ICFMO by Machado (1999, 2005), Schirru et al (2006), Caldas and Schirru (2008). The PBIL based on the genetic algorithm and competitive learning, using a vector that undergoes change probability via an evolutionary operator. We present the preliminary results of PBIL algorithm in solving the TSP Oliver30 for subsequent application to other TSPs and then the ICFMO.

Keywords: *Optimization, Nuclear Fuel Reload, Population-Based Incremental Learning, Travelling Salesman Problem.*