10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

ABORDAGEM DE ENXAME DE PARTÍCULAS COOPERATIVO PARALELO APLICADO NA OTIMIZAÇÃO DA PREDIÇÃO DA ESTRUTURA DE PROTEÍNAS UTILIZANDO O MODELO AB EM 2D

Fabio Alessandro Guerra[1,2], Leandro dos Santos Coelho[2], Diego Humberto Kalegari[1], Thiago Enrique Volpe Pereira[1], Helon Vicente Hultmann Ayala[1], Mariana Cristina Coelho[1]

[1]LACTEC - Instituto de Tecnologia para o Desenvolvimento, DPEL - Departamento de Eletricidade, DVSE - Divisão de Sistemas Elétricos, BR 116 - Km 98 - Nº 8813, CEP: 81531-980, Curitiba, Paraná, Brasil.
[2]PUCPR - Pontifícia Universidade Católica do Paraná, PPGEPS - Programa de Pós-Graduação em Engenharia de Produção e Sistemas, LAS - Laboratório de Automação e Sistemas, Rua Imaculado Conceição - Nº 1155, CEP: 80215-901, Curitiba, Paraná, Brasil.
E-mails: guerra@lactec.org.br; leandro.coelho@pucpr.br; kalegari@lactec.org.br; thiago.pereira@lactec.org.br; helonayala@gmail.com; mariana@lactec.org.br

Abstract – Protein structure prediction is a well-known problem in bioinformatics. Identifying protein native conformation makes it possible to predict its function within the organism. Knowing this also helps in the development of new medicines and in comprehending how some illnesses behave. During the past year some techniques have been proposed to solve this problem, but its high cost made it necessary to build models that simplify the protein structures. However, even with the simplicity of these models identifying the protein native conformation remains a highly complex, computationally challenging problem. Particle swarm optimization is a population-based swarm intelligence algorithm that shares many similarities with evolutionary computation techniques. However, this algorithm is driven by the simulation of a social psychological metaphor motivated by collective behaviors of bird and other social organisms instead of the survival of the fittest individual. This paper proposes a Particle Swarm Optimization with a Cooperative and Parallel approach to solve the protein structure prediction problem. The model used to represent the protein structure is the Toy Model (also known as the AB Model) in 2D. This work compares the implementations of three versions of PSO algorithm using a parallel architecture (master-slave).

Keywords – Swarm Intelligence, Cooperative Particle Swarm Optimization, Parallel Computation, Protein Structure Prediction, Toy Model (AB Model).

1 Introdução

A otimização de sistemas baseadas em técnicas de inteligência coletiva (ou também denominada de inteligência de enxames) tem obtido maior interesse do meio acadêmico na última década. Estas são caracterizadas pela descentralização da maneira de interação do coletivo ou enxames. A vantagem destas técnicas sobre as tradicionais é a robustez e a flexibilidade, propriedades que tornam a inteligência coletiva uma abordagem capaz de melhorar o desempenho da solução de problemas complexos. A otimização de sistemas utilizando algoritmo de enxame de partícula tem sido largamente estudada e testada desde sua criação por Kennedy e Eberthart em 1995 [1],[2]. Os algoritmos de otimização de sistemas tem por objetivo principal ser um método de busca, onde o foco é encontrar uma solução global compatível ao sistema que está sendo analisado. Algumas características podem ser mencionadas para problema de otimização, sendo: número e tipo de variáveis, não linearidade da função objetivo, ou seja, da função que se deseja otimizar, restrições e critérios de parada da busca [3].

A otimização por enxame de partículas (*PSO – Particle Swarm Optimization*) provê uma estrutura capaz de resolver funções complexas de maneira eficaz. O PSO clássico (original) é uma técnica de otimização baseado em população chamada de enxame (*swarm*). Simplificadamente cada partícula representa uma possível solução para a tarefa de otimização, neste artigo sendo uma minimização sem restrições. Durante cada interação, cada partícula acelera na direção de sua melhor solução pessoal (*pbest*) adicionalmente também na direção da melhor solução global (*gbest*) do conjunto de partículas. Isto significa que se uma partícula descobre uma nova solução próspera, todas as outras soluções são influenciadas por essa partícula movendo-se para sua proximidade, assim explorando a região mais a fundo. Através do PSO clássico foi embutida, no mesmo, a aprendizagem cooperativa proposta por Bergh e Engelbrecht em 2004 [4] (*CPSO – Cooperative Particle Swarm Optimization*) que tem por objetivo principal definir uma cooperação entre as partículas dividindo o problema em diversas partes criando o enxame cooperativo. Deste modo, também direcionou-se o estudo utilizando a computação paralela, afim de distribuir a execução propriamente dita da otimização e dividir as partículas cooperadas em um espaço amostral computacional altamente otimizado.

Isto posto, o objetivo principal deste artigo é apresentar a otimização da predição da estrutura de proteínas utilizando o modelo AB 2D, computação paralela em enxame de partículas cooperativo. Assim, este artigo está organizado da seguinte maneira. Na seção 2 será apresentado o problema da predição da estrutura de uma proteína assim como o modelo AB 2D, que é o estudo de caso a ser otimizado. Na seção 3 serão apresentados os algoritmos de exames de partículas clássico e cooperativo. A paralelização dos algoritmos será apresentada na seção 4. E os devidos resultados e conclusões são ilustrados nas seções 5 e 6 respectivamente.

2 Estudo de Caso – Modelo AB 2D

O problema da predição da estrutura de proteínas (PPEP) é um dos problemas mais desafiadores da área da Bioinformática, e vêm sendo explorado a fundo nos últimos anos. A identificação da conformação nativa de uma proteína, estado em que uma proteína apresenta a sua organização estrutural máxima e em que o grau de eficiência de utilização de energia é o menor possível (energia livre será a menor possível), permite predizer a sua função no organismo. Este conhecimento também é útil no desenvolvimento de novos fármacos ou na compreensão do mecanismo de várias doenças. Várias técnicas têm sido propostas para resolver este problema. Porém, o alto custo envolvido levou ao surgimento de vários modelos que simplificam, em parte, as estruturas proteicas. No entanto, mesmo com os modelos mais simplificados, a complexidade do problema traz inúmeros desafios computacionais na busca da sua conformação nativa.

O modelo para representar a estrutura de uma proteína utilizado neste trabalho é conhecido como Modelo AB, ou *Toy Model*, foi apresentado primeiramente por Stillinger e Head-Gordon [5],[6] e utiliza o princípio da hidrofobicidade, assim como o Modelo HP [7]-[9], para a simplificação dos aminoácidos que compõe a estrutura de uma proteína. A hidrofobicidade indica o grau de repulsão à água de um determinado aminoácido e é conhecida como a principal força de interação entre os que compõe a estrutura de uma proteína. Neste modelo os aminoácidos são convertidos em dois grupos: A (hidrofóbicos) e B (hidrofílicos/polar).

No modelo AB em 2D, as ligações entre A-A (hidrofóbico-hidrofóbico) continuam apresentando valor de energia igual a 1. Nas ligações B-B (polar-polar) apresentam energia igual a $+1/2$. E as ligações A-B (hidrofóbico-polar) apresentam energia igual a $-1/2$. Analisando os valores de energia previamente definidos, percebe-se, no presente modelo, que as ligações entre os aminoácidos AA apresentam uma energia de grande atração, enquanto entre os BB apresentam uma energia de média atração e entre os AB apresentam características de média repulsão. A força das ligações entre os aminoácidos será considerada na avaliação da energia do dobramento (avaliação ou *fitness*). As ligações entre os aminoácidos estão agrupadas por meio de ângulos diedrais formado entre eles, sendo que o ângulo é sempre representado em relação ao aminoácido predecessor, esses ângulos estão sempre restritos aos valores de $-\pi$ e π radianos.

A Figura 1 apresenta a representação hipotética de uma proteína no modelo AB em 2D. A proteína é composta por 9 (nove) aminoácidos, sendo que cada um está conectado ao próximo da cadeia por meio do ângulo diedral, também conhecido como ângulo de torção, que é responsável pelo seu dobramento na cadeia. No caso do dobramento 2D, o modelo é composto de $n-2$ ângulos necessários para gerar o dobramento, pois a ligação entre os dois primeiros aminoácidos é fixa com ângulo 0.

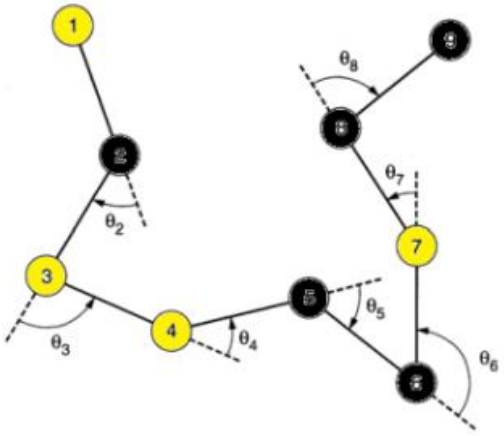

Figura 1 - Representação Genérica de uma proteína composta de nove aminoácidos.

A energia livre do dobramento do modelo 2D é composta, basicamente, por duas partes: (i) a energia intermolecular; e (ii) a energia potencial. A energia potencial é a energia formada entre os monômeros não conectados, ou seja, é aquela em que as forças dos aminoácidos não interconectados vão exercer sobre o aminoácido atual em um dobramento. Essa energia é conhecida como potencial de Lennard-Jones. Por sua vez, a energia intermolecular é a energia que depende apenas dos ângulos entre os aminoácidos e representa os *backbones* potenciais [10],[11]. O modelo matemático que descreve a energia livre do dobramento AB em 2D para uma proteína com N aminoácidos é definido pela equação:

$$E = \frac{1}{4}\sum_{k=1}^{N-2}(1-\cos\theta_k) + 4\sum_{i=1}^{N-2}\sum_{j=i+2}^{N}\left(\frac{1}{r_{ij}^{12}} - \frac{C(\sigma_i,\sigma_j)}{r_{ij}^{6}}\right) \quad (1)$$

O primeiro somatório representa o custo para dobrar os aminoácidos interconectados na sequência por meio do ângulo θ_k. O segundo termo, denominado potencial de Lennard-Jones, depende das distâncias entre os aminoácidos não adjacentes no

decorrer da estrutura e é influenciado pelas forças das ligações entre aminoácidos da sequência analisada, sendo que $\mathfrak{G}_i = A$ para aminoácidos hidrofóbicos, $\mathfrak{G}_j = B$, para monômeros polares e r_{ij} é a distância entre os aminoácidos i e j, tal que:

$$C(\mathfrak{G}_i, \mathfrak{G}_j) = \begin{cases} +1 \; se \; \mathfrak{G}_i, \mathfrak{G}_j = A \\ +\frac{1}{2} \; se \; \mathfrak{G}_i, \mathfrak{G}_j = B \\ \frac{1}{2} \; se \; \mathfrak{G}_i \neq \mathfrak{G}_j \end{cases} \quad (2)$$

3 Algoritmo Cooperativo de Enxame de Partículas

A proposta do algoritmo PSO surgiu do desenvolvimento de simulações por computador do movimento de bando de pássaros (enxames/nuvem/população) e cardumes. Baseado fortemente no controle das distâncias entre os indivíduos da população foi observado e determinado o esforço dos pássaros para manter uma distância ótima entres os indivíduos.

Este desenvolvimento baseou-se no controle das distâncias entre os indivíduos da população sendo que o sincronismo no comportamento dos enxames foi percebido como um esforço dos pássaros ou partícula em manter uma distância ótima entre os mesmos. Teoricamente cada indivíduo pode obter informações das descobertas e da experiência anterior de todos os outros indivíduos, enxame, na busca de seu objetivo. É exatamente neste sentido que o desenvolvimento do PSO se baseia, sendo uma troca de informações entres os indivíduos para oferecer uma vantagem evolucionária [12],[13].

Neste sentido Kennedy e Eberthart em 1995 [1],[2] desenvolveram a técnica de PSO que de maneira semelhante aos algoritmos genéticos é baseado em uma população/enxame, e cada elemento desta população é denominado de indivíduo/partícula, e cada partícula é uma solução em potencial. Entretanto, diferentemente dos algoritmos genéticos, o PSO não possui operadores como o cruzamento e a mutação. O PSO não implementa a sobrevivência do indivíduo mais adequado ao meio, mas, ao invés disso, implementa a simulação do comportamento social e cognitivo. Este algoritmo pode ser facilmente implementado e possui características de convergência estáveis com boa eficiência computacional [1],[14].

O PSO possui dois operadores: velocidade e posição. Inicialmente é determinado um enxame inicial com posições aleatórias. A cada uma das partículas que fazem parte deste enxame é definido uma velocidade, assim fazendo as partículas se movimentarem através do espaço de busca. Cada partícula possui seu $pbest$, que pode ser considerado uma memória que contém sua melhor posição visitada no espaço de busca. Da mesma maneira que a partícula, o enxame possui seu $gbest$ que é a melhor posição global visitada por todas. Assim o algoritmo acelera as partículas em direção às posições e, com um peso de aceleração aleatório a cada passo de tempo. Formalmente as partículas obtém seu valor de velocidade e posição através das equações:

$$V_{id}(t+1) = W.V_{id}(t) + c_1.rand_1.\left(pbest_{id} - X_{id}(t)\right) + c_2.rand_2.\left(gbest - X_{id}(t)\right) \quad (3)$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1)\Delta t \quad (4)$$

onde $\Delta t = 1$, t representa a iteração atual e $t+1$ representa a próxima iteração, V_{id} e X_{id} representam a velocidade e a posição da partícula i, com dimensão d, respectivamente, $rand_1$ e $rand_2$ são dois números aleatórios com distribuição uniforme dentro do intervalo [0,1], $pbest$ é o melhor pessoal da partícula e o $gbest$ é o melhor global do enxame de partículas [15].

A equação (3) é utilizada para atualizar a velocidade de cada uma das partículas. Nesta equação é utilizada a velocidade na iteração anterior, multiplicada pelo momento de inércia, como um fator que pondera sua velocidade sendo responsável por um ajuste dinâmico da velocidade, portanto, responsável por balancear a busca realizada pelo algoritmo entre local e global. Quando o momento de inércia linearmente decrescente é adotado, normalmente se adota a equação (5) para a atualização de W, onde t_{max} é o número máximo de iterações e t é a iteração atual [16],[17]. O segundo fator é composto pelo componente de cognição c_1, multiplicado por um número gerado aleatoriamente com distribuição uniforme no intervalo [0,1], multiplicado pela diferença existente entre a posição atual da partícula e a melhor posição que a partícula já atingiu ao longo da execução do algoritmo ($pbest$). O último fator da equação é composto pela componente social c_2, multiplicado por um número aleatório entre "$0 \; e \; 1$", multiplicado pela diferença existente entre a posição atual da partícula e a melhor posição já atingida por qualquer partícula ao longo da execução do algoritmo ($gbest$). O limite para estes componentes é $c_1 + c_2 \leq 4$ [18],[19].

A equação (4) representa a atualização da posição da partícula, de acordo com a sua posição anterior e sua velocidade, levando em conta que $\Delta t = 1$ foi adotado [20],[21].

$$W = W_{max} - \frac{W_{max} - W_{min}}{t_{max}}.t \quad (5)$$

O pseudocódigo para o PSO tradicional é apresentado a seguir na Figura 2.

Criar e inicializar um PSO n -dimensional: P
***Repetir*:**
Para cada partícula (i) da população:
$f(X.y_i)$ = Avaliação da partícula X_i através da função objetivo f
se $f(X.y_i) < f\_pbest_i$ → Minimização
$f\_pbest_i = f(P.y_i)$
$pbest_i = X_i$
se $f\_pbest_i < f\_gbest$
$f\_gbest_i = f\_pbest_i$
$gbest = pbest_i$
Fim do Para
Executar atualizações do PSO em P utilizando as Eqs. de (3-4)
Até critério de parada ser satisfeito

Figura 2 - Pseudocódigo do algoritmo PSO.

Para o desenvolvimento do enxame cooperativo (CPSO) em vez de conter um enxame (de i partículas), tentando encontrar o vetor ótimo n-dimensional, o vetor é dividido em seus componentes de modo que os n enxames (de i partículas cada) estão otimizando um vetor $1 - D$, onde cada enxame representa uma dimensão do problema original. Neste sentido serão apresentados dois algoritmos cooperativos sugeridos por Bergh F. e Engelbrecht [4] sendo o CPSO-S_k e CPSO-H_k.

Para o CPSO-S_k realizar a minimização da f, requer um vetor n-dimensional como entrada. Assim cada enxame irá representar uma única dimensão do espaço de busca. Com isso o cálculo direto da função objetivo/*fitness* não pode ser efetuado. Para resolver este problema calcula-se separadamente cada um, assim cada enxame terá seu $gbest_j$, diferentemente do PSO que possuía apenas um $gbest$. Neste momento é criado um vetor de contexto para aglutinar todos os $gbest_i$, sendo necessário para prover um contexto adequado em que os indivíduos de uma população possam ser avaliados. Para calcular os *fitness* de todas as partículas i do enxame j, outros $n - 1$ componentes do vetor contexto são mantidos constantes (com seus valores definidos iguais aos da melhores partículas globais dos outros $n - 1$ enxames), enquanto o componente $j - ésimo$ do vetor contexto é substituído sucessivamente por cada partícula do $j - ésimo$ enxame. O algoritmo CPSO-S_k é normalmente capaz de resolver qualquer problema que o PSO padrão pode resolver. Existe a possibilidade o algoritmo ficar preso em um local onde todos os enxames não são capazes de descobrir melhores soluções. Este é um exemplo de estagnação, causado pela restrição de que apenas um enxame é atualizado por vez, ou seja, apenas um subespaço é procurado ao mesmo tempo [4],[22]-[25]. O funcionamento do CPSO-S_k pode ser observado na figura 3.

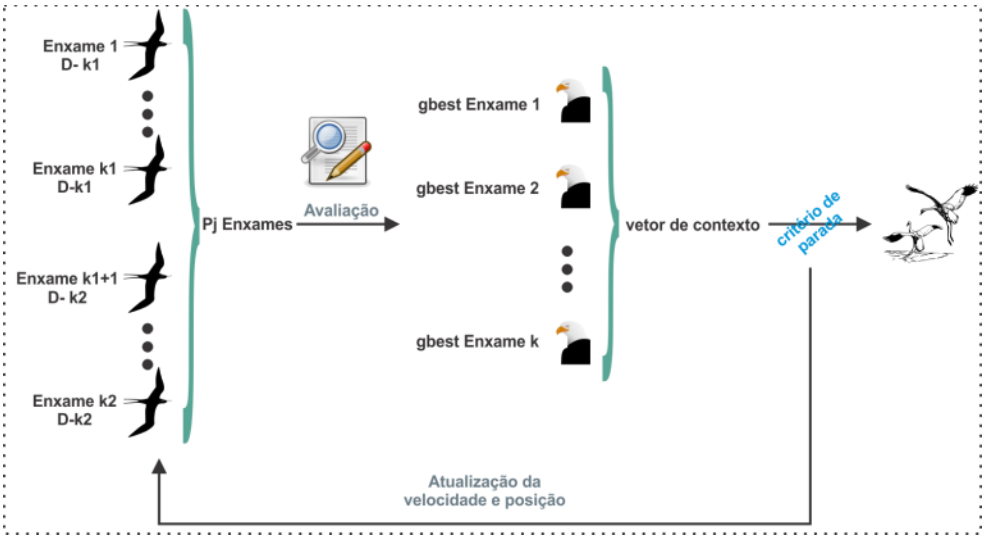

Figura 3 – Funcionamento do CPSO-S_k.

Neste momento observando a diferença entre o PSO e o CPSO-S_k levantou-se a possibilidade de unir os dois algoritmos realizando uma troca de informações sobre as melhores soluções encontradas por qualquer um dos enxames no final de cada iteração. Essa troca de informação entre o PSO e o CPSO-S_k é na forma de cooperação por quadro-negro (*black-board*), semelhante ao tipo descrito por Clearwater *et al.* [26].

Uma maneira simples de desenvolver este quadro-negro realizando a troca de informações é a substituição de algumas das partículas em uma metade do algoritmo com a melhor solução descoberta até o momento pela outra metade do algoritmo. Isto quer dizer que dentro do quadro-negro será utilizado o vetor de contexto do CPSO-S_k para substituir uma partícula escolhida aleatoriamente da metade do PSO representado pelo enxame Q, como pode ser observado na figura 4. Isso é seguido

por uma iteração do componente enxame Q do algoritmo, que produz uma nova partícula global melhor. Este vetor é então dividido em sub-vetores das corretas dimensões e utilizado para substituir as posições das partículas escolhidas aleatoriamente nos enxames. O funcionamento geral do CPSO-H_k pode ser observado na figura 4.

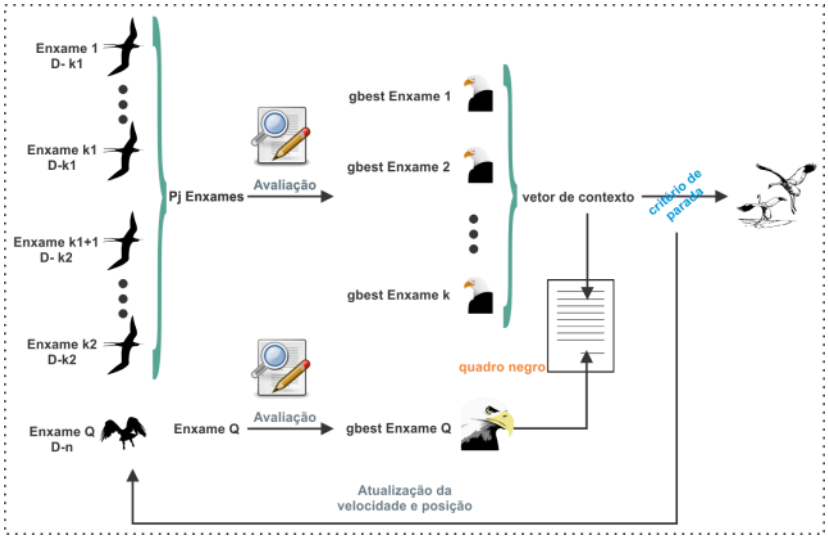

Figura 4 – Funcionamento do CPSO-Hk.

4 Paralelização do CPSO

A computação paralela e distribuída surgiu devido à necessidade de processar grandes quantidades de dados em pouco tempo, e também para possibilitar a solução de problemas complexos dividindo-os em pequenas tarefas. A computação evoluiu na última década para um modelo integrador de componentes de custo reduzido e de elevada capacidade, através do surgimento dos computadores com processadores *multi-core*.

A definição dada por Hwang e Zhiwei [27] define o processamento paralelo como uma forma eficiente de processamento de informações, em que se deve dar ênfase à exploração dos eventos concorrentes de um processo computacional. O conceito básico por trás do processamento paralelo é o de dividir uma tarefa em várias partes, sendo que essas serão distribuídas entre os vários processos. Esses, por sua vez, cooperam entre si por meio de comunicação e sincronismo, para realizar a tarefa melhorando o desempenho e reduzindo o tempo de processamento na obtenção do resultado final.

Para que o paralelismo apresente a eficiência desejada, ou seja, para que se possa implementar e executar um algoritmo ou um programa de forma paralela, em um sistema com múltiplos processadores (*cluster*), é necessário que três problemas fundamentais sejam resolvidos: (i) Identificação do paralelismo potencial; (ii) Particionamento do programa em tarefas sequenciais; (iii) Escalonamento das tarefas na execução concorrente. De acordo com Grit [28], além dos problemas fundamentais, são críticos para o desempenho dos sistemas paralelos: (i) a proporção entre a velocidade de comunicação e a capacidade de co'mputação; (ii) o custo da sobrecarga na criação das tarefas.

No caso dos algoritmos de PSO é possível identificar que o paralelismo potencial está no cálculo da função de avaliação/objetivo dos indivíduos dos enxames. Uma vez definido a estrutura dos enxames que serão paralelizados foi definido o modelo de paralelismo conhecido como mestre/escravo, *master/slave*, em que o processo mestre é responsável por distribuir os indivíduos dos enxames e controlar a execução das tarefas. Os escravos são responsáveis por executar o cálculo da função de avaliação, retornado esses resultados para o mestre [29],[30].

A Figura 5 representa o funcionamento do algoritmo CPSO-H_k. Nesta figura os indivíduos do enxame são distribuídos entre os processos escravos, representados pelos núcleos, *cores*, de um processador, que avaliam cada indivíduo e retornam os valores para o processo mestre que irá comparar e atualizar o vetor de contexto, definindo o PCPSO-H_k. Essa abordagem foi à mesma utilizada nos algoritmos do PSO clássico (PPSO) e do CPSO-S_k (PCPSO-S_k).

O computador utilizado para as simulações de todos os algoritmos foi um computador Intel Core I5 2,8GHz, com 4GB de RAM (*Random Access Memory*). Este processador possui 4 cores, assim as avaliações serão distribuídas para estes cores afim de equilibrar e acelerar a execução de cada algoritmo.

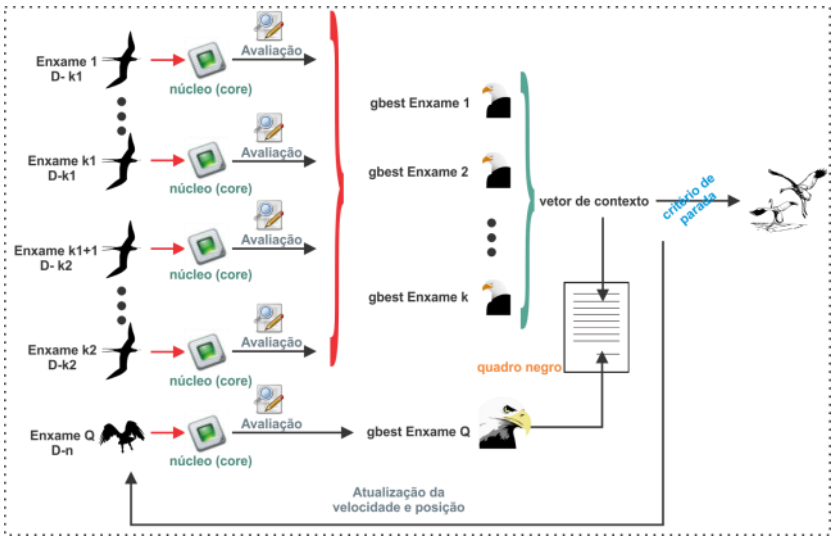

Figura 5 – Funcionamento do CPSO-Hk Paralelo.

5 Resultados

Para análise dos resultados foi desenvolvida a Tabela 1 onde são apresentados os resultados de 50 simulações de cada algoritmo sendo estes o PSO/PPSO (não paralelo e paralelo), CPSO-S_k/PCPSO-S_k (não paralelo e paralelo) e CPSO-H_k/PCPSO-H_k (não paralelo e paralelo) para a sequência de *benchmark* composta por 13 aminoácidos (ABBABBABABBAB). O tamanho do enxame foi definido sendo 1000, o número de gerações adotado foi 1000. Também são apresentados os gráficos das melhores simulações de cada algoritmo, conforme ilustram as Figuras de 6 a 8.

Tabela 1 – Resultados.

Algoritmo 50 simulações	**Número da melhor/pior simulação**	**Melhor**	**Pior**	**Média**	**Desvio padrão**	**Mediana**	**Tempo (segundos)**	**Número de avaliações**
PSO	*16/44*	*-3,198*	*-0,785*	*-1,967*	*0,431*	*-1,793*	*28540*	*1001000*
PPSO	*16/44*	*-3,199*	*-0,848*	*-2,083*	*0,465*	*-2,201*	*6255*	*1001000*
CPSO-S_k	*29/26*	*-3,293*	*-0,942*	*-2,245*	*0,742*	*-2,285*	*30780*	*6002000*
PCPSO-S_k	*29/26*	*-3,293*	*0,917*	*-2,033*	*0,999*	*-1,994*	*72956*	*6002000*
CPSO-H_k	*2/38*	*-3,294*	*-0,778*	*-2,458*	*0,686*	*-2,425*	*63465*	*7008001*
PCPSO-H_k	*2/38*	*-3,294*	*-1,144*	*-2,552*	*0,590*	*-2,430*	*52876*	*7008001*

Na Tabela 1 observando o PSO, este obteve um valor menor da função objetivo. Sua execução na forma paralela conseguiu melhorar expressivamente o desempenho, em questão do tempo, em 78,08%, e contém um número menor de avaliações. O gráfico da Figura 6 ilustra que este algoritmo estabilizou com praticamente o dobro de número de gerações.

Para o algoritmo CPSO-S_k o mesmo obteve sua melhor resposta em -3,293 e em sua forma paralelizada não melhorou sua execução nas 50 simulações realizadas por conta do número de avaliações executadas, do tempo de comunicação entre o mestre/escravo e sincronismo do algoritmo. Mesmo assim o algoritmo obteve uma boa resposta levando em consideração que o número de avaliação comparado com o CPSO-H_k é 14,35% menor.

Na Tabela 1 pode-se observar que o melhor valor atingindo na otimização do estudo de caso foi o CPSO-H_k e o PCPSO-H_k com o valor de -3,294. Vale ressaltar que estes dois algoritmos realizam um maior número de avaliações da função objetivo em relação às outras abordagens de PSO, neste estudo 7008001 avaliações foram realizadas, uma vez que o procedimento de busca deste algoritmo equivale a otimizar cada dimensão separadamente. Com a utilização de computação paralela o PCPSO-H_k reduziu o tempo de processamento em 16,68%, sendo que os tempos de processamento totais foram de 17,63 horas e 14,68 horas, assim a diferença foi de 2,95 horas.

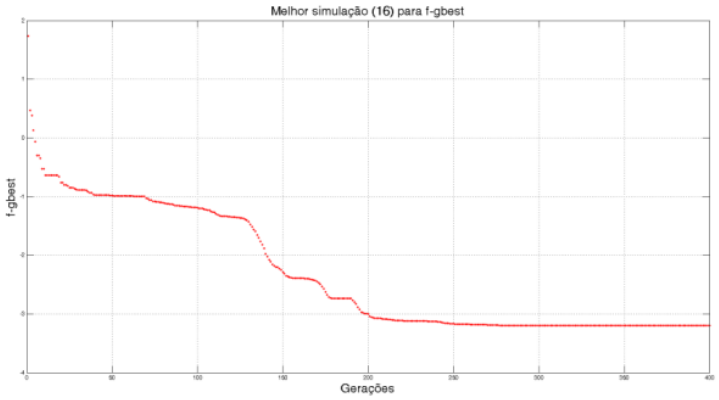

Figura 6 – Melhor PPSO (das 1000 gerações foram inseridas no gráfico até a geração 400).

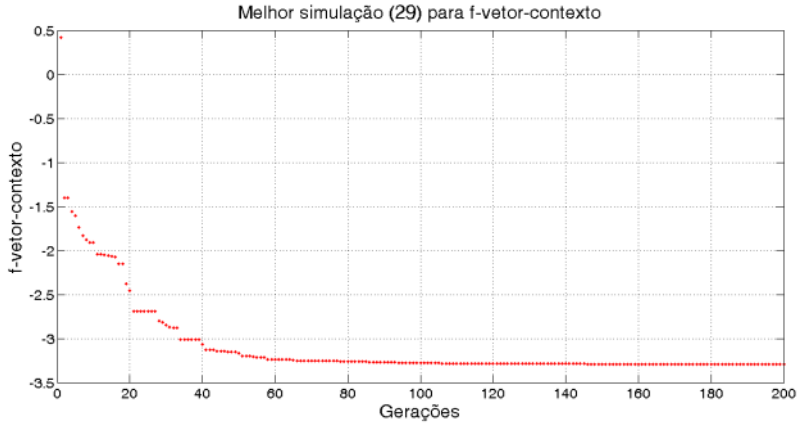

Figura 7 – Melhor CPSO-S_k (das 1000 gerações foram inseridas no gráfico até a geração 200).

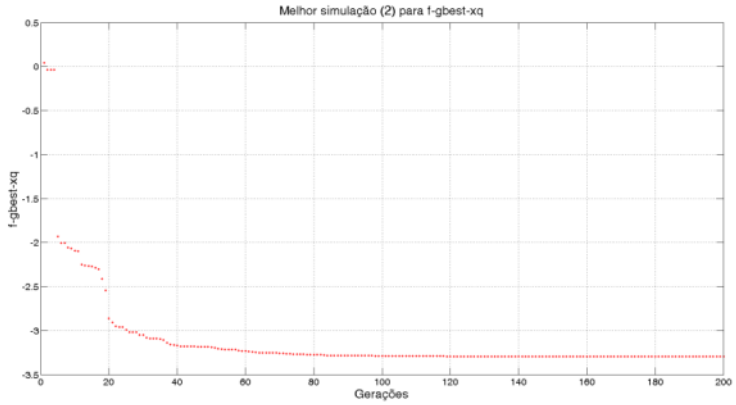

Figura 8 – Melhor PCPSO-H_k (das 1000 gerações foram inseridas no gráfico até a geração 200).

6 Conclusões

Este artigo apresentou uma metodologia na área de inteligência computacional focado no desenvolvimento de algoritmos de otimização por enxame de partículas aplicado ao estudo de caso da predição da estrutura de proteínas utilizando o modelo AB 2D. Foram desenvolvidos os algoritmos cooperativos e paralelos do PSO visando a melhora de resultados e agilidade do processamento dos mesmos. No desenvolvimento conseguiu-se utilizar as principais características dos algoritmos, sendo que no CPSO-S_k dividiu-se as dimensões do problema para enxames diferentes e no CPSO-H_k com a utilização do quadro-negro foi possível realizar trocas de informações entre os enxames.

Trabalhos futuros visarão a aplicação de uma estrutura computacional de cluster agregando maior desempenho de processamento na computação paralela. Ainda outros *benchmarks* de estrutura de proteínas conhecidos podem ser testados para avaliar a robustez e flexibilidade dos algoritmos. Para agregar outras funcionalidades pode-se utilizar a teoria de computação quântica criando algoritmos inspirados objetivando a melhora de desempenho e na qualidade da solução.

7 References

[1] R. C. Eberhart, J. Kennedy, A new optimizer using particles swarm theory, **Proceedings of Sixth International Symposium on Micro Machine and Human Science**, EUA, (1995) 39-43.

[2] J. Kennedy, R. C. Eberhart, Swarm Intelligence. EUA, **Morgan Kaufmann Publishers**, (2001).

[3] A. P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, University of Pretoria, South Africa, **John Wiley & Sons** Ltd, England, (2005).

[4] F. Bergh, A. P. Engelbrecht, A Cooperative Approach to Particle Swarm Optimization, **IEEE Transactions on Evolutionary Computation**, 8 (2004), 225-239.

[5] F. H. Stillinger, T. H.-Gordon, C. Hirshfeld, Toy model for protein folding, **Physical Review E**, 48 (1993) 1468-1477.

[6] F. H. Stillinger, T. H.-Gordon, C. Hirshfeld, Collective aspects of protein folding illustrated by a toy model, **Physical Review E**, 52 (1995), 2872-2877.

[7] N. Krasnogor, Protein structure prediction with evolutionary algorithms. In: Genetic and Evolutionary Computation Conference, **Morgan Kaufmann Publishers**, (1999), 1569-1601.

[8] H.S. Lopes, M. P. Scapin, An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic polar model, **Lecture Notes in Computer Science**, 3871 (2055), 238-246.

[9] M. P. Scapin, Um Algoritmo Genético Híbrido Aplicado à Predição da Estrutura de Proteínas utilizando o modelo Hidrofóbico-Polar dibimensional, Tese de Doutorado, **Universidade Tecnológica Federal do PR**, Curitiba, PR (2005).

[10] H. Hsu, V. Mehra, P. Grassberger, Structure optimization in an off-lattice protein model. **Physical R. E**, 68 (2003), 20–23.

[11] M. Bachmann, H. Arkm, W. Janke, Multicanonical study of coarse-grained off-lattice models for folding heteropolymers. **Physical Review E**, 71 (2005), 1-11.

[12] S. Chen, T. Mei, M. Luol, X. Yang, Identification of Nonlinear System Based on a New Hybrid Gradient-Based PSO Algorithm, **Proc. of the International Conf. on Information Acquisition**, Seogwip, SI, Coreia do Sul (2007) 265-268.

[13] C. Huang, and F. Wang, A RBF Network With OLS and EPSO Algorithms for Real-Time Power Dispatch, **IEEE Transactions on Power Systems**, 22 (2007), 96-104.

[14] V. G. Gudise, G. K. Venayagamoorthy, Evolving Digital Circuits Using Particle Swarm, **Proceedings of the International Joint Conference on Neural Networks**, (2003) 468-472.

[15] C. -L. Lin, S. -T. Hsieh, T. -Y. Sun, C. -C. Liu, Cluster Distance Factor Searching by Particle Swarm Optimization for Self-Growing Radial Basis Function Neural Network, **Proceedings of International Joint Conference on Neural Networks**, Vancouver, BC, Canada, (2006) 4825-4830.

[16] X. Xie, W. Zhang, Z. Yang, A Dissipative Particle Swarm Optimization, **Proceedings of the Congress on Evolutionary Computation**, Honolulu, HI, USA, 2 (2002) 1456-1461.

[17] K. E. Parsopoulos, M. N. Vrahatis, Recent Approaches to Global Optimization Problems through Particle Swarm Optimization, **Natural Computing**, 1 (2002) 235-306.

[18] Y. Shi, R. C. Eberhart, Parameter Selection in Particle Swarm Optimizer", **Proceedings of 7th Annual Conference on Evolutionary Programming**, San Diego, CA, USA, (1998) 591-601.

[19] J. -M. Xiao, X. -H. Wang, Nonlinear Neural Network Predictive Control for Power Unit Using Particle Swarm Optimization, **Proceedings of the 5th International Conference on Machine Learning and Cybernetics**, Dalian, China, (2006) 2851-2856.

[20] Z. -H. Zhan, J. Zhang, Y. Li, H. S. -H. Chung, Adaptive Particle Swarm Optimization, **IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics**, 39 (2009) 1-20.

[21] Y. Shi, R. C. Eberhart, Fuzzy Adaptive Particle Swarm Optimization, **Proceedings of the Congress on Evolutionary Computation**, Honolulu, HI, USA, 1 (2002) 101-106.

[22] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, **Parallel Problem Solving from Nature PPSN III In Proceedings of the Third Conference on Parallel Problem Solving from Nature**, Jerusalem, Israel ,(1994) 249–257.

[23] C. H. Tan, C. K. Goh, K. C. Tan, A. Tay, A cooperative coevolutionary algorithm for multiobjective particle swarm optimization, **IEEE Congress on Evolutionary Computation**, Singapore, (2007) 3180–3186.

[24] C. L. Su, Using Cooperative Particle swarm for Optimizing the Engineering Design Problems, **Proc. of the 9th WSEAS International Conference on Applications of Computer Engineering**, Stevens Point, WI, USA , (2010) 153-156.

[25] R. A. Krohling, L. S. Coelho, Y. Shi, Cooperative particle swarm optimization for robust control system design. **Proceedings of the 7th Online World Conference on Soft Computing in Industrial Applications**, (2002).

[26] S. H. Clearwater, T. Hogg, and B. A. Huberman, Cooperative problem solving, **Computation: The Micro and Macro View, Singapore: World Scientific**, (1992) 33–70.

[27] K. Hwang, X. Zhiwei, Scalable Parallel Computing: Technology, Architecture, Programming. **1st ed. New York: McGraw-Hill**, (1998) 832

[28] D. H. Grit, Sisal on a Message Passing Architecture. **Proceedings of Joint International Conference on Vector and Parallel Processing**. Berlin: Springer-Verlag, (1990) 721-721.

[29] S. C. Chu, J. F. Roddick, J. S. Pan, A Parallel Particle Swarm Optimization Algorithm with Communication Strategies, **Journal of Information Science and Engineering** 21 (2005) 809-818.

[30] D. Gies, Y. Rahmat-Samii, Reconfigurable array design using parallel particle swarm optimization. **Proceedings of the IEEE International Conference on Antennas and Propagation**, Columbus, OH, USA (2003) 177-180.