Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXI, págs. 925-943 (artículo completo)
Alberto Cardona, Paul H. Kohan, Ricardo D. Quinteros, Mario A. Storti (Eds.)
Salta, Argentina, 13-16 Noviembre 2012

COMPUTAÇÃO PARALELA E SERIAL APLICADAS À OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS EM PROBLEMAS DE ENGENHARIA

Adelano Esposito[a], Herbert M. Gomes[b] e Letícia Fleck Fadel Miguel[c]

[a] *Programa de Pós-Graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, R. Sarmento Leite, 425, sala 202, 2º. Andar, 90050-170, Porto Alegre, RS, Brasil, adeesp_@hotmail.com, http://www.mecanica.ufrgs.br/promec/*

[b] *Programa de Pós-Graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, R. Sarmento Leite, 425, sala 202, 2º. Andar, 90050-170, Porto Alegre, RS, Brasil, herbert@mecanica.ufrgs.br, http://www.mecanica.ufrgs.br/promec/*

[c] *Programa de Pós-Graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, R. Sarmento Leite, 425, sala 202, 2º. Andar, 90050-170, Porto Alegre, RS, Brasil, letffm@ufrgs.br, http://www.mecanica.ufrgs.br/promec/*

Palavras-chave: otimização por enxame de partículas, computação paralela, computação serial, algoritmos heurísticos.

Resumo. Um dos métodos heurísticos bastante explorados em engenharia é o PSO (*Particle Swarm Optimization*). O PSO é uma metaheurística baseada em populações de indivíduos, na qual candidatos à solução evoluem através da simulação de um modelo simplificado de adaptação social. Este método vem conquistando grande popularidade, no entanto, o elevado número de avaliações da função objetivo limita a sua aplicação em problemas de grande porte de engenharia, tornando a computação paralela uma alternativa atraente para sua utilização. Neste trabalho, são desenvolvidas duas versões paralelas do PSO original em multiprocessadores, utilizando funções disponíveis na biblioteca do *MATLAB*®, e também uma versão serial assíncrona. Os algoritmos diferem entre si pelo modo de comunicação entre os processadores (síncrono ou assíncrono) e pela forma de atualização das partículas do enxame (imediata ou por revoada). Os modelos propostos foram aplicados em problemas de engenharia conhecidos na literatura (*benchmarks*) e seus resultados são comparados, em termos de performance e acurácia. Em geral, a acurácia se manteve, salvo que no algoritmo assíncrono (na forma serial e paralela), houve um ganho de performance significante.

1 INTRODUÇÃO

Otimização por enxame de partículas (*Particle Swarm Optimization* – PSO) é um método heurístico muito utilizado em problemas das mais diversas áreas, dentre elas a engenharia estrutural. Similar a outros métodos de otimização evolutiva, o PSO é um algoritmo que utiliza uma população, chamada de partículas, que voam por um espaço de busca com direção e velocidade, influenciadas dinamicamente pelo grupo e histórico das partículas, voltadas a encontrar melhores soluções. Embora os algoritmos PSOs apresentem diversas propriedades atraentes, eles perdem espaço para outros métodos determinísticos em função de seu moderado desempenho. Entretanto, com os avanços na tecnologia dos computadores, vários algoritmos de otimização estão sendo implementados em formas paralelas, e em sua maioria em versões síncronas. O algoritmo de otimização síncrono requer um ponto de sincronização no final de cada iteração antes de continuar para a próxima. Essa abordagem resulta em ganhos em termos de eficiência, mas não melhora a performance do método. Este artigo apresenta as versões seriais e paralelas do algoritmo sequencial por enxame de partícula e como alternativa para diminuir ainda mais o tempo computacional, melhorando a performance do PSO, é lançada uma nova abordagem assíncrona e paralela. Para comparar e provar os ganhos obtidos em termos de performance, os métodos são submetidos a um exaustivo experimento que representa os problemas de otimização de estruturas metálicas.

2 MODELOS PARALELOS DE PSO

Os altos desempenhos computacionais dos modernos computadores somados às atuais pesquisas realizadas com métodos determinísticos, heurísticos e híbridos buscam constantemente melhorar o tempo de processamento de inúmeros problemas de engenharia, no entanto, alguns destes permanecem computacionalmente custosos. Segundo Yang (2010), não há um método universal que forneça garantidamente a solução ótima com o melhor desempenho para todos os problemas de otimização global. O autor afirma que o desempenho independe do algoritmo, ou seja, todos os algoritmos de otimização apresentarão desempenho similar se considerar a média do desempenho dos algoritmos aplicados a vários problemas de otimização diferentes.

É importante ressaltar que um algoritmo pode apresentar melhor performance para um determinado problema de otimização, entretanto, não significa que para funções diferentes seu desempenho continue sendo melhor do que os demais algoritmos. Isso dá indícios de que não existe um método universal que garanta um melhor desempenho sob todos os problemas de otimização existentes.

Sendo assim, inspirados em encontrar uma forma de melhorar o desempenho destes algoritmos, pesquisadores de diversas áreas veem desenvolvendo modelos paralelos aos algoritmos seriais desenvolvidos.

Um dos métodos heurísticos frequentemente explorados na computação paralela são os AGs (AGP) Cantú-Paz (2000), apresentando-se basicamente em três formas paralelas: Mestre-Escravo, onde apenas a função de aptidão *fitness* é distribuída entre os processadores; Ilhas (malha grossa), onde sub-populações evoluem paralelamente trocando informações através de migrações, segundo uma dada estratégia; Celular (malha fina), onde cada candidato (indivíduo) a solução é alocado em um processador/processo (célula) e geralmente seguem uma arquitetura de malha 2D, interligada em forma de toroide.

Outro método que vem conquistando grande popularidade devido a sua robustez, eficiência e simplicidade, é o PSO, entretanto, como as demais heurísticas, quando deparado com problemas grandes e complexos, sua eficiência pode reduzir drasticamente. Esse comportamento se deve em questão ao aumento na quantidade de avaliações da função de

aptidão ("*fitness*" forma pela qual o enxame evolui em direção ao ótimo) causado pela dificuldade do algoritmo de encontrar o ótimo. Por outro lado, o PSO pode ser facilmente paralelizado, o que torna a computação paralela uma alternativa atraente e dependendo do problema, indispensável para utilização deste método.

O primeiro trabalho de paralelização de PSO foi desenvolvido por Shutte *et al.* (2004), propondo uma implementação síncrona com o modelo mestre-escravo. Desde então, arquiteturas paralelas do PSO têm sido largamente empregadas em problemas de otimização. Aplicações deste tipo são encontradas nos trabalhos de Belaland El-Ghazawi (2004), Venter (2005), Koh *et al.* (2006), Mostaghim *et al.* (2007) entre outros.

A ideia básica da maioria dos programas paralelos, como pode ser verificada na Figura 1 (b), consiste em dividir um problema em vários problemas menores (subproblemas). Esses subproblemas são distribuídos entre os vários processadores disponíveis e executados simultaneamente. Ao final da computação, cada subproblema prevê um resultado, que é combinado com os demais resultados obtidos, gerando a solução para o problema tratado inicialmente. Diferentemente da forma serial na qual uma única instrução é executada após a outra em um dado instante de tempo, como pode ser verificado na Figura 1 (a).

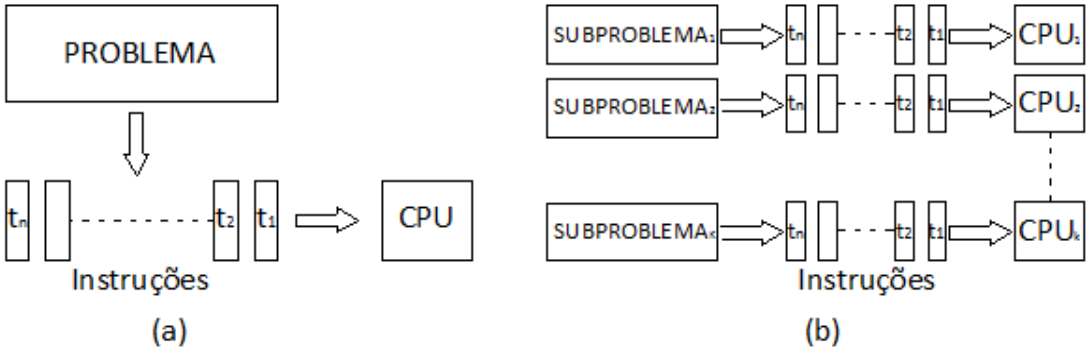

Figura 1: (a) Processamento sequencial, (b) Processamento paralelo

Trazendo o conceito de processamento paralelo ao contexto do algoritmo PSO, de um modo geral, a paralelização do PSO consiste em distribuir as partículas de um enxame entre os processadores disponíveis, os quais são responsáveis por executar uma determinada tarefa com a respectiva partícula e posteriormente devolvê-la ao enxame.

A topologia ou estrutura de comunicação destas partículas caracterizam o modelo de programação adotado para paralelizar o PSO.

Os modelos de programação paralela do PSO podem ter diversas classificações, estas divergem de autor para autor. A classificação feita neste trabalho, estabelece dois modelos principais de implementação do PSO em paralelo, o modelo de paralelização global e modelo de múltiplas populações.

Os modelos de paralelização global geralmente são implementados em algoritmos com a topologia do tipo mestre-escravo (Figura 2 (a)). Nesses algoritmos, existe um processador principal (ou "mestre") que controla a evolução do enxame. Os demais processadores secundários (ou "escravos") recebem frações do enxame e executam as operações pertinentes ao algoritmo. Já no modelo de múltiplas populações, também chamado de múltiplos enxames, a implementação dos algoritmos ocorre em uma topologia em forma de estrela (Figura 2 (b)). Nesta topologia cada um dos enxames seleciona a sua melhor partícula e a envia para cada um dos demais enxames. Dessa forma as piores partículas de cada enxame são substituídas pela melhor partícula de cada um dos demais enxames, evitando a perda de tempo com uma subpopulação cujo resultado seja ruim, pois esta será melhorada com a chegada de partículas

dos enxames vizinhos.

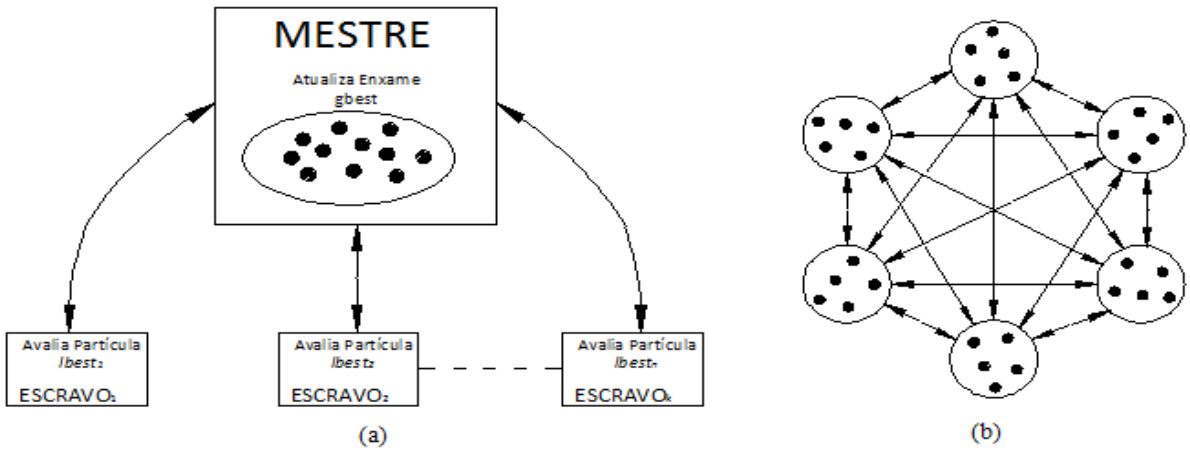

Figura 2: (a) Topologia mestre-escravo, (b) Topologia de comunicação em estrela

3 ALGORITMO DE OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS (PSO)

O algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO) foi inicialmente proposto por Kennedy e Eberhart (1995), como sendo uma técnica inspirada no comportamento social de bandos de pássaros. A busca por alimento e a interação entre os pássaros ao longo do voo são modeladas como um mecanismo de otimização. Fazendo uma analogia, o termo partícula foi adotado para simbolizar os pássaros e representar as possíveis soluções do problema a ser resolvido. A topologia ou região percorrida pelos pássaros é equivalente ao espaço de busca e encontrar o local com comida corresponde a encontrar a solução ótima. Para que as partículas sempre se aproximem do ponto ótimo, ao invés de se perderem e nunca alcançá-lo quando estiverem percorrendo a função objetivo, utiliza-se a função de aptidão (*fitness*), responsável por avaliar o desempenho de cada partícula e informar o quanto ela está apta a encontrar a melhor posição. O termo que indica a experiência ou conhecimento individual de cada partícula, representada pela melhor solução encontrada individualmente pela mesma e gravada para uso posterior é o *lbest*. O responsável por representar o conhecimento do enxame como um todo e a melhor solução encontrada até então por algum dos indivíduos que formam o enxame é representado na literatura por *gbest*.

Ao contexto do algoritmo de otimização do PSO, dois parâmetros guiam a direção de procura que cada partícula apresenta ao longo do processo iterativo: a posição atual (vetor) da partícula e a sua velocidade. Inicialmente esses parâmetros são especificados de forma aleatória, entretanto, ao longo do processo, novas posições e velocidades são atribuídas às partículas, as quais são influenciadas por três "forças" representadas matematicamente em forma de vetor: inércia, memória e cooperação. De forma simplificada, a atualização da velocidade da partícula i no tempo t ($v_{i,j}(t+1)$) no espaço de busca n dimensional, leva em conta a sua distância atual para a melhor posição já encontrada pela mesma partícula ($c_1 r_1 \left(xlbest_{i,j}(t) - x_{i,j}(t)\right)$) e a distância da sua posição atual à melhor posição encontrada pelo enxame ($c_2 r_2 \left(xgbest_j(t) - x_{i,j}(t)\right)$). Estes vetores são ponderados pelos fatores $c_1 r_1$ e $c_2 r_2$, os quais serão descritos a seguir. De posse dessas informações da partícula e do enxame, o algoritmo do PSO utiliza as seguintes regras de atualização das posições e velocidades dispostas na equação (1).

$$v_{i,j}(t+1) = \chi\left(wv_{i,j}(t) + c_1 r_1\left(xlbest_{i,j}(t) - x_{i,j}(t)\right) + c_2 r_2\left(xgbest_j(t) - x_{i,j}(t)\right)\right)$$
$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (1)$$
$$\chi = \frac{2}{\left|2 - (c_1 + c_2) - \sqrt{(c_1 + c_2)^2 - 4(c_1 + c_2)}\right|}$$

Onde:

- $v_{i,j}(t+1)$: é a velocidade atualizada da partícula *i* correspondente a variável de projeto *j*;
- $v_{i,j}(t)$: é a velocidade atual da partícula *i* correspondente a variável de projeto *j*;
- $x_{i,j}(t+1)$: é a posição atualizada da partícula *i* correspondente a variável de projeto *j*;
- $x_{i,j}(t)$: é a posição atual da partícula *i* correspondente a variável de projeto *j*;
- $xlbest_{i,j}(t)$: é a melhor posição já encontrada pela partícula *i* correspondente a variável de projeto *j*;
- $xgbest_j(t)$: é a melhor posição já encontrada pelo enxame de partículas correspondente a variável de projeto *j*;
- c_1: constante de aceleração cognitiva (individual), referente à $xlbest_{i,j}(t)$;
- c_2: constante de aceleração social (enxame), referente à $xgbest_j(t)$;
- r_1e r_2: números aleatórios que podem estar entre zero e um;
- χ: fator de constrição;
- w: peso de inércia.

As características supracitadas quanto ao procedimento de atualização da velocidade e posição da partícula são melhor compreendidas na forma de vetores na Figura 3.

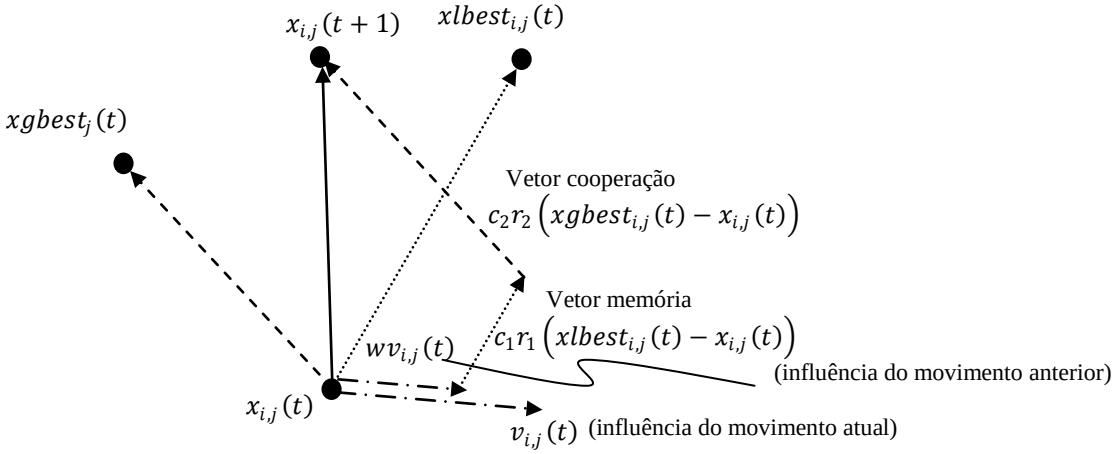

Figura 3: Representação vetorial da atualização da velocidade e posição do PSO

FONTE: adaptado de Hassan *et al*. (2005)

Na implementação do algoritmo PSO, inicialmente as posições e velocidades das partículas são geradas de maneira aleatória. Posteriormente as partículas entram no laço do algoritmo responsável pelas atualizações, segundo a equação (1) até atingirem um critério de parada. Dentro desse laço, uma questão muito importante deve ser levada em consideração quanto à atualização da posição e velocidade das partículas, trata-se da abordagem síncrona e assíncrona.

No modelo de implementação síncrono do PSO (Figura 4 (a)), representado no presente trabalho pelas siglas SSPSO (sequencial síncrono PSO), enquanto um critério de parada ou convergência não é atingido, uma nova *revoada* (ou iteração) é reinicializada e a posição e velocidade de cada partícula são atualizadas de acordo com os melhores valores das partículas e do enxame, previamente calculados. Em outras palavras, pode-se dizer que, primeiramente todas as partículas do enxame devem ser avaliadas para só então, (não satisfeito a condição de parada) atualizarem suas posições e velocidades. Já na implementação assíncrona do PSO (Figura 4 (b)), atualização de cada partícula é realizada logo após sua avaliação, ou conforme Koh *et al.* (2005), em algoritmos PSO assíncronos, a atualização da velocidade e posição da partícula ocorre continuamente com base em dados atualmente disponíveis. Assim, uma partícula que antes (forma síncrona) era alterada apenas após a *revoada* do enxame, agora é alterada a cada *pseudo-revoada.* Possibilita ao algoritmo que ao encontrar um novo ótimo global (*xgbest*), ele é imediatamente ajustado e disponibilizado (informado) às partículas ainda não atualizadas naquela iteração. O código assíncrono do respectivo artigo é representado pelas siglas SAPSO (sequencial assíncrono PSO).

```
Inicializando Algoritmo
    Informando constantes do algoritmo
    Gerando posição e velocidade das partículas
    randomicamente
Inicio da Otimização
    Para k=1, número de iterações
        Para i=1, número de partículas
            Avaliar função f(x_k^i)
        Fim
        Testa convergência
        Atualiza posição e velocidade
    Fim
Apresenta Resultados
```

```
Inicializando Algoritmo
    Informando constantes do algoritmo
    Gerando posição e velocidade das partículas
    randomicamente
Inicio da Otimização
    Para k=1, numero de iterações
        Para i=1, número de partículas
            Avaliar função f(x_k^i)
            Testa convergência
            Atualiza posição e velocidade
        Fim
    Fim
Apresenta Resultados
```

Figura 4: (a) Pseudocódigo PSO síncrono, (b) Pseudocódigo PSO assíncrono

4 IMPLEMENTAÇÃO PARALELA

Conforme citado anteriormente, uma forma de melhorar o desempenho dos algoritmos sequências consiste em adaptá-los a versões paralelas. Neste contexto, os algoritmos SSPSO e SAPSO são explorados a fim de verificar as dependências entre as etapas que compõem os respectivos códigos e assim particioná-los em grupos que possam processar em paralelo. Essa fase permite expor as oportunidades de paralelização dos códigos de modo que, a ordenação da execução das tarefas durante o processamento do algoritmo seja mantida.

A implementação paralela do modelo SSPSO, cuja denominação adotada pode ser representada pelas siglas PSPSO (Figura 6 (a)), consiste em avaliar as partículas dentro de uma iteração *k* em paralelo, sem alterar a lógica global do algoritmo em si. Nesta implementação, todas as partículas dentro de uma *revoada* são enviadas para o ambiente de computação paralela, e o algoritmo deve esperar que todas as partículas sejam analisadas antes de prosseguir para a próxima etapa (atualização). Esta característica torna evidente um ponto fraco do método, que é a dependência entre as etapas do algoritmo, ou seja, o processador mestre só pode prosseguir para uma nova etapa (atualização do *xlbest*, *xgbest* e velocidade) após receber as informações de todos os processadores escravos. Sendo assim, embora a implementação síncrona forneça uma extensão fácil para um ambiente em paralelo, não é uma solução ideal e geralmente resulta em baixa eficiência. Para contornar esses

problemas, foi implementado o código paralelo assíncrono do PSO, assim denominado por PAPSO (Figura 6 (b)), para a comunicação de dados entre os processadores. Diferente do PSPSO, na abordagem paralela assíncrona o processador mestre é responsável apenas pela verificação quanto a convergência do método, as demais etapas do processo ocorrem isoladamente em cada processador (as tarefas desempenhadas pelos processadores mestre/escravos tanto para o PSPSO quanto para o PAPSO, podem ser verificadas na Figura 5(a) e 5(b) respectivamente). Também, nessa configuração, manteve-se a característica de *pseudo-revoada* do algoritmo serial assíncrono, pois assim que as partículas, presentes nos processadores escravos disponíveis (N_p), receberem sua nota, esta é repassada (pela topologia de comunicação global) entre os N_p, os quais selecionam (isoladamente) a melhor partícula daquela rodada. Essa característica do PAPSO permite ao processador mestre não precisar aguardar o término da avaliação das $n_{part\,ículas}$ nos processadores escravos para continuar na etapa seguinte, assim, a dependência entre os processos é destruída. Além disso, o fato da paralelização do algoritmo manter a característica de *pseudo-revoada* da forma sequencial assíncrona, tende a melhorar ainda mais seu desempenho.

- Processador Mestre

1. Inicializa todos os parâmetros de otimização, posições e velocidades iniciais das partículas;
2. Atualiza posição e velocidade das partículas;
3. Mantém uma fila de partículas (ou seja: posição das partículas) para os processadores escravos avaliarem;
4. Envia a posição da partícula seguinte da fila para um processador escravo disponível;
5. Recebe o valor da função *fitness* e o guarda numa fila até que todas as partículas sejam avaliadas;
6. Atualiza *xlbest*, *xgbest* e seleciona a melhor partícula do enxame;
7. Verifica convergência;

- Processadores Escravos

1. Recebe a posição da partícula do processador mestre;
2. Analisa a posição da partícula com a função *fitness*;
3. Envia o valor da *fitness* para o processador mestre.

(a)

- Processador Mestre

1. Inicializa todos os parâmetros de otimização, posições e velocidades iniciais das partículas;
2. Envia uma fila de partículas para os processadores escravos;
3. Recebe as informações da melhor partícula do enxame;
4. Verifica convergência;

- Processadores Escravos

1. Cada processador escravo recebe e mantém uma parcela da fila de partículas para serem avaliadas;
1. Atualiza posição e velocidade da partícula com base em informações atualmente disponíveis (*xlbest*, *xgbest*);
2. Analisa a posição da partícula com a função *fitness*;
3. Realiza a troca de informações entre os demais processadores;
4. Atualiza *xlbest*, *xgbest* e seleciona a melhor partícula do enxame;
5. Mantém uma fila com as melhores partículas do enxame até que todas as partículas sejam avaliadas;
6. Envia as informações da melhor partícula do enxame.

(b)

Figura 5: (a) Tarefas desempenhadas pelo PSPSO, (b) Tarefas desempenhadas pelo PAPSO

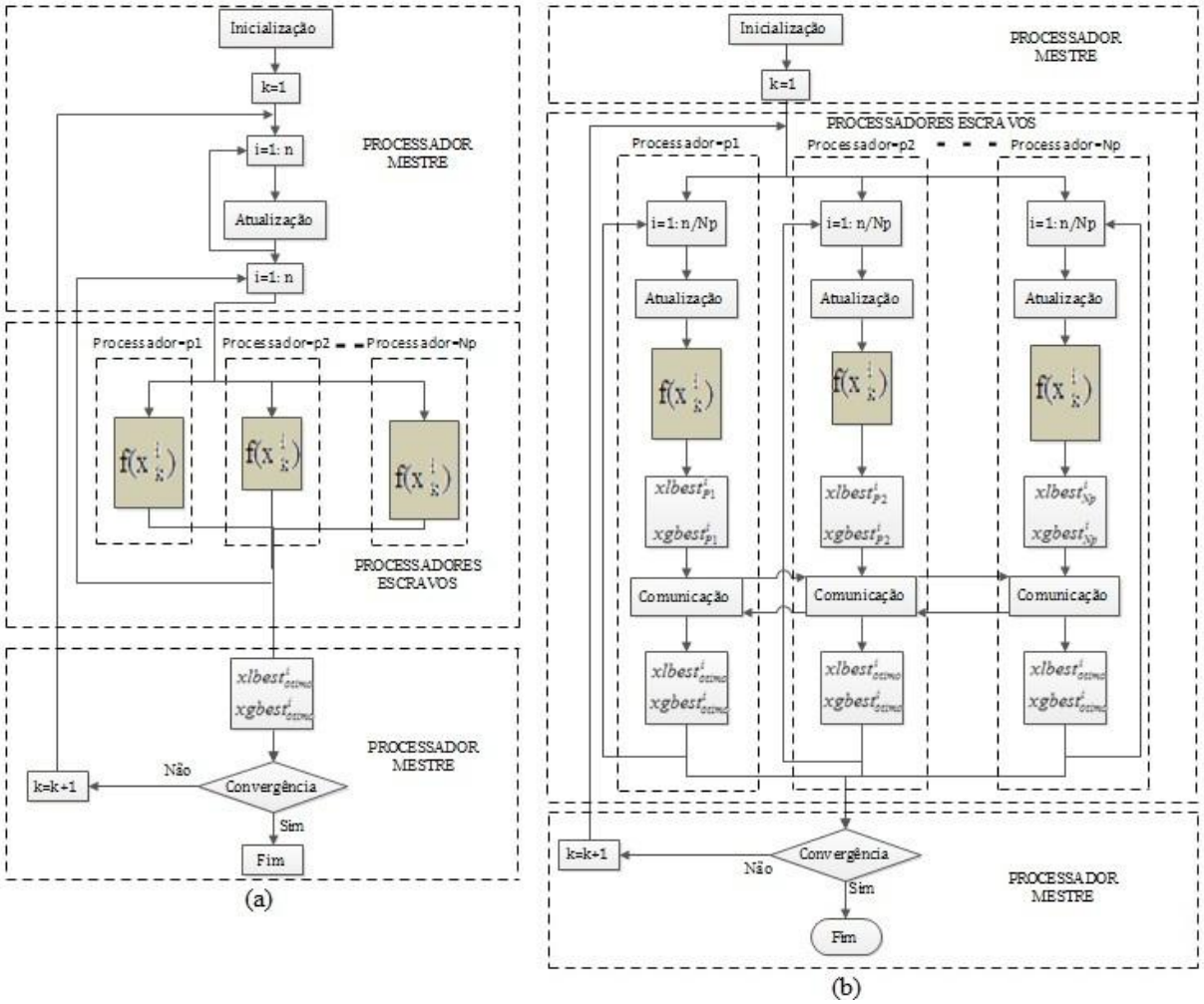

Figura 6: (a) Fluxograma do PSPSO, (b) Fluxograma do PAPSO

Considerando a ordem de atualização das partículas verificada nos algoritmos paralelos desenvolvidos neste trabalho, é possível perceber que a implementação paralela aplicada às versões seriais deve afetar o resultado final obtido (comparar as soluções que serão apresentadas nas Tabelas 4 e 7), mesmo quando uma mesma semente de números aleatórios for utilizada. A justificativa para a não coerência entre as soluções obtidas com o algoritmo serial e paralelo pode ser tratada da seguinte forma: como a ordem de retorno das partículas depende do tempo que os processadores escravos levam para calcular a função objetivo, as partículas podem não pertencer, a princípio, à mesma revoada, sendo assim, elas podem tomar caminhos diferentes, a depender da ordem de retorno das partículas pelos escravos.

5 IMPLEMENTAÇÃO DOS CÓDIGOS

O ambiente de desenvolvimento escolhido para a implementação dos programas sequenciais e paralelos é o *MATLAB*®, devido à facilidade para programação da lógica envolvida nos algoritmos (há um conjunto de funções para determinação de parâmetros necessários ao algoritmo utilizado) e permitir explorar o paralelismo das máquinas que vão hospedar o processamento. Nesse contexto, um fator a ser considerado, para transformar um algoritmo serial num algoritmo paralelo na linguagem *MATLAB*®, consiste em utilizar funções que o paralelizem concorrentemente. Sendo assim, dentre outras possibilidades, uma das

bibliotecas que contém as funções de interesse para esse trabalho é a *Parallel Computing Toolbox* (PCT) do próprio *MATLAB*®. Essa biblioteca, de propriedade da *Mathworks*, possibilita que o usuário execute uma tarefa em paralelo no ambiente de trabalho de uma máquina, usando até 8 processos para auxiliar o programa principal. Algumas de suas funções, as quais são de interesse do presente trabalho, são o SPMD e o PARFOR. Para que o código acesse estas funções, é necessário realizar a entrada no laço *matlabpool*, essa operação demanda um tempo de execução notadamente competitivo comparado com outras bibliotecas (por exemplo: MATLABMPI, PMATLAB e etc.). Portanto, a utilização da biblioteca PCT é indicada para processamento de grandes quantidades de dados, do contrário, a paralelização com o PCT possivelmente não apresente grandes vantagens.

Funções do tipo SPMD permitem que sejam definidos blocos de códigos executados simultaneamente em vários processadores comunicando-se entre si, ou seja, um programa é executado no processador mestre e partes dele, definidas como blocos SPMD, são executadas nos processadores escravos com a possibilidade de comunicação entre eles. Quando todos os blocos SPMD estiverem concluídos, o programa continua a ser executado no processador mestre. A Figura 7 (a) apresenta a forma de comunicação do SPMD, a qual se enquadra na topologia em estrela.

As funções presentes no comando PARFOR possibilitam ao algoritmo realizar a paralelização próxima à que ocorre utilizando o comando SPMD, exceto quanto à comunicação entre os processadores, ou seja, o PARFOR não permite que os processadores se comuniquem entre si. A Figura 7 (b) ilustra a lógica envolvida durante a execução do PARFOR. Nela, verifica-se que um programa, executado no processador mestre, é repartido em blocos PARFOR e distribuído entre os processadores escravos. Quando todos os blocos PARFOR estiverem concluídos, o programa continua a ser executado no processador mestre. Cabe salientar que neste caso não pode haver comunicação entre os processadores escravos, devendo ser as execuções completamente independentes entre si. A forma de comunicação utilizada pelo comando PARFOR, como pode ser verificada pela Figura 6 (b), corresponde à topologia mestre-escravo.

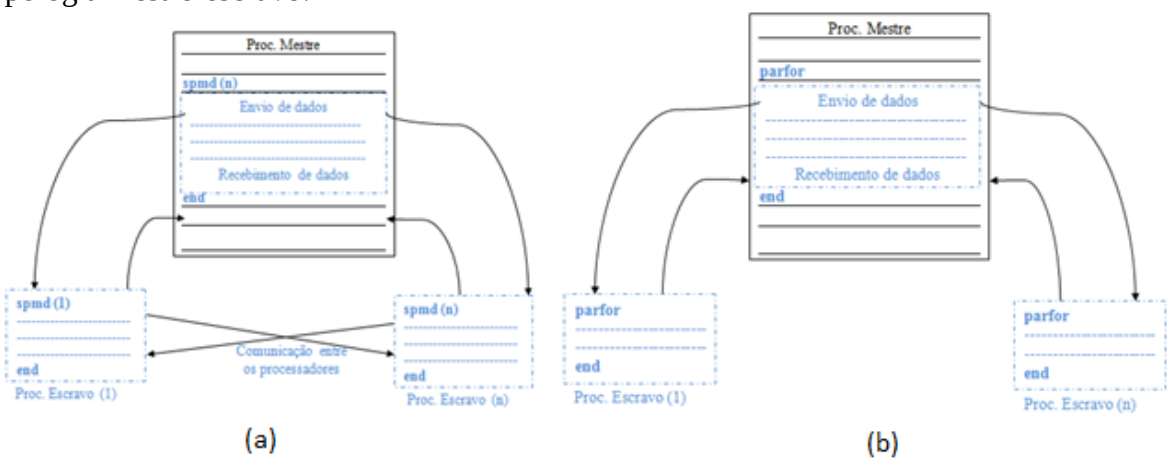

Figura 7: (a) Etapas do processamento SPMD, (b) Etapas do processamento PARFOR

As características supracitadas permitem adequar o PAPSO à paralelização com o SPMD e o PSPSO à paralelização com o PARFOR.

A paralelização do PSPSO, utilizando o comando PARFOR, consiste em realizar várias tarefas em múltiplos processadores simultaneamente. Neste contexto, uma tarefa corresponde na avaliação de uma determinada partícula atribuindo-a uma nota, no algoritmo PSO esta etapa representa o cálculo da função de aptidão (*fitness*). Uma característica atraente do

PARFOR consiste em manter a carga de trabalho uniforme entre os processadores escravos. Isso só é possível graças à independência entre o tempo de resposta dos $N_{escravos}$, ou seja, assim que concluído o trabalho em um processador escravo, este pode solicitar nova tarefa ao mestre, independente dos demais processadores terem finalizados suas tarefas.

Quanto à paralelização do PAPSO com o SPMD, esta consiste em distribuir as $n_{part\,ículas}$ entre os processadores disponíveis. Isso é atingido dividindo as $n_{part\,ículas}$ entre os N_p, alocando uma quantia de partículas igual para cada processador. Este esquema permite que a estrutura original do algoritmo sequencial assíncrono seja mantida, salvo que, a ordem de execução das tarefas (ou ordem de avaliação das partículas) é alterada a cada *pseudo-revoada*, ou seja, assim que os N_p concluírem o primeiro lote de partículas, é estabelecida uma nova sequência de distribuição das partículas entre os N_p.

6 MÉTRICAS PARA AVALIAÇÃO DOS ALGORITMOS

Dentre as métricas existentes para se avaliar um algoritmo pode-se citar: acurácia, robustez e desempenho. A acurácia (μ), pode ser definida como a média das distâncias obtida entre a posição encontrada pela melhor partícula do enxame ($\mathrm{xgbest_i}$), após satisfeita a condição de parada, e a posição ótima conhecida do problema ($\mathrm{x^*}$). Essa métrica é implementada no algoritmo de acordo com a equação (2).

$$\mu = \frac{1}{\mathrm{r}}\sum_{\mathrm{i}=1}^{\mathrm{r}} \left\| \overrightarrow{\mathrm{xgbest_i}} - \vec{\mathrm{x}}^* \right\| \tag{2}$$

A robustez (σ) relaciona quantas vezes o algoritmo reinicializou a procura e quantas destas vezes ele atingiu o mesmo local, que não necessariamente é o ótimo global, sob condições diferentes de início. Vale ressaltar que para cada reinicialização (r) do algoritmo, uma semente diferente de números aleatórios é utilizada para iniciar o enxame. A equação (3) apresenta a formulação adotada para a obtenção da robustez, onde $\vec{\mu}_{\mathrm{xgbest}}$ corresponde ao vetor que contém a média das melhores posições do enxame ($\mathrm{xgbest_i}$):

$$\sigma = \sqrt{\frac{\sum_{\mathrm{i}=1}^{\mathrm{r}} \left(\left\| \overrightarrow{\mathrm{xgbest_i}} - \vec{\mu}_{\mathrm{xgbest}} \right\| \right)^2}{\mathrm{r}-1}} \tag{3}$$

A performance pode ser definida como o número médio de avaliações da função objetivo, necessárias para resolver o problema. Sendo assim, quanto menor for o número de avaliações da função objetivo, melhor será a performance do método.

7 PROBLEMAS DE OTIMIZAÇÃO ESTRUTURAL

Os algoritmos propostos no presente artigo são testados em problemas clássicos de otimização de estruturas treliçadas (*benchmarks*), permitindo assim, demonstrar a validade e desempenho da metodologia proposta na otimização estrutural. Esses exemplos têm sido estudados por diversos pesquisadores, tais como Haftka e Gürdal (1992), Rajeev e Krishnamoorthy (1992), Kripka (2004), entre outros. As treliças testadas foram as de 18 barras e 72 barras, cujo objetivo consiste na minimização da massa respeitando as restrições estabelecidas para cada treliça. As variáveis de projeto são tratadas como variáveis contínuas. Sendo assim, a formulação do problema para encontrar a massa mínima das treliças dos exemplos pode ser representada matematicamente pela equação (4).

$$\begin{aligned} &\text{Minimizar:} && \text{Função Objetivo:} && Massa(A) = \rho \sum_{i=1}^{mbarras} A_i L_i \\ &&& \text{Variáveis de Projeto:} && A_i \end{aligned} \tag{4}$$

Sujeito a: Restrições:

$$g_j(A) = \omega_j \leq \overline{\omega}_j \qquad j=1,2,..., n_\omega$$
$$g_j(A) = \sigma_j \leq \sigma_j^T \qquad j=1,2,..., n_\sigma$$
$$g_j(A) = \sigma_j^F \leq \sigma_j \qquad j=1, 2,.., n_\sigma$$
$$\bar{A}_i^{min} \leq A_i \leq \bar{A}_i^{max} \qquad i=1,2,..., n_A$$
$$\bar{x}_q^{min} \leq x_q \leq \bar{x}_q^{max} \qquad q=1,2,..., n_{coord}$$

Onde g_j são as restrições secundárias (restrições de comportamento), representadas nos problemas testes por: frequência natural da estrutura (ω_j) e tensão de escoamento do material (σ_j) quanto à tração (σ_j^T) ou compressão por flambagem (σ_j^F). O $\overline{\omega}_j$ é o valor limite para os diferentes modos de vibração (n_ω) existentes na treliça, o qual dever ser respeitado pelo valor de frequência obtido com o programa de otimização (ω_j). As restrições laterais são representadas pelas limitações mínimas ($\bar{A}_i^{min}$, $\bar{x}_q^{min}$) e máximas ($\bar{A}_i^{max}$, $\bar{x}_q^{max}$) impostas às variáveis de projeto, que correspondem área da seção transversal da barra (A_i) e as coordenadas de alguns nós (x_q). Resolvendo o problema estrutural, caso alguma restrição secundária não seja respeitada, uma vez encontrado o valor da função objetivo, este será penalizado pelo fator de penalização (FP) na solução corrente. A equação (5) apresenta a formulação dada à função penalizada.

$$Massa_p^* = \rho \sum_{i=1}^{mbarras} A_i L_i \,(1 + FP) \tag{5}$$

Para a otimização das treliças, foram utilizadas 60 partículas (*n*) e nas Tabelas 2 e 3, estão dispostos os demais parâmetros do PSO, estes são definidos de acordo com o melhor desempenho registrado pelo algoritmo perante o problema proposto. Dois deles representam a condição de parada para os métodos, os quais estão implementados nos algoritmos de forma a garantir que a solução apresentada seja a melhor possível (tolerância quanto à covariância (tol_{COV})) e a qual não esteja violando nenhuma restrição (tolerância quanto ao fator de penalização (tol_{FP})). Quanto à plataforma computacional, esta é composta por um processador *quad core* (quatro núcleos), 298.09GB de HD, memória RAM de 8.0 GB, sistema operacional de 64 Bits e Windows Vista. Foram realizadas 40 reinicializações (otimizações para cada treliça, com a posição inicial das partículas do enxame aleatória e diferente em cada reinicialização), para calcular as métricas de avaliação dos algoritmos, as quais são mensuradas assim que o método tenha convergido.

8 EXEMPLOS NUMÉRICOS

8.1 Treliça espacial de 72 barras

A estrutura treliçada espacial, apresentada na Figura 8, possui 72 barras e 20 nós, sendo que os nós localizados na extremidade superior (1, 2, 3, 4), possuem cada, uma massa concentrada de 2270 kg. Neste problema há 16 variáveis a serem otimizadas para obter a massa ótima da estrutura, as quais correspondem à área da seção transversal das barras. Quanto às restrições, essas são relativas às frequências naturais da estrutura, onde $\omega_1 = 4$Hz e $\omega_2 \geq 6$Hz. Um dos primeiros autores a analisar este problema foi Konzelman (1986) com o método *Dual Method* (DM). As áreas da seção transversal das barras são classificadas em grupos da seguinte forma: G1 – A_1 a A_4, G2 – A_5 a A_{12}, G3 – A_{13} a A_{16}, G4 – A_{17} a A_{18}, G5 – A_{19} a A_{22}, G6 – A_{23} a A_{30}, G7 – A_{31} a A_{34}, G8 – A_{35} a A_{36}, G9 – A_{37} a A_{40}, G10 – A_{41} a A_{48}, G11 – A_{49} a A_{52}, G12 – A_{53} a A_{54}, Q13 – A_{55} a A_{58}, G14 – A_{59} a A_{66}, G15 –

A_{67} a A_{70}, G16 – A_{71} a A_{72}. As propriedades do material e da estrutura são listadas na Tabela.1. A presente pesquisa referencia-se ao trabalho publicado por Miguel e Fadel Miguel (2012), onde, utilizando a metaheuristica *Firefly Algorithm* (FA) para o processo de otimização, obteve uma massa média otimizada da estrutura de 329,89kg e uma massa mínima de 327,691 kg, não se preocupando com o desempenho do método e sim com o melhor resultado otimizado, para cada execução eram realizadas 10 mil iterações. Seus respectivos valores de áreas das barras para a massa mínima são apresentados na Tabela 4 e 5.

Propriedades	Valores	Unidades
E (Módulo de Elasticidade)	6,98 x 10^{10}	N/m^2
ρ (Densidade do material)	2770.0	kg/m^3
Massa concentrada	2270.0	Kg
Limite inferior das variáveis de projeto	0.6452 x 10^{-4}	m^2
Restrições de frequência	ω_1 = 4.0, $\omega_2 \geq 6.0$	Hz

Tabela 1: Propriedades do material e restrições da estrutura de 72 barras.

$N°_{part\ículas}$	ω	C_1	C_2	tol_{COV}	tol_{FP}
60	0.62	2.03	2.03	10^{-3}	10^{-5}

Tabela 2: Parâmetros utilizados para execução do PSO na estrutura de 72 barras.

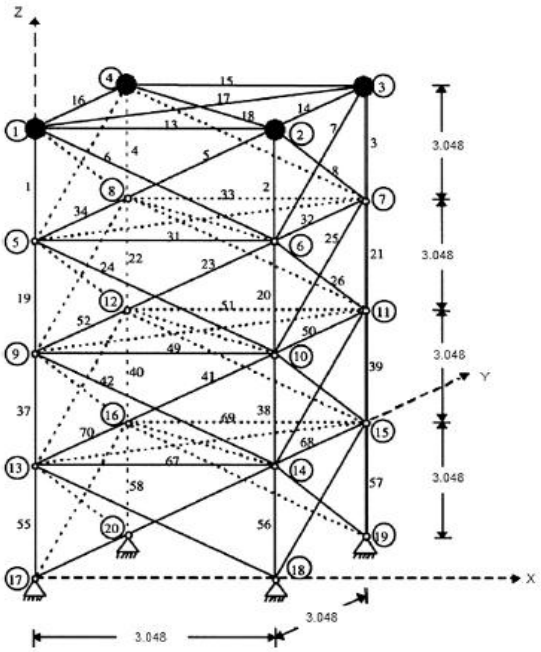

Figura 8: Estrutura espacial de 72 barras com massa concentrada (dimensões em m)

8.2 Treliça plana de 18 barras

A estrutura plana de 18 barras corresponde a um dos casos clássicos de otimização estrutural já analisado por diversos autores com diferentes métodos (Imai e Schmit (1981), Felix (1981), Yang (1996), Soh e Yang (1996), entre outros). Como referência, será utilizado o artigo publicado por Lee e Geem (2005), os quais utilizaram o algoritmo *Harmony Search* (HS) para chegar a uma massa mínima otimizada da estrutura de 2048,20 kg após 25 mil

iterações. Os valores das variáveis de projeto para essa massa mínima são apresentados na Tabela 8 e 9 respectivamente. O presente artigo resolve este problema usando o algoritmo por enxame de partículas na forma síncrona (SSPSO) e assíncrona (SAPSO), bem como suas respectivas versões seriais e paralelas (PSPSO e PAPSO), detalhados anteriormente. O problema corresponde a uma treliça composta por 18 barras e 11 nós, que esta sujeita à aplicação da carga concentrada P nos nós 1, 2, 4, 6 e 8, ilustrada na Figura 9, com a magnitude de 20 kips (88964 N). A densidade do material bem como módulo de elasticidade são, respectivamente, 0,1 lb/in³ (2768 kg/m³) e 10000kpsi (68,95 GPa). A seção transversal inicial das barras adotada para este trabalho é de 10,25 in² (0,00661 m²), a qual pelo processo de otimização pode variar de forma contínua entre 3,5 in² (0,00226 m²) e 18 in² (0,01161 m²). Também, visando à minimização da massa, as coordenadas dos nós 3, 5, 7 e 9, podem ser alteradas. As áreas da seção transversal são designadas a grupos de barras, atendendo ao projeto da estrutura, da seguinte forma: G1 – $A_1 = A_4 = A_8 = A_{12} = A_{16}$, G2 – $A_2 = A_6 = A_{10} = A_{14} = A_{18}$, G3 – $A_3 = A_7 = A_{11} = A_{15}$ e G4 – $A_5 = A_9 = A_{13} = A_{17}$. Portanto, as variáveis a serem otimizadas correspondem às coordenadas dos quatro nós (cada nó pode variar as coordenas em x e y) e os quatro grupos de barras, totalizando doze variáveis de projeto. São consideradas apenas restrições de tensão. Quanto à tração, a tensão normal atuante na barra não pode exceder a tensão escoamento do material igual a 20 kpsi (137,90 MPa), quanto à compressão, pela condição de flambagem, a tensão normal não pode exceder a tensão crítica de flambagem da barra, dada pela equação (6) de Euler.

$$\sigma_j^F = \frac{-kEA_j}{L_j^2}, \quad j = 1, \ldots, 18 \tag{6}$$

Onde *k* representa a constante determinada pela geometria da seção transversal da barra, como sendo, para o presente estudo, igual a 4, E corresponde ao módulo de elasticidade do material, A_j é a área da seção transversal da barra *j* e L_j é o comprimento da barra.

$N°_{part\,ículas}$	ω	C_1	C_2	tol_{COV}	tol_{FP}
60	0.5	1.5	1.5	10^{-3}	10^{-5}

Tabela 3: Parâmetros utilizados para execução do PSO na estrutura de 18 barras.

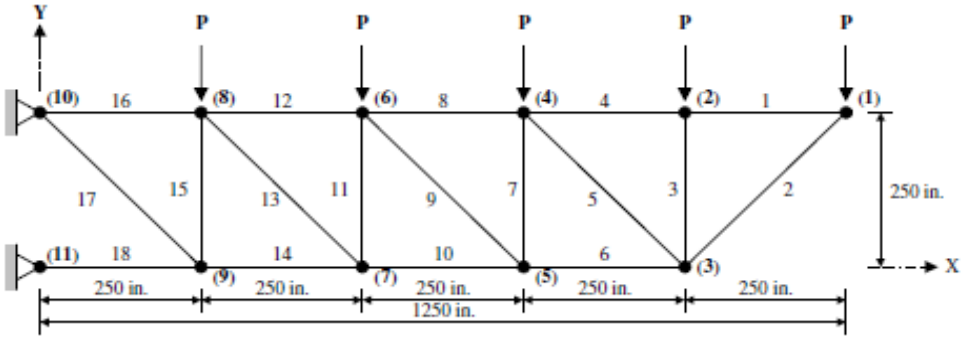

Figura 9: Treliça plana de 18 barras

9 RESULTADOS E DISCUSSÕES

Atendendo as informações apresentadas na seção 6, quanto às métricas para avaliação dos algoritmos, os resultados obtidos por Miguel e Fadel Miguel (2012) correspondem ao ponto ótimo pelo qual os algoritmos propostos tendem a chegar. Sendo assim, na Tabela 4, estão dispostos os valores de massa média encontrados pelos algoritmos propostos bem como as

respectivas métricas para sua avaliação.

	Método	$Massa_{média}$ [kg]	*Acurácia* [kg]	*Robustez* [kg]	$Perf._m$
Presente Trabalho	SSPSO	332,041	4,350	9,112	6828
	SAPSO	332,635	4,944	9,450	4887
	PSPSO	334,808	7,117	13,773	6485
	PAPSO	331,945	4,250	9,310	2940
	Método	$Massa_{min}$ [kg]			
Miguel (2012)	FA	327,691			

Tabela 4: Massa otimizada e respectivas métricas para a treliça de 72 barras.

A estrutura otimizada pelos algoritmos síncronos, (SSPSO e PSPSO) correspondem a uma massa média de 332,041 e 334,808 kg, repercutindo valores pouco atraentes em termos de acurácia e robustez a uma performance média próxima entre eles. Já nos algoritmos assíncronos, apesar da similaridade com os algoritmos síncronos, quanto à massa, acurácia e robustez, os ganhos em termos de performance começam a ser significativos e se tratando do algoritmo assíncrono paralelo (PAPSO), verificam-se consideráveis ganhos em termos de performance, com relação aos demais métodos. A Figura 10 complementa as informações supracitadas, na qual é possível perceber a robustez através da dispersão de valores para a massa de cada método ao longo das reinicializações.

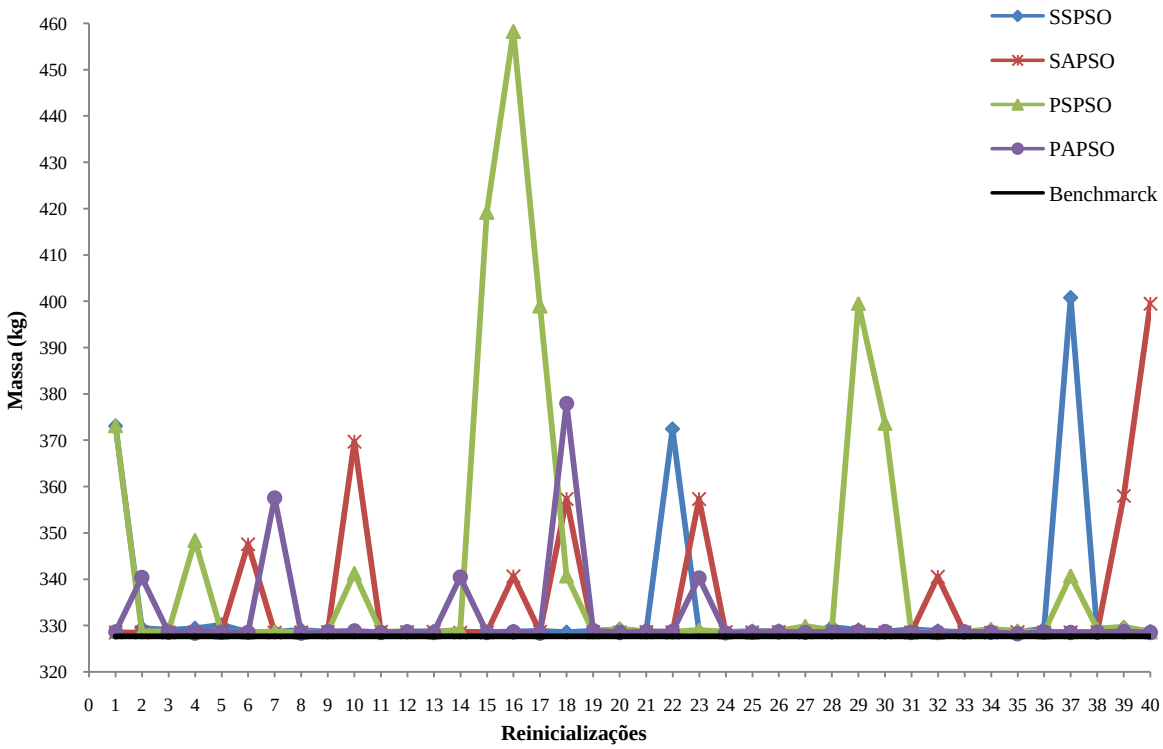

Figura 10: Estrutura espacial de 72 barras

Analisando o gráfico da Figura 10, fica evidente que os pontos mais dispersos foram obtidos com o PSPSO, denunciando a robustez desfavorável encontrada pelo algoritmo se comparado com os demais. A acurácia esta representada no gráfico pelos pontos que representam a solução encontrada por cada método ao longo das reinicializações em relação à

solução conhecida do problema (*benchmarck* – Miguel e Fadel Miguel (2012)). Também para esta métrica, o PSPSO não tem demonstrado bons resultados, os quais para inúmeras buscas ficaram longes da solução conhecida do problema. É importante ressaltar que o motivo pelo qual não foi registrada nenhuma massa menor que a do *benchmarck* corresponde a condição de parada do algoritmo, a qual é atendida quando satisfeita as tolerâncias apresentadas na Tabela 2, ou quando o algoritmo chega a uma solução pré-estabelecida (*benchmarck*) sem violar nenhuma restrição (tol_{FP}). Quanto à performance, a Figura 11 contempla o comportamento de cada método na busca pela solução ótima (número de iterações) durante as 40 reinicializações.

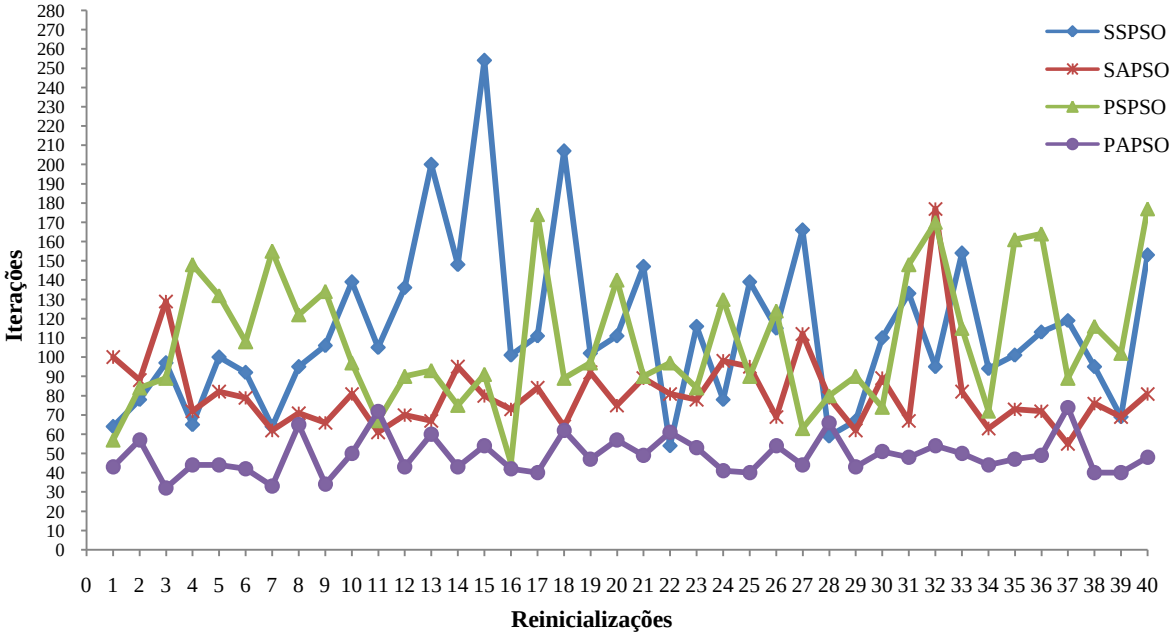

Figura 11: Performance, representada pelas iterações, durante a realização dos 40 experimentos

Pela posição dos pontos e suavidade da curva, fica evidente o bom e uniforme desempenho obtido com o algoritmo paralelo assíncrono, o qual supera os demais métodos. Comportamento similar é verificado com a versão sequencial do algoritmo assíncrono, o qual, apesar de apresentar menor performance que o PAPSO, demonstrou-se mais regular, exceto pela terceira e trigésima segunda iteração, se comparados com as versões sequenciais e síncronas do PSO. Com as Tabelas 5 e 6, pode-se comparar a média das 40 rodadas para cada variável de projeto, bem como seu respectivo desvio padrão, com a melhor solução encontrada por Miguel e Fadel Miguel (2012).

Grupos de Barras	Miguel (2012)	Presente Trabalho			
	FA	SSPSO		SAPSO	
	Var.Proj. (cm²)	Var. Proj.$_{\text{média}}$	Desv. Padrão	Var. Proj.$_{\text{média}}$	Desv. Padrão
G1	3,341	3,440	0,211	3,364	0,305
G2	7,759	7,908	0,183	7,945	0,097
G3	0,645	0,771	0,297	0,645	0,000
G4	0,645	0,645	0,000	0,645	0,000
G5	9,020	10,209	6,978	10,056	7,042
G6	8,257	8,086	0,113	7,991	0,125

G7	0,645	0,645	0,000	0,645	0,000
G8	0,645	0,645	0,000	0,645	0,000
G9	12,045	12,608	0,632	12,337	1,094
G10	8,040	8,061	0,049	8,112	0,084
G11	0,645	0,645	0,001	0,646	0,002
G12	0,645	0,645	0,000	0,645	0,000
G13	17,380	17,273	1,359	18,408	4,176
G14	8,056	8,155	0,160	8,155	0,127
G15	0,645	0,645	0,000	0,645	0,000
G16	0,645	0,645	0,000	0,645	0,000

Tabela 5: Média das variáveis de projeto obtidas com os algoritmos sequenciais.

Grupos de Barras	Miguel (2012)	Presente Trabalho			
	FA	PSPSO		PAPSO	
	Var.Proj. (cm²)	Var. Proj.$_{\text{média}}$	Desv. Padrão	Var. Proj.$_{\text{média}}$	Desv. Padrão
G1	3,341	6,153	8,381	3,505	0,185
G2	7,759	8,011	0,174	7,927	0,184
G3	0,645	0,645	0,000	0,645	0,001
G4	0,645	0,650	0,013	0,655	0,030
G5	9,020	7,728	0,567	8,159	0,204
G6	8,257	7,898	0,275	8,025	0,134
G7	0,645	0,647	0,005	0,645	0,001
G8	0,645	0,657	0,039	0,645	0,000
G9	12,045	14,437	5,511	12,766	0,704
G10	8,040	8,160	0,212	8,106	0,096
G11	0,645	0,646	0,002	0,645	0,000
G12	0,645	0,645	0,000	0,654	0,020
G13	17,380	16,987	1,423	17,370	0,609
G14	8,056	8,160	0,128	8,141	0,122
G15	0,645	0,645	0,000	0,646	0,001
G16	0,645	0,645	0,000	0,645	0,000

Tabela 6: Média das variáveis de projeto obtidas com os algoritmos paralelos.

Em média, os métodos propostos apresentam soluções próximas às obtidas por Miguel e Fadel Miguel (2012). Entretanto, alguns grupos de barras apresentaram valores de áreas com maior dispersão que os demais. Esse comportamento pode ser verificado nos grupos G5, G9, G13 e também, apenas para o PSPSO, G1. Esse fato não ocorre com o algoritmo PAPSO, o qual registrou uma baixa dispersão em todos os grupos.

De forma análoga, realizaram-se os testes com a treliça plana de 18 barras, tem revelado um comportamento similar ao encontrado no problema estrutural da treliça espacial de 72 quanto à performance média dos métodos (verificar Figura 12 (a) e 12 (b)). Entretanto, observou-se que os algoritmos assíncronos apresentaram um ganho em termos de acurácia e robustez no problema da treliça de 18 barras, em relação aos algoritmos síncronos. Os

respectivos valores das métricas discutidas são ilustrados na Tabela 7.

	Método	$Massa_{média}$ [kg]	$Acur.$ [kg]	$Robust.$ [kg]	$Perf._{média}$
Presente Trabalho	SSPSO	2.338,341	290,141	251,158	10176
	SAPSO	2.160,359	100,090	87,139	4641
	PSPSO	2.388,202	340,002	202,871	9800
	PAPSO	2.169,990	121,790	75,581	2982
	Método	$Massa_{min}$ [kg]			
Lee e Geem (2005)	HS	2.048,200			

Tabela 7: Massa otimizada e respectivas métricas para a treliça de 18 barras.

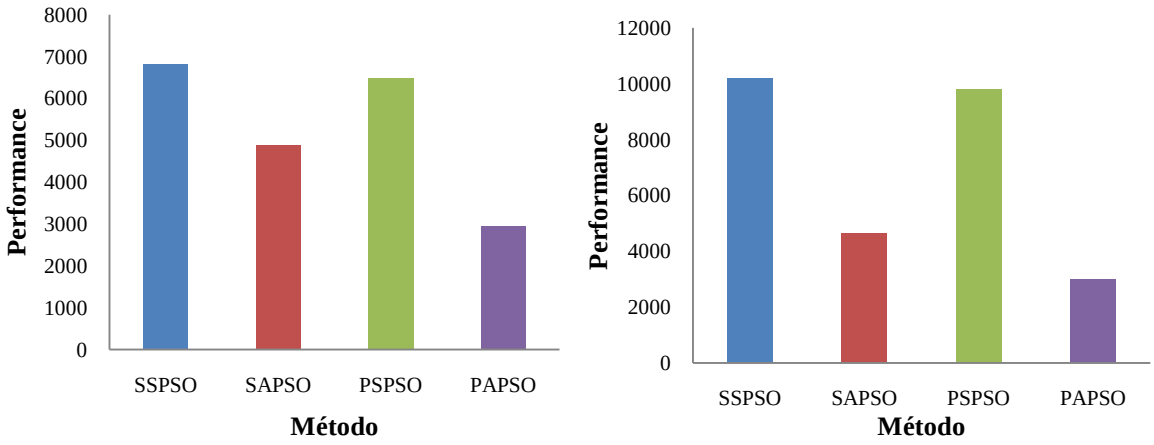

Figura 12: (a) Performance para a treliça de 72 barras, (b) Performance para a treliça de 18 barras

Pela Figura 12, pode-se notar uma semelhança entre a atuação registrada pelos modelos do PSO na otimização das treliças de 72 e 18 barras, demonstrando claramente um desempenho superior da forma assíncrona do PSO em relação ao PSO síncrono. Esse comportamento é dado em função da característica de assincronismo do algoritmo, possibilitando que a cada pseudo revoada, haja uma verificação dos valores ótimos apresentados pela partícula, os quais, se melhores que os da pseudo-revoada anterior, permitem que informações adicionais sejam fornecidas ao enxame. Mais expressivo e observado nos dois problemas analisados, é o desempenho registrado pela versão paralela do algoritmo assíncrono. Esse fato se da em função do PAPSO disponibilizar eventuais valores de partículas ótimas encontradas para as outras partículas que estão sendo analisadas paralelamente, fato este não possível numa versão serial do algoritmo. Este comportamento do algoritmo paralelo possibilita que a convergência para o ótimo se processe com uma significativa diminuição de avaliações da função objetivo.

As Tabelas 8 e 9, apresentam a média das 40 rodadas para cada variável de projeto, bem como seu respectivo desvio padrão, os valores médios podem ser comparados com a melhor solução encontrada por Lee e Geem (2005), para confirmar a proximidade entre as soluções.

Lee e Geem (2005)	Presente Trabalho			
HS	SSPSO		SAPSO	
Var.Proj.	Var. $Proj._{média}$	Desv. Pad.	Var. $Proj._{média}$	Desv. Pad.

Coordenadas Nodais (m)	X3	22,939	23,676	1,541	23,225	0,796
	Y3	4,427	4,498	0,683	4,571	0,319
	X5	16,010	16,741	0,786	16,275	0,643
	Y5	3,462	3,145	0,745	3,195	0,336
	X7	10,213	10,544	0,555	10,187	0,563
	Y7	2,299	1,874	0,439	1,973	0,254
	X9	4,961	5,136	0,181	5,119	0,184
	Y9	0,777	0,222	0,459	0,421	0,362
Seção Trans. Grupos de Barras (cm²)	G1	81,613	75,018	10,915	76,139	4,861
	G2	111,097	115,017	2,716	115,519	1,882
	G3	39,806	73,839	35,843	46,594	17,514
	G4	22,903	39,847	19,895	31,920	8,369

Tabela 8: Média das variáveis de projeto obtidas com os algoritmos sequenciais.

		Lee e Geem (2005)	Presente Trabalho			
		HS	SSPSO		SAPSO	
		Var.Proj.	Var. Proj.$_{média}$	Desv. Pad.	Var. Proj.$_{média}$	Desv. Pad.
Coordenadas Nodais (m)	X3	22,939	23,614	1,065	23,789	1,456
	Y3	4,427	4,710	0,482	4,533	0,482
	X5	16,010	16,626	0,595	16,781	0,818
	Y5	3,462	3,119	0,732	3,157	0,368
	X7	10,213	10,281	0,606	10,471	0,603
	Y7	2,299	1,911	0,463	1,870	0,352
	X9	4,961	5,144	0,135	5,162	0,121
	Y9	0,777	0,217	0,576	0,274	0,425
Seção Trans. Grupos de Barras (cm²)	G1	81,613	74,964	11,364	74,198	5,598
	G2	111,097	115,930	0,611	115,389	1,954
	G3	39,806	87,056	36,515	48,146	12,015
	G4	22,903	37,194	11,402	35,366	9,752

Tabela 9: Média das variáveis de projeto obtidas com os algoritmos paralelos.

10 CONCLUSÕES

Alguns resultados iniciais foram apresentados quanto à avaliação dos algoritmos desenvolvidos na otimização dos problemas estruturais das treliças de 18 e 72 barras. Cada método foi submetido a exaustivos testes, os quais revelaram algumas particularidades associadas à acurácia, robustez e performance, confirmando o comportamento aleatório que os métodos de otimização apresentam de acordo com o problema ao qual são submetidos. Os resultados numéricos indicam que a forma assíncrona do algoritmo por enxame de partícula encontrou mais facilmente a solução ótima dos problemas estruturais analisados, superando significativamente os algoritmos síncronos em termos de performance. É preciso observar que para os dois casos de otimização, os modelos SSPSO, SAPSO e PAPSO convergiram para soluções ótimas semelhantes, já o modelo PSPSO apresentou soluções sub-ótimas, repercutindo numa massa média otimizada superior as demais versões do PSO, o que demonstra uma maior susceptibilidade para ficar "preso" em ótimos locais. Tratando de

processamento paralelo aplicado à forma assíncrona do PSO, o algoritmo tende a encontrar mais rapidamente a solução desejada, tornando o método ainda mais eficiente. Sendo assim, como o foco do trabalho é o desempenho apresentado pelo processamento paralelo com multiprocessadores, os resultados apresentados demonstram que esta técnica mostrou-se bastante atrativa. Além disso, foi demonstrado que a solução encontrada com a forma serial do algoritmo seja mantida na versão paralela e em alguns casos (como no problema da treliça de 72 barras) até melhorada. Espera-se que o desempenho apresentado represente uma significante redução no tempo computacional. Entretanto, para isso se faz necessário a utilização de outras métricas de avaliação dos algoritmos, as quais não são abordadas no presente artigo mas com a manutenção da pesquisa na área, novos resultados em termos de eficiência devem surgir, bem como, novas alternativas de implementações utilizando os métodos heurísticos.

REFERÊNCIAS

Belal, M. and El-Ghazawi, T. Parallel Models for Particle Swarm Optimizers. IJICIS, v. 4, n. 1, p. 100-111, 2004.

Cantú-Paz, E., Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers, Boston, 2000.

Eberhart, R. and Kennedy, J. A New Optimizer Using Particle Swarm Theory. *Proccedings of the Sixth International Symposium on Micro Machine and Human Science*, v.,39-43, 1995.

Felix, J. E., Shape Optimization of Trusses Subjected to Strength, Displacement, and Frequency Constraints, Master's Thesis, Nava Postgraduate School, 1996.

Imai, K., Schmit, A. L. Jr., Configuration Optimization of Trusses, ASCE J. Struct. Div 107 (ST5) 745-756, 1981.

Koh, B., George, A. D., Haftka, R., T. and Fregly, B., J., Parallel Asynchronous Particles Swarm Optimization. *International Journal for Numerical Methods in Engineering*, 67:578-595, 2006.

Kozelman, C., J., Dual Methods and Approximation Concepts for Optimization. M.A.Sc. Theseis, Departament of Mechanical Engineering, University of Toronto, 1986.

Lee, K. S., Geem, Z. W., A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice, *Computer Methods in Applied Mechanics and Engineering*, 194:3902-3933, 2005.

Miguel, L. F. F., Fadel Miguel, L. F., Shape and Size Optimization of Truss Structures Considering Dynamic Constraints Through Modern Metaheuristic Algorithms, *Expert Systems with Applications*, 39:9458-9467, 2012.

Mostaghim, S., Branke, J. and Schmeck, H., Multi-objective Particle Swarm Optimization on Computer Grids. GECCO, p. 869-875, 2007.

Schutte, J. F.; Reinbolt, J.A.; Fregly, B. J.; Haftka, R. T. and George, A. D., Parallel Global Optimization With The Particle Swarm Algorithm. *International Journal for Numerical Methods in Engineering*, v. 61, p. 2296-2315, 2004.

Soh, C. K., Yang, J. P., Fuzzi Controlled Genetic Algorithm Search for Shape Optimization, ASCE J. Comput. Civ. Engrg. 10(2) 143-150, 1996.

Venter, G. and Sobieszczanski-Sobieski, J., A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evalutions, *6th World Congresses of Structural and Multidisciplinary Optimization*, 2005.

Yang, J. P., Development of Genetic Algorithm-Based Approach for Structural Optimization, Ph.D. Thesis, Nanyang Technology University Singapore, 1996.

Yang, X. Y., Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley & Sons, 25-28, 2010.