Evolução Diferencial com ensemble de Operadores de Mutação em GPGPUs para o Despacho Econômico de Energia Elétrica

Differential Evolution with ensemble of Mutation Operators in GPGPUs for Economic Load Dispatch

Gustavo H. Czaikoski [1, 2]
Paulo R. Urio [1, 3]
Richard A. Gonçalves [1, 4]
Carolina P. Almeida [1, 5]
Josiel N. Kuk [1, 6]
Sandra M. Venske [1, 7]

Data de submissão: 15/04/2015, *Data de aceite: 20/06/2016*

Resumo: O Problema do Despacho Econômico de Energia Elétrica (PDEE) visa minimizar o custo de produção de energia de uma usina termoelétrica. Após a análise do algoritmo sequencial, neste trabalho, o PDEE será tratado com um algoritmo paralelo para GPGPUs em CUDA. O algoritmo proposto é uma Evolução Diferencial (ED) utilizando a técnica de *ensemble* de operadores de mutação. A ED é uma técnica estocástica de otimização baseada em população, desenvolvida para a otimização de valores reais enquanto o *ensemble* de operadores de mutação permite que várias configurações de parâmetros e estratégias possam ser utilizadas em cada etapa da evolução do algoritmo. Três instâncias de teste, considerando os efeitos de ponto de válvula, são adotadas para verificar a eficiência do método proposto. Os resultados obtidos são favoravelmente comparados com aqueles descritos na literatura da área em termos de qualidade das soluções obtidas. A versão paralela obteve *speedups* significativos mantendo a boa qualidade das soluções encontradas.

***P*alavras-chave:** computação evolucionária, *ensemble* de operadores de mutação, processamento paralelo.

[1] Departamento de Ciência da Computação, UNICENTRO - Universidade Estadual do Centro-Oeste - Guarapuava, Paraná, Brasil. Caixa Postal 730.
[2] {gczaikoski@hotmail.com}
[3] {paulourio@gmail.com}
[4] {richard@unicentro.br}
[5] {carol@unicentro.br}
[6] {jnkuk@unicentro.br}
[7] {ssvenske@unicentro.br}

Abstract: The Economic Load Dispatch Problem (EDP) aims to minimize the total cost of fossil fuel consumed in power generators of a thermoelectric power plant. In this work the EDP will be treated by a parallel algorithm for GPGPUs (*General Purpose Graphics Processing Units*) in CUDA (*Compute Unified Device Architecture*). The proposed algorithm is the Differential Evolution (DE) using the ensemble of parameters technique. DE is a stochastic optimization technique based on population developed for the improvement of real values while the ensemble of parameters allows the utilization of various parameters settings and strategies through each stage of the evolutionary process. Three test instances considering the valve point effects are adopted to verify the effectiveness of the proposed method. The results are favourably compared with those described in the literature.

***K*eywords:** evolutionary computation, ensemble of parameters, parallel processing

1 Introdução

A otimização é uma área da Matemática que possui muitas aplicações e visa minimizar ou maximizar pelo menos uma função objetivo [1]. Muitos problemas de otimização pertencem à classe de problemas NP-difícil [2], por isso o uso de bons métodos aproximativos tem ganhado destaque para solução desses problemas.

O Problema do Despacho Econômico de Energia Eletrica (PDEE) é um problema na área de geração e distribuição de energia e tem aplicações práticas quando se deseja otimizar os custos operacionais de tais tarefas [1]. O PDEE se encaixa na classe de problemas de otimização contínua. Procura-se minimizar o consumo de combustível dos geradores de energia elétrica para que seja produzida a quantidade de energia demandada. Devido a sua complexidade este problema tem sido abordado com o uso de metaheurísticas, principalmente Algoritmos Evolucionários [3].

Algoritmos Evolucionários (AEs) são conhecidos pela capacidade de lidar com problemas complexos de otimização [4]. As metaheurísticas surgiram nos últimos anos como soluções eficientes, genéricas e robustas para uma gama de problemas NP-difíceis [5]. A Evolução Diferencial é um AE de otimização numérica desenvolvido por Storn e Price em meados da década de noventa [6] e pode ser aplicada ao PDEE. A ED é um AE eficiente no contexto da otimização não-linear com variáveis contínuas, devido às características de robustez, versatilidade e autoadaptação [7]. A ED foi bem sucedida quando aplicada em diversos campos, como engenharia mecânica, comunicação, óptica, reconhecimento de padrões, processamento de sinais e sistemas de energia [8].

A ED utiliza alguns parâmetros como tamanho da população (NP), constante de cruzamento (CR) e constante de mutação (F) - peso dos vetores de diferenças. Neste trabalho a escolha dos valores de CR e F, bem como do operador de mutação será realizada utilizando a técnica de *ensemble* de operadores de mutação [8, 9]. No *ensemble* cada parâmetro possui um conjunto de valores competindo para produzir uma prole futura baseada em seus sucessos em gerações passadas [8].

A utilização de técnicas de programação paralela tendo sido estudada para acelerar a obtenção de soluções em problemas de otimização. Recentemente foi proposto o uso de Unidades de Processamento Gráfico de Propósito Geral ou GPGPUs para essa aceleração [10]. GPGPUs possuem diversos multiprocessadores, cada um contendo vários núcleos. Portanto, é um recurso próprio para o processamento paralelo. A programação em GPGPUs, neste trabalho, utilizará a linguagem C com extensões CUDA. A Evolução Diferencial, assim como outros algoritmos evolucionários, pode obter um bom ganho de desempenho quando paralelizada [11].

Sendo assim, este trabalho aplica o algoritmo de ED com *ensemble* de operadores de mutação para a solução do PDEE. O algoritmo será implementado em CUDA para ser executado em GPGPUs visando reduzir o tempo computacional de execução do algoritmo proposto. Os resultados serão avaliados tanto em termos de qualitativos quanto em relação ao *speedup* alcançado com a paralelização.

O restante desse artigo está organizado como segue: a Seção 2 apresenta conceitos importantes para a compreensão do trabalho. A Seção 3 descreve a ED implementada para o PDEE em sua implementação sequencial e paralela. A Seção 4 detalha os experimentos realizados e os resultados obtidos. Por fim, tem-se a Seção 5 onde as conclusões obtidas e alguns possíveis trabalhos futuros são descritos.

2 Revisão Bibliográfica

Nesta seção são apresentados alguns conceitos relacionados ao tema deste trabalho. Inicialmente, o problema abordado é descrito formalmente. A seguir, na subseção 2.2, o algoritmo evolucionário (ED) é descrito, bem como a técnica de *ensemble* de operadores de mutação. Na subseção 2.3, a linguagem CUDA, escolhida para a implementação paralela, é brevemente descrita.

2.1 Problema do Despacho Econômico de Energia Elétrica

O Problema do Despacho Econômico de Energia Elétrica é um dos mais importantes problemas a serem resolvidos durante o planejamento e a operacionalização de um sistema de geração de energia elétrica [1]. O PDEE tem por objetivo minimizar o custo de combustível

dos geradores de energia elétrica sujeitos a restrições operacionais. Formalmente o PDEE pode ser descrito como [12]:

$$\begin{array}{c} \text{Minimizar FC} = \sum_{j=1}^{n} FC_j(P_j) \\ \text{sujeito a } \sum_{j=1}^{n} P_j - P_D = 0 \quad \text{e} \quad P_j^{min} \leq P_j \leq P_j^{max} \end{array} \tag{1}$$

sendo que FC é o custo total considerando todos os geradores, $FC_j(P_j)$ é a função de custo do *j-ésimo* gerador (em \$/hr), P_j é a potência de saída da *j-ésima* unidade, *n* é o número de unidades geradoras no sistema, P_D é a demanda total de potência, P_j^{min} e P_j^{max} são, respectivamente, a potência de saída mínima e máxima da *j-ésima* unidade. A restrição de igualdade da Equação 1 é chamada de restrição de balanceamento de potência enquanto a restrição de desigualdade é chamada de restrição operacional. Seguindo as restrições, a potência total gerada deve ser igual a demanda total do sistema e a potência gerada em cada unidade deve estar entre os seus limites mínimos e máximos.

O cálculo de custo (FC) é dado pela soma do custo de cada gerador. Neste trabalho, o custo leva em consideração o efeito de ponto de válvula, que é a interferência no custo por conta da abertura de válvulas [3]. Considerando o efeito de ponto de válvula o custo de cada gerador pode ser representado da seguinte forma:

$$FC_j(P_j) = a_j * P_j^2 + b_j * P_j + c_j + |e_j * \sin(f_j * (P_j^{min} - P_j))| \tag{2}$$

tal que a_j, b_j, c_j, e_j e f_j são os coeficientes de custo de combustível da *j-ésima* unidade com efeito de ponto de válvula. Soluções para o PDEE podem ser encontradas utilizando-se técnicas da Evolução Diferencial. Um exemplo da utilização da ED no PDEE está em [13].

2.2 Evolução Diferencial e *Ensemble* de Operadores de Mutação

A Evolução Diferencial é um algoritmo evolutivo para otimização numérica aplicado com sucesso em diferentes problemas [7]. Os passos do algoritmo de Evolução Diferencial são apresentados a seguir. A ED é iniciada com a geração aleatória da população inicial sendo, em seguida, cada indivíduo avaliado por uma função de *fitness*. Na sequência o algoritmo entra em seu laço principal. Neste laço, para cada solução presente na população, são selecionados aleatoriamente três soluções diferentes da solução corrente. Então, os procedimentos de mutação, cruzamento e seleção são aplicados.

A mutação funciona como segue: calcula-se as diferenças entre dois ou mais vetores (soluções), essas diferenças são multiplicadas pelo fator de mutação (F) e este resultado é somado a um outro vetor diferente daqueles utilizados anteriormente (o vetor alvo), gerando o vetor experimental [4]. Neste trabalho foram utilizadas as estratégias de mutação *rand/1/bin*, *rand/2/bin*, *best/1/bin* e *best/2/bin*. Na ED, o desempenho de cada estratégia está associado a diferentes característica do problema de otimização considerado. As estratégias *rand/1/bin*

e *rand/2/bin* têm velocidade de convergência mais lenta, mas possuem forte capacidade exploratória para evitar a convergência prematura. Assim, elas apresentam melhor desempenho na solução de problemas multimodais, caso do problema do despacho econômico de energia elétrica. Por outro lado, *best/1/bin* e *best/2/bin* possuem rápida velocidade de convergência, por utilizarem a melhor solução durante a exploração do espaço de busca. Neste trabalho, foram utilizadas estas diferentes estratégias para compor o *ensemble* de operadores a fim de combinar suas vantagens e características. Em [9, 8, 14] foram utilizadas também as estratégias *rand-to-best/1/bin* e *current-to-rand/1/bin*. Em testes empíricos, a estratégia *rand-to-best/1/bin* não apresentou bons resultados para o problema considerado neste trabalho. A estratégia *current-to-rand/1/bin* é mais adequada a problemas rotacionais, que não é o caso do problema do despacho econômico de energia elétrica.

Uma mutação se diferencia pelos três itens que a nomeiam X/Y/Z, onde X corresponde a forma de seleção do vetor alvo (aleatória ou gulosa), Y corresponde ao número de diferenças ponderadas utilizadas e Z é o tipo de cruzamento adotado (exponencial ou binomial) [4].

A mutação *rand/1/bin* [6] é representada matematicamente pela Equação 3.

$$\mathbf{u}_{i,Ger} = \mathbf{x}_{i1,Ger} + F(\mathbf{x}_{i2,Ger} - \mathbf{x}_{i3,Ger}) \tag{3}$$

sendo Ger a geração atual e $\mathbf{u}_i$ o vetor experimental. Nesta mutação são selecionados aleatoriamente três soluções ($\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$ e $\mathbf{x}_{i3}$) diferentes da solução corrente. A diferença entre duas dessas soluções é calculada, o valor obtido é multiplicado pelo fator de mutação F e somado ao valor da terceira solução (vetor alvo), obtendo-se o vetor experimental.

A mutação *rand/2/bin* [15] é representada matematicamente pela Equação 4.

$$\mathbf{u}_{i,Ger} = \mathbf{x}_{i1,Ger} + F(\mathbf{x}_{i2,Ger} - \mathbf{x}_{i3,Ger}) + F(\mathbf{x}_{i4,Ger} - \mathbf{x}_{i5,Ger}) \tag{4}$$

Aqui são selecionadas aleatoriamente cinco soluções ($\mathbf{x}_{i1}$, $\mathbf{x}_{i2}$, $\mathbf{x}_{i3}$, $\mathbf{x}_{i4}$ e $\mathbf{x}_{i5}$) diferentes da solução corrente. Quatro das soluções obtidas aleatoriamente são subtraídas uma da outra em pares, as duas diferenças obtidas são multiplicadas separadamente pelo fator de mutação F e somadas ao valor da quinta solução, obtendo-se o vetor experimental.

A mutação *best/1/bin* [15] é representada matematicamente pela Equação 5.

$$\mathbf{u}_{i,Ger} = \mathbf{x}_{ibest,Ger} + F(\mathbf{x}_{i1,Ger} - \mathbf{x}_{i1,Ger}) \tag{5}$$

A equação é muito semelhante a Equação 3, mas possui a solução $\mathbf{x}_{ibest,Ger}$ substituindo uma solução aleatória. Duas soluções são selecionadas aleatoriamente ($\mathbf{x}_{i1}$ e $\mathbf{x}_{i2}$) diferentes da solução corrente. A diferença entre as duas soluções aleatórias é calculada, essa diferença é multiplicada pelo fator de mutação F, e o resultado é somado a solução $\mathbf{x}_{ibest,Ger}$ (a melhor solução encontrada até o momento).

Por fim, a mutação *best/2/bin* tem sua representação expressada pela Equação 6.

$$\mathbf{u}_{i,Ger} = \mathbf{x}_{ibest,Ger} + F(\mathbf{x}_{i1,Ger} - \mathbf{x}_{i2,Ger}) + F(\mathbf{x}_{i3,Ger} - \mathbf{x}_{i4,Ger}) \quad (6)$$

Quatro soluções são aleatoriamente selecionadas, essas soluções são subtraídas umas das outras em pares, as duas diferenças obtidas são multiplicadas separadamente pelo valor de mutação F e somadas a solução $\mathbf{x}_{ibest,Ger}$, a melhor solução encontrada até o momento.

Depois de finalizada a etapa de mutação, tem-se como resultado um vetor experimental, que será utilizado na operação de cruzamento. No cruzamento são escolhidas partes do vetor corrente e partes do vetor experimental gerado pela mutação para compor o vetor descendente. Os dois principais cruzamentos utilizados na ED são o cruzamento binomial e o cruzamento exponencial [4]. A principal diferença entre os dois tipos de cruzamento é que no cruzamento binomial, as escolhas dos componentes que serão herdados do pai são independentes, enquanto no cruzamento exponecial é selecionada uma subsequência contínua de componentes. O cruzamento adotado foi o binomial, no qual os componentes da solução corrente ou do vetor experimental são escolhidos de maneira aleatória para compor o vetor descendente. A probabilidade de se escolher um componente do vetor experimental é dada pelo parâmetro CR. O processo de cruzamento binomial pode ser descrito conforme a Equação 7.

$$x'_{i,j} = \begin{cases} u_{i,j}, & \text{se } r_j \leq CR \text{ ou } j = d \\ x_{i,j}, & \text{caso contrário} \end{cases} \quad (7)$$

O vetor descendente $\mathbf{x}'_i$ é gerado pela cópia dos componentes do vetor experimental $\mathbf{u}_i$ e do vetor alvo $\mathbf{x}_i$, dependendo de números aleatórios ($r_j \in [0, 1]$) que são comparados ao parâmetro CR. Para cada componente (j) dos vetores alvo e experimental, se r_j for menor ou igual a CR, o vetor descendente recebe o componente do vetor experimental, e caso contrário, recebe o componente do vetor alvo. É também gerado aleatoriamente um valor d que varia de 1 ao número de componentes, o qual garante que pelo menos um dos componentes do vetor alvo é alterado. Formando assim o vetor descendente.

A seleção é o próximo passo da ED, sendo que o vetor descendente é comparado com o vetor alvo e o melhor dentre eles é escolhido para fazer parte da população na próxima geração. Na seleção a solução gerada para cada solução da população (vetor experimental) substitui a solução corrente se ela possui um menor custo de produção.

2.2.1 *Ensemble* de Operadores de Mutação O algoritmo da ED pode ser utilizado com a técnica de *ensemble* de operadores de mutação. O *ensemble* de operadores de mutação propõe um conjunto de estratégias de mutação e um conjunto de valores para os parâmetros da ED. O conjunto de estratégias de mutação, juntamente com um conjunto de valores para F e CR, concorre para produzir uma população bem sucedida [8, 9].

O conjunto de estratégias e parâmetros deve ser restrito para evitar influências adversas de parâmetros e estratégias de mutação menos eficazes [16]. As estratégias de mutação e os parâmetros presentes nos conjuntos devem possuir características diversificadas, para expor comportamentos distintos durante os estágios da evolução.

A ED com *ensemble* de operadores de mutação consiste de um conjunto de estratégias de mutação com características diversas e um conjunto de valores para cada parâmetro. Cada indivíduo da população inicial (soluções correntes) é associado aleatoriamente a uma estratégia de mutação e parâmetros obtidos de seus respectivos conjuntos, os quais são pré estabelecidos. As soluções da população (vetores correntes) produzem uma prole (vetores descendentes) usando a estratégia de mutação e o cruzamento junto com os parâmetros associados. Se o vetor descendente produzido for melhor do que o vetor pai (corrente), a estratégia de mutação e os valores dos parâmetros são armazenados em um conjunto, e o vetor descendente passa a fazer parte da próxima geração. Caso o vetor pai seja melhor do que o vetor descendente, o vetor pai receberá aleatoriamente novamente uma nova estratégia de mutação e valores de parâmetros para a próxima geração, onde a nova atribuição pode ser totalmente aleatória, ou vir de um conjunto de tamanho limitado que possui configurações que geraram vetores descendentes melhores do que os vetores alvo em gerações anteriores.

2.3 Programação Paralela - CUDA

CUDA[8] é uma plataforma de programação paralela lançada em 2007 pela NVIDIA. É usada para desenvolvimento de software para processadores gráficos e para desenvolver uma variedade de aplicações para GPGPUs que são altamente paralelas e executam em centenas de núcleos [9].A plataforma CUDA permite quase a transferência direta de um código C para dentro da GPGPU. Portanto, a sintaxe da programação CUDA utilizada neste trabalho consiste de extensões para a linguagem C [17]. Essa característica proporciona um ambiente adequado para migrar códigos na linguagem C existentes para a plataforma CUDA.

A arquitetura da GPGPU conta com múltiplos *cores* no modelo SIMD (Single Instruction Multiple Data) [17]. O ideal é que para cada *core* sempre haja uma *thread* para ser executada, maximizando-se, assim, a utilização da GPGPU. Uma *thread* é uma sequência de instruções que podem ser executadas em paralelo.

CUDA permite a criação funções chamadas *kernel* que serão executadas na GPU. Um *kernel* decompõe a tarefa a ser executada em múltiplas *threads* que são executadas em paralelo pelos *cores* da GPGPU. CUDA também disponibiliza memória compartilhada e sincronização de *threads*, entre outros recursos [17].

[8]https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/
[9]http://www.nvidia.com.br/object/cuda_home_new_br.html

3 Algoritmo Proposto

Essa seção descreve o algoritmo proposto baseados em ED com *ensemble* de operadores de mutação. A primeira subseção descreve o algoritmo sequencial, enquanto a segunda subseção descreve a versão paralela para GPGPUs.

3.1 Evolução Diferencial Sequencial: $\text{ED}_{ensemble}$

O Algorimto 1 detalha os passos do algoritmo sequencial de evolução diferencial com *ensemble* de operadores de mutação implementado.

Algoritmo 1: Algoritmo da ED com *ensemble* de Operadores de Mutação - $\text{ED}_{ensemble}$

```
Entrada: Parâmetros NP
Saída: Melhor solução encontrada ao longo da evolução.
início
    Inicializar população de tamanho NP;
    Aplicar o procedimento de reparação;
    Avaliar cada solução da população inicial;
    repita
        para cada solução faça
            Selecionar uma estratégia de mutação e parâmetros F e CR do
              ensemble (Algoritmo 3);
            início
                Aplicar a Mutação de acordo com a estratégia e parâmetros
                  associados;
                Aplicar o Cruzamento baseado no CR associado;
                Aplicar o procedimento de reparação;
            fim
            Avaliar o custo da experiência gerada;
            Aplicar a seleção;
        fim para
    até que o critério de parada seja satisfeito;
fim
```

A ED com *ensemble* de operadores de mutação sequencial para o PDEE se inicia com a geração da população inicial, e cada indivíduo da população recebe, como produção de seus geradores, um valor aleatório dentro do limite mínimo e máximo permitido de cada gerador. Em seguida, cada indivíduo da população que não possui sua produção igual a

demanda passa pelo procedimento de reparação. A energia produzida deve ser exatamente igual a demanda de produção, porque a energia não pode ser armazenada de maneira eficiente quando produzida em excesso. O procedimento de reparação utilizado pelo ED$_{ensemble}$ é apresentado no Algoritmo 2. Seus parâmetros de entrada são: Solução (indivíduo que será reparado); P^{min} e P^{max}, um vetor com a produção mínima e máxima, respectivamente, de cada gerador e Demanda (demanda a ser atendida).

Algoritmo 2: Reparação

Entrada: Solução, P^{min}, P^{max}, Demanda.
Saída: Solução reparada.
início
  Calcular diferença entre produção e Demanda;
  repita
    Sortear aleatoriamente um gerador i;
    se $P_i^{min} <$ *(*P_i *+ diferença)* $< P_i^{max}$ **então**
      Adicionar diferença a P_i;
      Diferença recebe zero;
    fim se
    se *(*P_i *+ diferença)* $> P_i^{max}$ **então**
      Calcular diferença-parcial, P_i^{max} - P_i;
      $P_i = P_i^{max}$;
      Subtrair diferença-parcial de diferença;
    fim se
    se *(*P_i *+ diferença)* $< P_i^{min}$ **então**
      Calcular diferença-parcial, $P_i - P_i^{min}$;
      $P_i = P_i^{min}$;
      Somar diferença-parcial a diferença;
    fim se
  até *diferença == 0*;
fim

A lógica do procedimento é como segue: escolhe-se aleatoriamente um gerador. Se o gerador escolhido é capaz de absorver toda a diferença entre a energia demandada e a energia produzida, modifica-se o valor do gerador de maneira que isso ocorra e encerra-se o procedimento. Caso contrário, o gerador escolhido absorve o maior valor possível, ou seja, minimiza-se a diferença entre a energia demandada e a energia produzida ao máximo possível considerando apenas aquele gerador e repete-se esses passos até que a demanda seja atendida.

A seguir, cada indivíduo é avaliado por uma função de *fitness*, ou seja, tem seus custos

de produção calculados. Na sequência o algoritmo entra em seu laço principal. Neste laço, para cada solução presente na população, é selecionada uma estratégia de mutação e parâmetros F e CR do conjunto de estratégias e do conjunto de parâmetros, ou uma estratégia de mutação com parâmetros associados que pertencem ao conjunto de configurações que já foram bem sucedidas. A origem da estratégia e dos parâmetros é feita aleatoriamente, onde ambos conjuntos possuem 50% de chance de serem escolhidos. Nesse trabalho foram armazenadas em uma fila as 50 melhores configurações. Os passos para a escolha da estratégia e dos parâmetros são apresentadas no Algoritmo 3.

Algoritmo 3: Escolha da Estratégia e Parâmetros

```
seleciona = aleatório entre 0 e 1;
se seleciona <= 0,5 então
    Escolhe estrátegia e parâmetros aleatoriamente;
senão
    Escolhe estratégia e parâmetros já bem sucedidos;
fim se
```

A mutação ocorre baseada nas estratégias de mutação que estão associadas aos indivíduos. As mutações utilizadas em cada estratégia foram explicadas na Seção 2.2. Depois de finalizada a etapa de mutação, tem-se como resultado um vetor experimental, que será utilizado na operação de cruzamento. No cruzamento são escolhidas partes do vetor alvo (solução corrente) e partes do vetor experimental gerado pela mutação para compor o vetor descendente. Com o término da etapa de cruzamento, o vetor descendente passa pelo procedimento de reparação (Algoritmo 2) para atender a demanda de produção, e então passa pelo processo de seleção. A seleção é o próximo passo da ED, sendo que o vetor descendente é comparado com o vetor pai e o melhor dentre eles é escolhido para fazer parte da população na próxima geração. A solução gerada para cada solução da população (vetor experimental) substitui a solução corrente se ela possui um menor custo de produção. A execução do algoritmo se encerra quando o critério de parada adotado é atingido. Neste trabalho, um número máximo de gerações é utilizado como critério de parada.

3.2 Evolução Diferencial Paralela: $\text{ED}_{ensemble_{CUDA}}$

A versão sequencial do algoritmo proposto, $\text{ED}_{ensemble}$, foi analisada utilizando a ferramenta *perf* [18]. Esta ferramenta permite a avaliação do consumo de recursos de cada passo do algoritmo. A avaliação mostrou que a função de cálculo dos custos consome aproximadamente 60% do tempo total de execução do algoritmo, sobrando 40% para ser dividido entre as outras operações. Sendo assim, o *speedup* máximo teórico é igual a 2,5. Contudo, *speedups* maiores podem ser obtidos na prática. Estes dados motivaram o desenvolvimento

de uma versão paralela do $\text{ED}_{ensemble}$ com o intuito de acelerar a execução do algoritmo sem perder a qualidade das soluções obtidas.

Como a função de cálculo de custos é a parte do algoritmo que toma mais tempo, o programa foi modificado para calcular ao mesmo tempo os custos de todas as experiências produzidas de uma geração, ou seja, depois que são feitas as operações de mutação, cruzamento e reparação para todos os indivíduos. Portanto, um *loop* que percorria todas as experiências para calcular o custo é substituído por um *kernel*.

O Algoritmo 1 também representa os passos do algoritmo paralelo de evolução diferencial com *ensemble* de operadores de mutação implementado. O que diferencia o algotimo $\text{ED}_{ensemble_{CUDA}}$ do $\text{ED}_{ensemble}$ é apenas a forma de avaliação: utilizando um *loop* na versão sequencial ou um *kernel* na versão paralela.

4 Experimentos e Resultados

Esta seção apresenta os resultados para três casos de teste do PDEE. Foram realizadas 50 execuções independentes (com populações iniciais diferentes) de cada algoritmo nos três casos de testes considerados. Sendo assim, o custo mínimo, médio e máximo são referentes a essas 50 execuções.

Os algoritmos foram implementados na linguagem C++ e executados em uma máquina Intel Core I5 2.6 GHz, 8 GB de RAM e placa de vídeo NVIDIA GeForce GT 640M. O algoritmo sequencial foi executado no sistema operacional Microsoft Windows 8 na IDE Code::Blocks 13.12. O algoritmo paralelo foi executado no sistema operacional Xubuntu 12.04 em um ambiente de programação UNIX (editor de texto e terminal). A versão da linguagem CUDA utilizada foi a mais recente, o CUDA 6.0. Foi utilizado para implementação do código paralelo o editor de texto Geany, sensível à sintaxe CUDA. Para suportar a linguagem CUDA foi utilizado o NVIDIA CUDA Toolkit 6.0 para C/C++.

Para todos os experimentos, o tamanho da população foi igual ao fator multiplicador vezes o número de geradores (n), sendo o fator multiplicador igual a 10 conforme sugerido em [6] e o critério de parada adotado foi 70.000n avaliações.

Primeiramente, a Subseções 4.1 e 4.2 analisam, respectivamente, os efeitos dos parâmetros (F e CR) e dos *ensembles* de mutações na qualidade das solução para o PDEE. A Subseção 4.3 compara qualitativamente os algoritmos propostos com os resultados reportados na literatura. O *speedup* obtido pela versão paralela é analisado na Subseção 4.4.

4.1 Efeito dos Parâmetros F e CR

Os parâmetros do algoritmo $ED_{ensemble}$ foram ajustados utilizando um Teste Fatorial [19]. Foram realizados testes estatísticos não-paramétricos (*Kruskal-Wallis*) para aferir os melhores ajustes [19]. Os testes tiveram como melhor resultado F = 0,3 e CR = 0,7. Efetuando a análise conjunta das tabelas[10], percebe-se que para valores de CR variando entre 0,7 e 0,9 o algoritmo gera melhores resultados. Também é desejável que o valor de F varie entre 0,1 e 0,4.

4.2 Efeito das Mutações

O objetivo dessa subseção é avaliar os efeitos das diferentes mutações e dos possíveis *ensembles* de mutações. A Tabela 1 mostra os efeitos das mutações.

Tabela 1. Efeito das Mutações.

Efeito das Mutações			
Mutação	**Custo Mín**	**Custo Médio**	**Custo Máx**
rand/1/bin	**121403,53**	121408,23	**121411,89**
rand/2/bin	121403,77	**121407,42**	121411,95
best/1/bin	**121403,53**	121456,81	121563,31
best/2/bin	**121403,53**	121418,76	121458,26
rand/1/bin e rand/2/bin	**121403,53**	121407,76	**121411,89**
best/1/bin e best/2/bin	**121403,53**	121445,02	121544,74
rand/1/bin, rand/2/bin, best/1/bin e best/2/bin	**121403,53**	121426,72	121494,53

A estratégia *rand/1/bin*, *rand/2/bin* e a união das duas, possuem resultados próximos. Contudo, a versão escolhida como melhor *ensemble* foi a *rand/1/bin* e *rand/2/bin* por obter melhores valores de custo mínimo e máximo. No restante do trabalho, apenas essa versão será considerada.

4.3 Comparação com a Literatura

Nesta subseção os resultados obtidos pelos algoritmos $ED_{ensemble}$ e $ED_{ensemble_{CUDA}}$ são comparados com aqueles reportados na literatura da área. Destaca-se que os valores ótimos apresentados foram obtidos de maneira empírica e que não há garantias que eles sejam ótimos globais.

[10] As tabelas dos testes fatoriais não são apresentadas por motivo de espaço.

4.3.1 Primeiro Estudo de Caso - 13 Geradores e Demanda de 1800 MW O primeiro estudo de caso considera 13 geradores com efeito de ponto de válvula e demanda de 1800 MW [13]. Este problema do despacho econômico de energia possui vários mínimos locais e, portanto, seu mínimo global é difícil de ser determinado [20]. A Tabela 2 mostra que os algoritmos propostos obtiveram o melhor valor para o custo mínimo de combustível. O algoritmo ICIS [21] também obteve o melhor valor para o custo mínimo, mas foi inferior ao $\text{ED}_{ensemble}$ nos demais quesitos (custo médio e custo máximo). O algoritmo $\text{ED}_{ensemble}$ também encontrou o menor custo máximo e a menor média de combustível. A Tabela 3 apresenta o melhor resultado obtido pelo $\text{ED}_{ensemble}$ codificado por um vetor P_i, i = 1, ..., 13 com custo mínimo de 17960,36.

Tabela 2. Resultados para o Primeiro Estudo de Caso. Custos em US$.

Caso com 13 geradores e demanda de 1800 MW				
Método	Custo Mín	Custo Médio	Custo Máx	Desv Pad
$\text{ED}_{ensemble}$	**17960,36**	**17960,36**	**17960,36**	0,00
$\text{ED}_{ensemble_{CUDA}}$	**17960,36**	17960,73	17969,34	1,75
ICIS [21]	**17960,36**	17968,59	17969,50	2,44
DEC-SQP [13]	17963,94	17973,13	17984,81	1,97
EP-SQP [22]	17991,03	18106,93	-	-
HQPSO5 [23]	17963,95	18273,86	18633,04	123,22
HDE [24]	17975,73	18134,80	-	-
ST-HDE [24]	17963,89	18046,38	-	-
DE [25]	17963,83	17965,48	17975,36	-
CDEMD [20]	17961,94	17974,68	18061,41	20,30
GSA [26]	17963,84	18041,21	18910,31	-
FAPSO-VDE [27]	17963,82	17963,82	17963,83	-
FA [28]	17963,83	18029,16	18168,80	148,54
θ-PSO [29]	17963,82	17965,20	17980,20	4,38

Tabela 3. Melhores resultados obtidos para o Primeiro Estudo de Caso utilizando o $ED_{ensemble}$ e $ED_{ensemble_{CUDA}}$.

Potência	Geração	Potência	Geração	Potência	Geração
P_1	628,3185	P_6	109,8665	P_{11}	40,0000
P_2	149,5996	P_7	109,8665	P_{12}	55,0000
P_3	222,7490	P_8	109,8665	P_{13}	55,0000
P_4	109,8665	P_9	60,0000	$\sum_{j=1}^{13} P_i$ =	1800,00
P_5	109,8665	P_{10}	40,0000	Custo =	17960,36

4.3.2 Segundo Estudo de Caso - 13 Geradores e Demanda de 2520 MW Também foi abordado o problema de otimização de custo de combustível para 13 geradores com efeito de ponto de válvula e demanda de 2520 MW [3]. A Tabela 4 mostra que o $ED_{ensemble}$ e o $ED_{ensemble_{CUDA}}$ obtiveram os melhores valores para o custo mínimo, custo médio e custo máximo de combustível, juntamente com o algoritmo SIC_c [3]. A Tabela 5 apresenta o melhor resultado obtido pelo $ED_{ensemble}$ codificado por um vetor P_i, i = 1, ..., 13 com custo mínimo de 24164,05.

Tabela 4. Resultados para o Segundo Estudo de Caso. Custos em US$.

Caso com 13 geradores e demanda de 2520 MW				
Método	Custo Mín	Custo Médio	Custo Máx	Desv Pad
$ED_{ensemble}$	**24164,05**	**24164,05**	**24164,05**	**0,00**
$ED_{ensemble_{CUDA}}$	**24164,05**	**24164,05**	**24164,05**	**0,00**
SIC_c [3]	**24164,05**	**24164,05**	**24164,05**	**0,00**
SDE [30]	**24164,05**	24168,28	24200,05	-
DTSA [31]	24169,95	-	-	-
DE [25]	24169,91	24169,91	24169,91	-

Tabela 5. Melhores resultados obtidos para o Segundo Estudo de Caso utilizando o $\text{ED}_{ensemble}$ e $\text{ED}_{ensemble_{CUDA}}$.

Potência	Geração	Potência	Geração	Potência	Geração
P_1	628,3185	P_6	159,7331	P_{11}	77,3999
P_2	299,1993	P_7	159,7331	P_{12}	92,3999
P_3	294,4839	P_8	159,7331	P_{13}	92,3999
P_4	159,7331	P_9	159,7331	$\sum_{j=1}^{13} P_i$ =	2520.00
P_5	159,7331	P_{10}	77,3999	Custo =	24164,05

4.3.3 Terceiro Estudo de Caso - 40 Geradores e Demanda de 10500 MW O Terceiro Estudo de Caso considera 40 geradores com efeito de ponto válvula e demanda de 10500 MW [32]. O espaço de soluções deste estudo de caso contém múltiplos ótimos locais [33]. A Tabela 6 mostra que o $\text{ED}_{ensemble_{CUDA}}$ obteve os melhores valores para o custo mínimo, médio e máximo de combustível. A Tabela 7 apresenta o melhor resultado obtido pelo $\text{ED}_{ensemble_{CUDA}}$ codificado por um vetor P_i, i = 1, ..., 40 com custo mínimo de 121403,48.

Tabela 6. Resultados para o Terceiro Estudo de Caso. Custos em US$.

Caso com 40 geradores e demanda de 10500 MW				
Método	Custo Mín	Custo Médio	Custo Máx	Desv Pad
ED$_{ensemble}$	121403,53	121408,54	121411,89	3,69
ED$_{ensemble_{CUDA}}$	**121403,48**	**121407,44**	**121411,83**	3,66
ICIS [21]	121411,97	121584,64	121751,75	108,13
DEC-SQP [13]	121741,97	122295,12	122839,29	386,18
NPSO-LRS [34]	121664,43	122209,31	122981,59	-
CEP-PSO [35]	123670,00	124145,60	124900,00	-
CEP [32]	123488,29	124793,48	126902,89	-
FEP [32]	122679,71	124119,37	127245,59	-
MFEP [32]	122647,57	123489,74	124356,47	-
IFEP [32]	122624,35	123382,00	125740,63	-
PSO [22]	123930,45	124154,49	-	-
PSO-SQP [22]	122094,67	122245,25	-	-
DEvol [36]	121412,91	121430,00	121464,40	-
HDE [24]	121813,26	122705,66	-	-
ST-HDE [24]	121698,51	122304,30	-	-
DE [25]	121416,29	121422,72	121431,47	-
CDEMD [20]	121423,40	121526,73	121696,98	54,86
CSO [37]	121461,67	121936,19	122844,53	-
BBO [38]	121426,95	121508,03	121688,66	-
ICA-PSO [39]	121413,20	121428,14	121453,56	-
QPSO [40]	121448,21	122225,07	-	-
IABC-LS [41]	121412,73	-	121471,61	-
FAPSO-VDE [27]	121412,56	121412,61	121412,78	-
FA [28]	121415,05	121416,57	121424,56	1,78
QGS [42]	121412,55	121423,52	121438,68	-
θ-PSO [29]	121420,90	121509,84	121852,42	92,39
DEPSO [33]	121412,56	121419,31	121468,25	-

Tabela 7. Melhores Resultados obtidos para o Terceiro Estudo de Caso utilizando o $\text{ED}_{ensemble_{CUDA}}$.

Potência	Geração	Potência	Geração
P_1	110,7998	P_{21}	523,2794
P_2	110,7999	P_{22}	523,2791
P_3	97,3998	P_{23}	523,2793
P_4	179,7330	P_{24}	523,2796
P_5	87,8001	P_{25}	523,2792
P_6	139,9999	P_{26}	523,2793
P_7	259.5996	P_{27}	10,0000
P_8	284.5995	P_{28}	10,0001
P_9	284,5997	P_{29}	10,0000
P_{10}	130,0000	P_{30}	87,8004
P_{11}	94,0000	P_{31}	189.9997
P_{12}	94,0000	P_{32}	189.9996
P_{13}	214,7597	P_{33}	189.9999
P_{14}	394,2792	P_{34}	164,8003
P_{15}	394,2794	P_{35}	199,9995
P_{16}	394.2794	P_{36}	194,3921
P_{17}	489,2794	P_{37}	110,0000
P_{18}	489,2792	P_{38}	110,0000
P_{19}	511,2792	P_{39}	109,9996
P_{20}	511,2791	P_{40}	511,2798
		$\sum_{j=1}^{40} P_i$	10500.00
		Custo =	121403,48

4.4 Análise do *Speedup* do Algoritmo Paralelo

O *speedup* é uma medida de ganho de desempenho calculada dividindo-se o tempo de execução do algoritmo sequencial pelo tempo de execução do algoritmo paralelo [43].

A Tabela 8 mostra que para o fator multiplicador utilizado no trabalho (10), somente para o caso de teste de 40 geradores o algoritmo paralelo supera o sequencial. Isso acontece porque o tempo de chamada do *kernel* é maior do que o tempo utilizado para realizar os cálculos dos custos, ou seja, para os casos de 13 geradores, o *kernel* seria chamado para executar somente 130 (tamanho da população) operações de cálculo de custo. Nota-se também que quando se aumenta o fator multiplicador, o *speedup* aumenta, devido ao fato do maior número de cálculos concorrentes que serão passados para a GPGPU. O melhor *speedup* encontrado

foi de 8,54, para o caso de teste de 40 geradores e fator multiplicador 50.

Para o fator multiplicador utilizado nos testes (10), o melhor *speedup* encontrado foi 3,38, também para o caso com 40 geradores.

Tabela 8. Tempos de Execução.

Fator Multiplicador	Caso de Teste	Sequencial	Paralelo	*Speedup*
10	13 geradores 1	3,39s	4,66s	0,73
	13 geradores 2	3,35s	4,57s	0,73
	40 geradores	32,44s	9,6s	3,38
30	13 geradores 1	13,61s	7,12s	1,91
	13 geradores 2	11,45s	6,88s	1,66
	40 geradores	117,18s	16,6s	7,06
50	13 geradores 1	22,5s	8,1s	2,78
	13 geradores 2	19,21s	7,8s	2,46
	40 geradores	207,8s	24,32s	8,54

5 Conclusões

Neste trabalho foi desenvolvido um algoritmo baseado em Evolução Diferencial com *ensemble* de operadores de mutação na sua versão sequencial ($\text{ED}_{ensemble}$) e paralela ($\text{ED}_{ensemble_{CUDA}}$) para resolver o Problema do Despacho Econômico de Energia Elétrica com efeito de ponto de válvula.

Foram realizados testes estatísticos não-paramétricos (*Kruskal-Wallis*) para aferir os melhores ajustes de parâmetros em termos da qualidade das soluções. Os testes mostraram que os melhores resultados são encontrados quando a constante de mutação F está entre 0,1 e 0,4, e quando a constante de cruzamente está entre 0,7 e 0,9. O fator multiplicador, que define o tamanho da população, influencia significativamente o tempo de execução dos algoritmos.

O $\text{ED}_{ensemble_{CUDA}}$ obteve os melhores valores para custo mínimo de todos os casos de teste. Para o primeiro caso de teste, o $\text{ED}_{ensemble}$ encontrou o menor valor para custo mínimo, médio e máximo. O algoritmo $\text{ED}_{ensemble_{CUDA}}$ também obteve o menor custo máximo para o terceiro caso de teste.

O algoritmo sequencial proposto tem um desempenho razoável para o caso com maior número de geradores. O algoritmo paralelo possui um desempenho melhor do que o sequencial para o caso de 40 geradores para o fator multiplicador adotado. Se o fator multiplicador fosse aumentado, consequentemente aumentando a população, o ganho de desempenho seria

ainda maior. Portanto, a paralelização também foi benéfica para a qualidade das soluções para o PDEE.

Os bons resultados obtidos pela versão paralela estimulam o uso de CUDA em outros algoritmos evolucionários, como por exemplo, outras versões de ED adaptativa (SaDE, *Probability Matching*, *Adaptive Pursuit* e *Multi-armed Bandit*). Outra possibilidade de trabalho futuro é a utilização do algoritmo desenvolvido neste trabalho em outros problemas complexos, como a Predição da Estrutura de Proteínas. Esta técnica de paralelização pode ser aplicada a outros problemas.

Agradecimentos
Os autores agradecem à UNICENTRO, à Fundação Araucária (Projeto 23.116/2012) e ao CNPq (Projeto 456.179/2014-3) pelo apoio financeiro.

Contribuição dos autores:

- Gustavo H. Czaikoski: Implementação da versão serial da Evolução Diferencial com ensemble de operadores de mutação, escrita da descrição do algoritmo serial proposto e execução dos exeperimentos realcionados à versão serial.

- Paulo R. Urio: Implementação da versão paralela da Evolução Diferencial com ensemble de operadores de mutação, escrita da descrição do algoritmo paralelo proposto e execução dos exeperimentos realcionados à versão paralela.

- Richard A. Gonçalves: Implementação do algoritmo de reparação das soluções infactíveis, escrita da revisão bibliográfica sobre programação paralela, supervisão da análise estatística dos resultados obtidos, escrita do resumo, introdução, conclusão e parte dos resultados.

- Carolina P. Almeida: Supervisão da implementação sequencial da Evolução Diferencial com ensemble de operadores de mutação, supervisão dos experimentos relacionados à versão sequencial e da análise dos resultados obtidos e escrita de parte dos resultados.

- Josiel N. Kuk: Implementação do algoritmo de reparação das soluções infactíveis, escrita da revisão bibliográfica sobre o Despacho Econômico de Energia Elétrica, supervisão dos experimentos relacionados à versão sequencial eda análise dos resultados obtidos e escrita de parte dos resultados.

- Sandra M. Venske: Supervisão da implementação paralela da Evolução Diferencial com ensemble de operadores de mutação, escrita da revisão bibliográfica de Evolução Diferencial, supervisão dos experimentos relacionados à versão paralelos e da análise dos resultados obtidos e escrita de parte dos resultados.

Referências

[1] J. Park, Y. Jeong, and W. Lee. An improved particle swarm optimization for economic dispatch problems with non-smooth cost functions. In *IEEE Power Engineering Society General Meeting*, Montreal, Que, 2006.

[2] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[3] Richard Aderbal Gonçalves. *Algoritmos Culturais para o Problema do Despacho de Energia Elétrica*. PhD thesis, Curso de Pós Graduação em Engenharia Elétrica e Informática Industrial, Universidade de Tecnologia Federal do Paraná, 2010.

[4] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition, 2007.

[5] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 2003.

[6] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995.

[7] S. Das and P.N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, Feb 2011.

[8] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Ensemble differential evolution algorithm for CEC2011 problems. In *IEEE Congress on Evolutionary Computation*, pages 1557–1564, 2011.

[9] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, and M.F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679 – 1696, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[10] El-Ghazali Talbi and Geir Hasle. Metaheuristics on GPUs. *Journal of Parallel and Distributed Computing*, 73(1):1 – 3, 2013.

[11] Enrique Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[12] M. Basu. Fuel constrained economic emission load dispatch using hopfield neural networks. *Electric Power Systems Research*, 2002.

[13] L. S. Coelho and V. C. Mariani. Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect. *IEEE Transactions on Power Systems*, 21(2):989–996, 2006.

[14] Qiuzhen Lin, Zhiwang Liu, Qiao Yan, Zhihua Du, Carlos A. Coello Coello, Zhengping Liang, Wenjun Wang, and Jianyong Chen. Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. *Information Sciences*, 339:332 – 352, 2016.

[15] R. Storn. On the usage of differential evolution for function optimization. In *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*, pages 519–523, 1996.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Trans. Evol. Comp*, 13(2):398–417, April 2009.

[17] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 2010.

[18] Roberto A. Vitillo. Performance tools developments. In *Future computing in particle physics*, 2011.

[19] W. J. Conover. *Practical Nonparametric Statistics*. Wiley, 3 edition, 1999.

[20] L. S. Coelho, R. T. T. Souza, and V. C. Mariani. Improved differential evolution approach based on cultural algorithms and diversisty measure applied to solve economic load dispatch problems. *Mathematics and Computers in Simulation*, 79:3136–3147, 2009.

[21] Richard A. Gonçalves, Carolina P. de Almeida, Marco César Goldbarg, Elizabeth Ferreira Gouvea Goldbarg, and Myriam Regattieri Delgado. Improved cultural immune systems to solve the economic load dispatch problems. In *IEEE Congress on Evolutionary Computation*, pages 621–628. IEEE, 2013.

[22] T.A.A. Victoire and A.E. Jeyakumar. Hybrid PSO-SQP for economic dispatch with valve-point effect. *Electric Power Systems Research*, 71(1):51–59, 2004.

[23] L.S. Coelho and V.C. Mariani. Particle swarm approach based on quantum mechanics and harmonic oscillator potential well for economic load dispatch with valve-point effects. *Energy Conversion and Management*, 49(11):1373–1378, 2008.

[24] S. K. Wang, J. P. Chiou, and C. W. Liu. Non-smooth/nonconvex economic dispatch by a novel hybrid differential evolution algorithm. *IET Generation, Transmission & Distribution*, 1(5):793–803, 2007.

[25] N. Noman and H. Iba. Differential evolution for economic load dispatch problems. *Electric Power Systems Research*, 78(3):1322–1331, 2008.

[26] R.K. Swain, N.C. Sahu, and P.K. Hota. Gravitational search algorithm for optimal economic dispatch. *Procedia Technology*, 6(0):411 – 419, 2012.

[27] B. Mohammadi-Ivatloo, A. Rabiee, A. Soroudi, and M. Ehsan. Iteration PSO with time varying acceleration coefficients for solving non-convex economic dispatch problems. *International Journal of Electrical Power & Energy Systems*, 42(1):508 – 516, 2012.

[28] Xin-She Yang, Seyyed Soheil Sadat Hosseini, and Amir Hossein Gandomi. Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *Applied Soft Computing*, 12(3):1180 – 1186, 2012.

[29] V. Hosseinnezhad and E. Babaei. Economic load dispatch using θ-PSO. *Energy Conversion and Management*, 49:160 – 169, 2013.

[30] R. Balamurugan and S. Subramanian. Self-adaptive differential evolution based power economic dispatch of generators with valve-point effects and multiple fuel options. *International Journal of Electrical, Electronic Science and Engineering*, 1(3):142 – 149, 2007.

[31] S. Khamsawang, S. Pothiya, and S. Boonseng. Distributed tabu search algorithm for solving the economic dispatch problem. In *TENCON*, pages 484–487, Chiang Mai, Thailand, 2004.

[32] N. Sinha, R. Chakrabarti, and P. K. Chattopadhyay. Evolutionary programming techniques for economic load dispatch. *Trans. Evol. Comp*, 7(1):83–94, February 2003.

[33] Samir Sayah and Abdellatif Hamouda. A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems. *Applied Soft Computing*, 13(4):1608 – 1619, 2013.

[34] A. Immanuel Selvakumar and K. Thanushkodi. New particle swarm optimization solution to nonconvex economic dispatch problems. *IEEE Transactions on Power Systems*, 22(1):42–51, 2007.

[35] N. Sinha and B. Purkayastha. PSO embedded evolutionary programming technique for nonconvex economic load dispatch. In *IEEE PES - Power Systems Conference and Exposition*, pages 66–71, Brasil, 2004.

[36] R.E. Perez-Guerrero and R.J. Cedenio-Maldonado. Economic power dispatch with nonsmooth cost functions using differential evolution. In *Proceedings of the 37th Annual North American Power Symposium*, pages 183–190, Arlington, Virginia, 2005.

[37] A.I. Selvakumar and K. Thanushkodi. Optimization using civilized swarm: solution to economic dispatch with multiple minima. *Electric Power Systems Research*, 79(1):8–16, 2009.

[38] A. Bhattacharya and P. K. Chattopadhyay. Biogeography-based optimization for different economic load dispatch problems. *IEEE Transactions on Power Systems*, 2010.

[39] J. G. Vlachogiannis and K. Y. Lee. Economic load dispatch - a comparative study on heuristic optimization techniques with an improved coordinated aggregation-based PSO. *IEEE Transactions on Power Systems*, 24(2):991–1001, 2009.

[40] K. Meng, H. G. Wang, Z. Dong, and K. P. Wong. Quantum-inspired particle swarm optimization for valve-point economic load dispatch. *IEEE Transactions on Power Systems*, 25(1):215–222, 2010.

[41] Dogan Aydin and Serdar Ozyon. Solution to non-convex economic dispatch problem with valve point effects by incremental artificial bee colony with local search. *Applied Soft Computing*, 13(5):2456 – 2466, 2013.

[42] Mohammad Moradi-Dalvand, Behnam Mohammadi-Ivatloo, Arsalan Najafi, and Abbas Rabiee. Continuous quick group search optimizer for solving non-convex economic dispatch problems. *Electric Power Systems Research*, 93(0):93 – 105, 2012.

[43] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.