Journal of Computer Science 10 (8): 1508-1516, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1508.1516 Published Online 10 (8) 2014 (http://www.thescipub.com/jcs.toc)

LOW COMPLEXITY CONSTRAINTS FOR ENERGY AND PERFORMANCE MANAGEMENT OF HETEROGENEOUS MULTICORE PROCESSORS USING DYNAMIC OPTIMIZATION

[1]Radhamani, A.S. and [1]E. Baburaj

[1]Department of Computer Science and Engineering,
Manonmanium Sundaranar University, Tirunelveli, India
[2]Department of Computer Science and Engineering,
Sun College of Engineering and Technology, Nagercoil, India

Received 2014-01-23; Revised 2014-02-06; Accepted 2014-04-03

ABSTRACT

Optimization in multicore processor environment is significant in real world dynamic applications, as it is crucial to find and track the change effectively over time, which requires an optimization algorithm. In massively parallel processing multicore processor architectures, like other population based metaheuristics Constraint based Bacterial Foraging Particle Swarm Optimization (CBFPSO) scheduling can be effectively implemented. In this study we discuss possible approaches to parallelize CBFPSO in multicore system, which uses different constraints; to exploit parallelism are explored and evaluated. Due to the ability of keeping good balance between convergence and maintenance, for real world applications, among the various algorithms for parallel architecture optimization CBFPSOs are attracting more and more attentions in recent years. To tackle the challenges of parallel architecture optimization, several strategies have been proposed, to enhance the performance of Particle Swarm Optimization (PSO) and have obtained success on various multicore parallel architecture optimization problems. But there still exist some issues in multicore architectures which require to be analyzed carefully. In this study, a new Constraint based Bacterial Foraging Particle Swarm Optimization (CBFPSO) scheduling for multicore architecture is proposed, which updates the velocity and position by two bacterial behaviours, i.e., reproduction and elimination dispersal. The performance of CBFPSO is compared with the simulation results of GA and the result shows that the proposed algorithm has pretty good performance on almost all types of cores compared to GA with respect to completion time and energy consumption.

Keywords: Particle Swarm Optimization, Constraint Based Bacterial Foraging Particle Swarm Optimization, Multicore Processor, Parallel Architecture Optimization

1. INTRODUCTION

Multicore processor task scheduling is a generalised form of machine class scheduling, where a task is processed by more than one core. In wide range of real world problems which are dynamic, requiring an optimization algorithm which is able to continuously track a change, for an optimum performance over time. Global optimization technique like Genetic Algorithm (GA) (Abdel-Magid *et al*., 1999), Tabu Search (TS) (Abido and Abdel-Magid, 2002) and Simulated Annealing (SA) (Abido, 2000) are attracting the attention in the field of parallel architecture parameters optimization in recent times. But when the system has highly correlated parameters to be optimized and the number of parameters to be optimized is large, GA exhibits degraded efficiency (Fogel, 1995). As a new evolutionary technique in (Passino, 2002), bacterial Foraging PSO has been proposed by considering certain constraints. To overcome the drawbacks of conventional methods for multicore

Corresponding Author: Radhamani, A.S., Department of Computer Science and Engineering, Manonmanium Sundaranar University, Tirunelveli, India

processor scheduling, a new optimization scheme known as Constraint based Bacterial Foraging Particle Swarm Optimization (CBFPSO) is used for multicore processor scheduling. CBFPSO appeared as a challenging algorithm for handling the optimization problems. This algorithm can converge to the optimal solution in real world problems and also in dynamic environments, as it is a computational intelligence based technique, which is not largely affected by size and non-linearity of the problem. Some new constraints are required to be included in the optimization algorithm, as the environment assumed is heterogeneous, to trigger various mechanisms and to track the optimum change effectively and efficiently.

Bacterial Foraging algorithm was inspired by foraging behaviour of bacteria and was proposed in (Passino, 2002). Bacterial Foraging Optimization Algorithm (BFOA) has been widely accepted as a global optimization algorithm of current interest for distributed optimization and control. BFOA is inspired by the social foraging behaviour of *Escherichia coli*. BFOA has already drawn the attention of researchers because of its efficiency in solving real-world optimization problems arising in several application domains. The underlying biology behind the foraging strategy of *E. coli* is emulated in an extraordinary manner and used as a simple optimization algorithm. Also Bacterial Particle Swarm Optimization (BPSO) is presented in (Zhen *et al*., 2009), in which two strategies namely PSO and BFA combined. This study starts with a lucid outline of the basic PSO. Further it proceeds with Bacterial Foraging Particle Swarm Optimization (BFPSO). It then analyses the dynamics of the simulated chemotaxis step in BFPSO with the help of a simple mathematical model. Taking a cue from the analysis, it presents a new adaptive variant of BFPSO, where the chemotactic step size is adjusted on the run according to the current fitness of a virtual bacterium. Next, an analysis of the dynamics of reproduction operator in BFPSO is also discussed.

In our work, along with PSO and bacterial behaviours, certain constraints are formulated which provides an account of most of the significant performance metric in terms of completion time and energy consumption. The CBFPSO performs velocity updating and position updating in sequence according to PSO. The bacterial properties like reproduction and elimination dispersal are applied to CBFPSO for helping the particles to achieve faster convergence rate and jump out from local minima.

1.1. Related Work

Several algorithms have been proposed for multicore processor scheduling and optimization problems. Among them, population based meta heuristic algorithms such as GAs and PSOs exhibited promising solutions to handle this kind of complex problems. To cope with multicore processor environment, several techniques were introduced in EAs; to maintain the diversity of population throughout the run in multicore environments, a mechanism named as Multi-nichie crowding was used in (Cedeno and Vemuri, 1997). To recall useful information from past generation, memory based approaches were discussed in (Cedeno and Vemuri, 1997; Blackwell and Branke, 2004), which provide the knowledge attained in previous generations was usually helpful to the search in the next generation. Branke (1999), it has been argued that Evolutionary Algorithms (EAs), may be a particularly suitable candidate for static type of problems. Recently, many real world problems are dynamic, i.e., they change over time, has been explored in (Branke, 2001; Parsopoulus and Vrathatis, 2002; Hu and Eberhart, 2002; Carliste and Dozier, 2000). Blackwell (2003) multi-swarm optimization in dynamic environments, with new variants of PSO is designed. In this single population PSO and charged PSO are extended by constructing interacting multiswarms. Also a new algorithmic variant, which broadens, the implicit atomic analogy of CPSO to a quantum model is added. Du and Li (2008), a new multi strategy PSO for dynamic optimization, in which all particles are divided into two parts, denoted as part I and part II respectively and two new strategies, Gaussian local search and differential mutation are introduced in those two parts respectively and this algorithms outperforms other algorithms when the dynamic environment is unimodal and changes severely. Advanced computational intelligence based optimization algorithms; PSO and BFO have been implemented in (Patnaik and Panda, 2012), to tune the coefficients of PI controller to improve the power performance. In a heterogeneous data centers, to enable the power savings of idle servers with instantaneous workload, an adaptive power aware virtual machine provisioner which considers the resources dynamically is described in (Jeyarani *et al*., 2012). The scheduling of independent tasks an advanced parallel cellular genetic algorithm (Pinel *et al*., 2013) to minimize the make span on a fixed number of machines is presented. Two different ways of exploiting GPU parallelism are explored and evaluated in (Mussi *et al*., 2011) also to determine the execution speed of the two parallel algorithms is compared. Yang (2003), authors introduced the application of a new variation of GA called the Primal-Dual Genetic Algorithm (PDGA) for problem

optimization in nonstationary environments. A hybrid learning algorithm to improve the stability performance of a power system with Distributed Generations (DGs) is studied (Latha and Kanakaraj, 2013). Here the distribution system stability is maintained with reduced power loss using an Adaptive Neuro-Fuzzy Inference Systems (ANFIS) and Particle Swarm Optimization (PSO) techniques. Sudarmani and Kumar (2013) proposed a method which combines load balanced clustering, transmission power control over normal nodes present in the cluster and mobile sink over HSN. They have used PSO to find the optimal path for mobile sink to collect data from cluster heads.

The existing techniques on optimization based scheduling; they have not considered the performance such as completion time and energy consumption concurrently, which are instinctive in modern multicore processor scheduling. Therefore we have introduced a new approach in which the different constraints make evaluation regarding the task assignment on various cores based on their frequencies.

The rest of the paper is structured as follows, section 3 explains the problem formulation, section 4 discusses the swarm algorithms and section 5 provides the experimental analysis and results followed by conclusion and future work in section 6.

1.2. Problem Formulation

The task scheduling problem of multicore processor architecture is scheduling problem to partition the tasks between different cases by accomplishing minimum completion time and energy consumptive simultaneously. If M different core M = {ci, u = 1,2,…n) and T different tasks T = {tj, j = 1,2,….n) are considered in a heterogeneous environment, where every core works in different speed (frequencies) and processing capabilities.

As it is one of the important performance metric in heterogeneous multicore processor, the completion time of a specific task is important, because of its ability to describe the performance of the system. As a result minimizing the completion time of a particular task can be considered as a goal of the proposed scheduling algorithm, due to its significant role. As the environment considered is heterogeneous, it is also necessary, to consider the energy consumption of the core. Since both completion time and energy consumption are highly dependent on each after and should not be optimized independently. If the processing speed is V(i,j), the execution time has been calculated on the basis of size of the task by proceeding speed on different cores Equation 1:

$$\text{Completion Time (or) Processing Time} = \frac{\text{TaskSize}}{\text{Speed}}$$
$$\text{Max completion time } C_{max} = \max_{i=1}\left(\sum_{j=1}^{n} T(i,j) / V(i,j)\right) \quad (1)$$

The objective function is minimizing the completion Time, which is given by the Equation 2:

$$\text{Min}(\max) = \max_{i=1}\left(\sum_{j=1}^{n} T(i,j) / V(i,j)\right) \quad (2)$$

Let E be the Energy consumed while running a task with an average power P at the processor speed (or) frequency (f) for T time units, the relationship can be represented by the following:

- E=P (f)xT
- Objective function = $\alpha e^{-\beta z1} + \alpha e^{-\beta z2}$
- z1 = completion Time
- z2 = energy consumption
- α,β-constants

1.3. Swarm Algorithms

1.3.1. The Basic PSO Algorithm

Particle Swarm optimization is a heuristic, powerful optimization algorithm introduced by (Kennedy and Eberhart, 1995; Pinel *et al*., 2013). PSO is a kind of search mechanism to find the best solution by simulating the motion of a flock of birds or insects. The birds or insects are called as "particles', which can be generally expressed by a group of vectors as $(\vec{x}_1, \vec{v}_1, \vec{p}_1)$ where $\vec{x}_i = (x_{i1}, x_{i2}, ...x_{iD})$ and $\vec{v}_i = (v_{i1}, v_{i2}, ...v_{iD})$ (i = 1, 2,…m) represents the position and velocity of i^{th} particle respectively. The particles fly through the problem space by following the comment optimum particles.

PSO is finalized with a group of random particles (solutions) and then searches for optima by updates generations. During every interaction, each particle is updated by following two best values. The first best solution (fitness) of with respect to position is called 'pbest'. Another best position is called 'gbest' which is tracked by the particle swarm optimizer is the best position optimized so far, by any particle in the population.

After finding the two best values the particle updates its velocity and position according 4 steps (1) and (2) respectively:

$$V^i_{k+1} = wV^i_k + c_1r_1(pbest^i - X^i_k) + c_2r_2(gbest_k - X^i_k)$$
$$X^i_{k+1} = X^i_k + V^i_{k+1}$$

Where:

V^i_k = Velocity of the i^{th} particle at the k^{th} iteration
X^i_k = Current solution of the i^{th} particle at the k^{th} iteration
r_1, r_2 = Random members generated between 0 to 1
c_1, c_2 = Positive constants
w = is a positive inertia parameter

PSO is a simple concept and can be implemented easily with few parameters and this method provides desirable solution of PSO in optimization to some extents.

1.4. Proposed Algorithm (CBFPSO)

BF is a new bio-inspired algorithm-Bacteria have the tendency to gather around nutrient- rich areas by the activity called chemotaxis. The bacteria which fail to reach nutrient-rich areas may die due to lacking of nutrient while the others survive and reproduce the heat generation in nutrient rich areas. In the bacterial environment, some bacteria will be dispersed to random regions once their current living environment is no longer to live in.

The BEPSO combines both BF and PSO algorithms. This combination aims to make use of PSO ability to exchange social information and BF ability in finding a new solution by elimination and dispersal.

For initialization, select S, Ns, Nc, Nre, Ned, Ped, C1, C2, R1, R2 and (Ci), i = 1,2,…S. Also initialize the position $P^i_n, i = 1,2,....S$ and velocity randomly initialized. The BF-PSO made Bacterial population, Chemo-taxis, swarming, reproduction, elimination and dispersal articled by PSO is given below.

The (BF-PSO) combines both algorithms BF and PSO. This combination aims to make use of PSO ability to ex-change social information and BF ability in finding a new solution by elimination and dispersal. For initialization, the user selects S, N_s, N_c, N_{re}, N_{ed}, P_{ed}, C_1, C_2, R_1, R_2 and c(i), i = 1,2. . . S. Also initialize the Position $P^i_{n,1,1,1}$, i = 1,2. . . S and Velocity randomly initialized. The (BF-PSO) models bacterial Population Chemo-taxis, swarming, reproduction, elimination and dispersal oriented by PSO is given below (Initially, j = k = ell = 0). Implicit subscribes will be dropped for simplicity. The following **Table 1** describes the algorithm.

Table 1: Algorithm BFPSO

1. Initialize parameters n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, c (i)(i= 1,2. . . S), Delta, C_1, C_2, R_1, R_2. where,
• n: Dimension of the search space,
• S: The number of bacteria in the population,
• S_r: Half the total number of bacteria,
• N_s: Maximum number of swim length,
• N_c : Chemo tactic steps,
• N_{re}: The number of reproduction steps,
• N_{ed}: Elimination and dispersal events,
• P_{ed}: Elimination and dispersal with probability,
• c (i) : The step size taken in the random direction,
• C_1, C_2: PSO random parameter,
• R_1, R_2: PSO random parameter.
2. Generate a random direction Delta (n,i) and position
For (ell = 1 to N_{ed})
For (k = 1 to N_{re})
For (j = 1 to N_e)
For (i = 1 to S)
Evaluate the cost function
J(i,j) = Func (P(i,j))
Store the best cost function in J last
Jlast = J(i,j)
The best cost for each bacteria will be selected to be the local best J local
Jlocal(i,j) = Jlast (i,j)
Update position and cost function
P(i,j+1) = P(i,j) + c(i) * Delta (n,i)
J(i,j+1) = Func (P(i,j+1))
While (m<Ns)
If J(i,j+1)<j last
Then
Jlast = J(i,j+1)
Update position and cost function
P(i,j+1) = P(i,j+1) + c(i) * Delta (n,i)
J(i,j+1) = Func (P(i,j+1))
Evaluate the current position and local cost for each bacteria
Pcurrent (i,j+1) = P((i,j+1)
Jlocal (i,j+1) = Jlast (i,j+1)
else
Pcurrent (i,j+1) = P((i,j+1)
Jlocal (i,j+1) = Jlast (i,j+1)
end if
m = m+1
end while
next i (next bacteria)
Evaluate the local best position (Pl best) for each bacteria and global best position (Pgbest)
Evaluate the new direction for each bacteria.
V= ω *V+Cl*Rl (Plbest-Pcurrent) + C2 * R2 (Pgbest-Pcurrent)

Delta = V
next j (nest chemotactic)
for (i=1 to S)

$$Jhealth = \sum_{j=1}^{N_e+1}(i, j, k, ell)$$

end
The Sr bacteria with the highest J health remove and the other Sr bacteria with the best values copied.
next k (next reproduction)
With probability Ped, eliminates and disperse each bacterium
next all (next elimination)

As stated earlier to accomplish the real world dynamic applications, some constraints are appended into BFPSO. Among them constraints which can enhance the performance in terms of completion time and energy of CBFPSO is projected as the most interest.

1.5. Experimental Analysis

Experimental analysis in this section is designed to investigate the performance of the formulated constraints in terms of completion time and energy consumption. To investigate and test the performance characteristics of CBFPSO, we formulated four constraints. The constraints are based on the varying parameters such as frequency and task size. The first one is with medium tasks and uniform frequencies, second is with small tasks and varying frequencies, third is with medium tasks and varying frequencies and the fourth is large tasks with varying frequencies. It is also assumed in all the formulated constraints that when the core is not allotted to any task it enters into sleep mode. The CBFPSO algorithm is written in the MATLAB program environment. The input to the program is a design that consists of the number of cores. Each core is associated with the varying parameters such as frequency and task. For experimental purpose, these parameters are randomly assigned. The following **Table 1 and 2** describes the parameters chosen for GA and PSO.

1.6. Experiment: 1. Comparison of GA and CBFPSO in Detecting and Tracking Optimal Performance for Different Constraints.

This experiment is performed to find the effectiveness if the CBFPSO algorithm in detecting and tracking the changing optima for the different constraints, with varying number of cores. In this scenario, the completion time, is optimized such a way that, by assigning each constraint, by varying the number of cores. The changes of optimum value under different workload and different frequencies (constraints) were compared and plotted in **Fig. 1 and 2** resulting in the reduction of completion time with the proposed CBFPSO, where as the GA fails to do so. Also among all constraints out performs than others as its keep good balance convergence and diversity maintenance.

1.7. Experiment 2: Energy Performance Trade Off

The aim of the experiment is to find the optimum energy consumption for varying the number of cores with different constraints, which fulfils the performance such as completion time and energy consumption. As the different constraints are assumed with varying workload and frequencies the multicore has to meet different optima points. The performance curve is shown in **Fig. 3 and 4.** The curves show that the second, third and fourth constraint curves significantly outperform the first in terms of energy consumption, particularly when the busy cores are heavily loaded and are depicted in **Fig. 4.** This is due to the fact that assigning large tasks onto the heavily loaded core always search for high frequency core to handle the load which is not possible by the first constraint as the cores are with uniform frequencies, whereas in second, third and fourth constraints tasks are assigned such that the low frequency cores are loaded first and all the high frequency cores are fed into sleep mode results in significant energy consumption. Thus optimal CBFPSO results well than GA due to their global best and local best values.

Table 1. Parameters chosen for GA and PSO

GA	Parameters
Population size	100
Generation	50
Population type	Double vector
Crossover type	Arithmetic crossover
Crossover fraction	0.6
Mutation type	Uniform mutation
Mutation fraction	0.3

Table 2. Parameters chosen for BFPSO

BFPSO	Parameter
Population size	100
Dimension of search space	Number of cores
Number of bacteria	Number of cores
Number of reproduction step	20
Probability of each bacteria eliminated	0.25
PSO parameter C1	0.12
PSO parameter C2	1.2
PSO momentum or inertia	w = 0.9

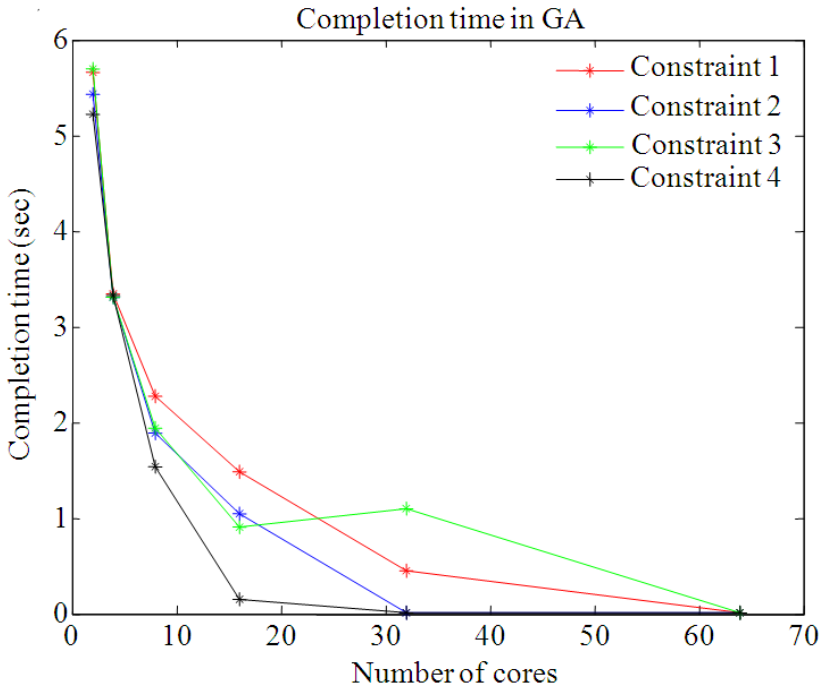

Fig. 1. Completion time with GA

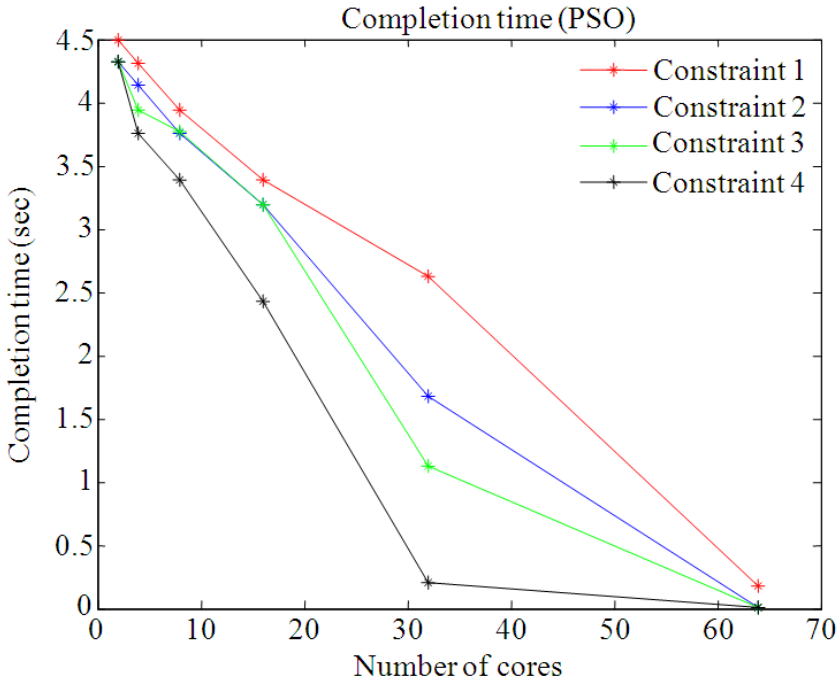

Fig. 2. Completion time with CBFPSO

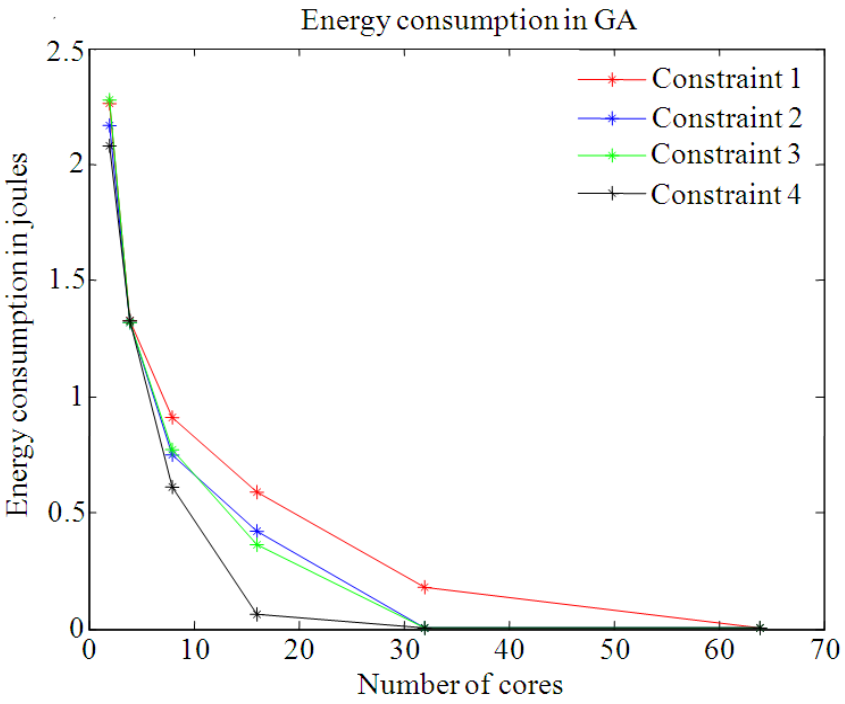

Fig. 3. Energy consumption with GA

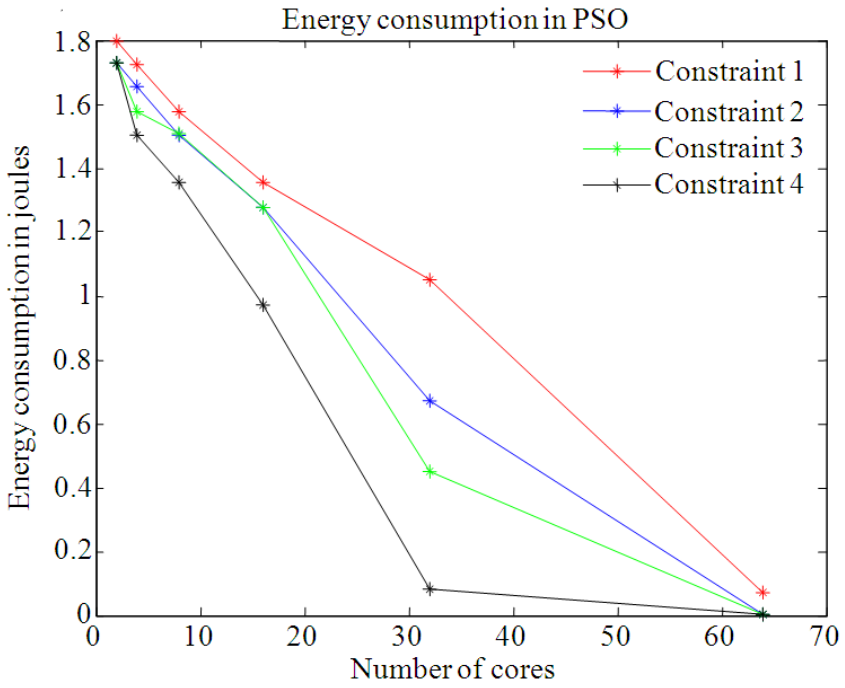

Fig. 4. Energy consumption with CBFPSO

In GA as the number of core increases the power consumption decreases exponentially because GA configuration parameter doesn't include number of core and task directly. In CBFPSO as the number of core increases the energy consumption decreases linearly and it is too small due to the PSO configuration parameter directly related with number of cores.

2. CONCLUSION

In this study, we presented the CBFPSO scheduling for the multi core processor and GA an extension of the preliminary work with different constraints which handles the tasks and frequencies under different conditions. The overall CBFPSO is described as an optimization problem with dual objective functions. The CBFPSO algorithm is better compared to GA as the former is self adaptive and preserves the fitness value corresponding to the optimal solution until it becomes superseded. The GA takes more time to find optimal solution as the re-randomization causes the particle to forget the memory and re-computation of fitness value brings computational overhead. The system is able to find new optimum solutions with the formulated constraints and the performance of the CBFPSO algorithm is well improved. As general reflection; the proposed system is a powerful mechanism and an important tool to enhance the searching capability of both genetic algorithms and CBFPSO in the optimization search. The key concern in designing the successful in CBFPSO algorithm is the solution representation as it describes a direct relationship between the problem domain and the bacterial particles. As a future work we focus on designing and implementing energy aware framework including more parameters in the constraints so that multicore processor can be used in modern datacenters.

3. REFERENCES

Abdel-Magid, Y.L., M.A. Abido, S. AI-Baiyat and A.H. Mantawy, 1999. Simultaneous stabilization of multimachine power system via genetic algorithms. IEEE Trans. Power Syst., 14: 1428-1439. DOI: 10.1109/59.801907

Abido, M.A. and Y.L. Abdel-Magid, 2002. Eigenvalue assignments in multi-machine power system using Tabu search algorithm. Comput. Elec. Eng., 28: 527-545. DOI: 10.1016/S0045-7906(01)00005-2

Abido, M.A., 2000. Robust design of multi-machine power system stabilizers using simulated anneleaing. IEEE Trans. Energy Conversion, 15: 297-304. DOI: 10.1109/60.875496

Blackwell, T. and J. Branke, 2004. Multi-swarm optimization in dynamic environments. Proceedings of the Environments EVO Workshops on Applications of Evolutionary Computing, April 5-7, Springer, Coimbra, Portugal, pp: 489-500. DOI: 10.1007/978-3-540-24653-4_50

Blackwell, T.M., 2003. Swarms in dynamic environments. Proceedings of the Genetic and Evolutionary Computer Conference, Jul. 12-16, Springer, USA, pp: 1-12. DOI: 10.1007/3-540-45105-6_1

Branke, J., 1999. Memory enhanced evolutionary algorithms for changing optimization problems. Proceedings of the Congress on Evolutonary Computation, Jul. 6-9, IEEE Xplore Press, Washington, DC., pp: 1875-1882. DOI: 10.1109/CEC.1999.785502

Branke, J., 2001. Evolutionary Optimization in Dynamic Environments. 1st Edn., Springer, ISBN-10: 0792376315, pp: 208.

Carliste, A. and G. Dozier, 2000. Adapting PSO to dynamic environments. Proceedingso of the International Conferecne on Artificial Intelligence, (CAI' 00).

Cedeno, W. and V.R. Vemuri, 1997. On the use of niching for dynamic landscapes. Proceedings of International Conference on Evolutionary Computation, Apr. 13-16, IEEE Xplore Press, Indianapolis, IN., pp: 361-366. DOI: 10.1109/ICEC.1997.592336

Du, W. and B. Li, 2008. Multi-strategy ensemble particle swarm optimization for dynamic optimization. Inform. Sci., 178: 3096-3109. DOI: 10.1016/j.ins.2008.01.020

Fogel, D.B., 1995. Evolutionary Computation: Toward a new Philosophy of Machine Intelligence. 1st Edn., IEEE Press, New York, ISBN-10: 0780310381, pp: 272.

Hu, X. and R.C. Eberhart, 2002. Adaptive PSO: Detective response to dynamic systems. Proceedings of the Cengness on Evluationary Computation, (EC' 02), pp: 1666-1670.

Jeyarani, R., N. Nagaveni and R.V. Ram, 2012. Design and implementation of Adaptive Power-Aware Virtual Machine Provisioner (APA-VMP) using swarm intelligence. Future Generat. Comput. Syst., 28: 811-821. DOI: 10.1016/j.future.2011.06.002

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Nov. 27-Dec. 01, IEEE Xplore Press, Perth, WA., pp: 1942-1948. DOI: 10.1109/ICNN.1995.488968

Latha, R. and J. Kanakaraj, 2013. Adaptive neuro-fuzzy inference system-particle swarm optimization based stability maintenance of power system networks. Am. J. Applied Sci., 10: 779-786. DOI: 10.3844/ajassp.2013.779.786

Mussi, L., F. Daolio and S. Gagnoni, 2011. Evaluation of parallel particle swarm optimization algorithms within the CUDA architecture. J. Inf. Sci., 181: 4642-4657. DOI: 10.1016/j.ins.2010.08.045

Parsopoulus, K.E. and M.N. Vrathatis, 2002. Recent appreciates to global optimization problems through particle. Swarm Opitme Natural Comput., 1: 235-306. DOI: 10.1023/A:1016568309421

Passino, K.M., 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Mag., 22: 52-67. DOI: 10.1109/MCS.2002.1004010

Patnaik, S.S. and A.K. Panda, 2012. Particle swarm optimization and bacterial foraging optimization techniques for optimal current harmonic mitigation by employing active power filter. Applied Computational Intell. Soft Comput. DOI: 10.1155/2012/897127

Pinel, F., B. Dorronsoro and P. Bouvry, 2013. Solving very larger instances of the scheduling of independent tasks problem on the GPU. J. Parallel Distributed Coput., 73: 101-110. DOI: 10.1016/j.jpdc.2012.02.018

Sudarmani, R. and K.R.S. Kumar, 2013. Particle swarm optimization-based routing protocol for clustered heterogeneous sensor networks with mobile sink. Am. J. Applied Sci., 10: 259-269. DOI: 10.3844/ajassp.2013.259.269

Yang, S., 2003. Non-stationary problems optimization using the primal-dual genetic algorithm. Proceedings of the IEEE Congress on Evolutionary Computation, Dec. 12-18, IEEE Xplore Press, pp: 2246-2253. DOI: 10.1109/CEC.2003.1299951

Zhen, J., W. Yiwei, C. Ying and W. Qinghua, 2009. Bacterial particle swarm optimization. Chinese J. Elec.