MATEC Web of Conferences **119**, 01047 (2017) DOI: 10.1051/matecconf/201711901047
IMETI 2016

A hybrid Big Bang-Big Crunch algorithm for energy optimization on heterogeneous distributed system

Yan Kang[1,a], Hao Li[2], Chunhui Wang[1] and Li Dai[1]

[1]*Department of Software Engineering, School of Software, Yunanan University, Kunming, 650091, China*
[2]*Department of Network Engineering, School of Software, Yunanan University, Kunming, 650091, China*

Abstract. A hybrid algorithm is presented for global optimization of the energy consumption rather than makespan of the system. The cost profile is optimized by extending the execution time of the tasks on dynamic voltage scalable processing elements in embedded environment with meeting the execution constraints. The modified Big Bang Big Crunch (BBBC) method improves the scheduling efficiency of the tasks by simulating one of the theories of the evolution of the universe. BBBC algorithm generates disorder data points inspired by energy dissipation procedure of the Big Bang phase, and moves those data points to a single representative data point inspired by a center of mass cost approach in the Big Crunch phase. The Logistic and Sinusoidal chaotic maps are investigated and utilized to improve the movement step of the BBBC algorithm. The proposed hybrid algorithm is tested on several benchmark data sets and its performance is compared with those of Ant Colony Optimization and HEFT strategies. The simulation experiment shows that the presented evolutionary optimization algorithm is robust and suitable for energy saving.

1 Introduction

Energy efficiency has become an important issue of the distributed system especially the embedded system which is composed of processing elements (PES) normally powered by batteries. There exist slacks among the system since the process ability of the processor is usually not fully utilized. The processors that enabled dynamic voltage scaling (DVS) strategy switches off gated clocks for unused circuit parts during idle periods and then reduce the power consumption. The power consumption is at least a quadratic function of the supply voltage (hence CPU speed) in the cased of function executed. It was shown in [1,2,3] that further energy consumption can be reduced by decreasing process frequencies during the voltage selection. Investigations of DVS processors have shown that the power consumption can be decreased by up to 10 times after using voltage scales when executing real-life applications [4].

DVS results in the different processing elements characteristics if utilize the slacks to extend the execution time of the processors. DVS decreases the energy consumption to extend the processing time of the tasks that have the slacks between the deadline and its completion time as slowly as possible in the case of the slacks. Yao. [5] Presented one of the earliest theoretical models for DVS, and presented an $O(n^3)$ algorithm for obtaining an optimal schedule and computing a characterization of the minimum-energy DVS schedule. The schedule gave a main benchmark for referencing other scheduling

[a] Corresponding author : kangyang@ynu.edu.cn

algorithms in both theoretical and simulation aspects under the assumption of convexity of the power consumption function.

There are static and dynamic techniques to using dynamic voltage scaling to decide appropriate operating voltage/speed. Static techniques take advantage of offline parameters, such as periods and worst-case execution cycles while dynamic techniques (based on slack reclamation) exploit early completions of tasks to further decrease the speed and energy consumption. Here, we investigate static one. Dynamic methods [6,7] normally reduce the speed based on these predicted values and achieve more energy saving than static ones. But the interest in static techniques is still high since static approaches can be enhanced to develop dynamic ones.

Three important co-synthesis steps are: a) Mapping: Determining the assignment of computational tasks to PEs and data transfers to communication links (CLs), b) Scheduling: Determining the execution order (sequencing) of tasks mapped to PEs and communications to CLs, and c) Evaluation: Determining the quality of the implementation candidate (timing feasibility, cost, power, area, etc.).

Pillai and Shin [8] computed a single optimal speed off-line and obtained the minimal speed to make a task set schedulable without changing it under Earliest Deadline First, and proposed a near-optimal method under Rate Monotonous. Without fixing the processor speed, Saewong and Rajkumar [9] assumed that the speed of the processor can be changed continuously in a given range and presented an algorithm to find the optimal speed value based on fixed priority assignments. Based on the task parameters, some methods in [10] were proposed to decide statically processor speed and assign different speed to every task before system execution. A more general scheme in [11] chose the speed switching instants more freely which typically occur at the activation/deadline of some job.

Classical programming approaches such as Non-linear programming (NLP), Linear programming (LP) and Dynamic programming methods are utilized to solve NP hard problem at the cost of increased dimensionality and depends on mathematical formula of the cost function [12]. Heuristics algorithms as the recent advances in AI techniques are often used as complementary approach to solve complex optimization problems such as NP hard problem. Heuristic techniques pave the way towards finding quality solutions to the NP hard problem during tolerable time from the inspiration of natural behavior of various social/natural procedures. And the heuristic methods are easy to transform and adapt according to specific problem. Genetic algorithms [13], Swarm intelligence[14], Bacteria foraging, ant colony search techniques are population based and stochastic in nature, and are utilized to find global/near global solutions to optimization problems.

As another population based algorithm, Big Bang and Big Crunch (BBBC) developed by Erol and Eksin [15] are inspired by the concept of universal evolution. BB-BC method has been proved to outperform genetic algorithm for benchmark test functions [16]. Big Bang-Big Crunch (BBBC) algorithm, one of the most recent heuristic optimization strategies, is utilized to solve optimization problem by simulating the Big Bang phase and Big Crunch phase during the evolution process of the universe. According to the Big Bang and Big Crunch theory, disorder and randomness are generated by energy dissipation during the Big Bang phase, and randomly distributed particles are drawn into an order during the Big Crunch phase. Based on the good performance in exploiting the local optimization solution during the BBBC evolution process, new solution is obtained as the influx of random solutions by means of a center of mass or minimal cost approach. The search strategy of cannot explore the global optimization effectively despite. The studies about BBBC optimization are very few as considering the premature convergence and the convergence speed of BBBC approach. This paper aims to improve the optimization performance of BBBC optimization by combining the self-adaptive operator and competitive strategy.

This paper is organized as follows: The next section briefly describes the problem model which includes task model and energy model. In Section 3, hybrid algorithm based on BBBC is presented while its performance is studied in section 4.

2 Task model and energy model

The energy optimization problem of the tasks can be divided into two problems: the task scheduling problem and the DVS problem. The task scheduling problem is to order a set of tasks and assign them to different processors of a distributed system for accomplishing an application. The heterogeneous of the system implies the processing time of each task may not be the same on different processors.

The task scheduling problem can be formulated as follows. There is an application consisting of n tasks at_i (i=1, 2,…,m) which processes on a set of processors ap_j (j=1,2,…,n)according to a predetermined order. Each task at_i is ordered by its priority and processed on a specific processor ap_j according to the earliest ending time. The task scheduling problem is thus to determine the routing sequence and the assignment of the tasks on the processors so that precedence constraint among tasks is satisfied. In this paper, the tasks and their executing constraints are depicted as the nodes and the directed edges of a Directed Acyclic Graph (DAG). There is an entry task and an exit task to imply the start and end of all the tasks. The directed edge implies starting point (the processor task) must complete before the beginning of its end point (successor task). When the task and its successor task are assigned on different processors, a communication time will be needed for transferring the data between two tasks. For simplification, the communication time is assumed the same among the whole communication network.

The power consumption is at least a quadratic function of the supply voltage. Here the static power is ignored and dynamic power is studied and reduced by extend the execution time of the tasks based on its slack. The relation between dynamic power dissipation and supply voltage is computed as Eq.1

$$P_v = C_f \cdot V_s^2 \cdot fc \quad P_v = C_f \cdot V_s^2 \cdot fc \tag{1}$$

where P_y, C_f, V_s, fc are the dynamic power, the workload capacitance, the supply voltage and the clock frequency, respectively.

And fc is computed according to Eq. 2

$$fc = k \cdot (V_s - V_k)^2 / V_{ds} \tag{2}$$

where k is a constant, and V_k is the threshold voltage. The rate between the clock frequency and the threshold voltage is a constant.

According to Eq. 1 and 2, the power consumption of a processor is obtained according to Eq. 3

$$E = P_{dv} \cdot t = C_{ef} \cdot V_{ds}^2 \cdot c \tag{3}$$

Normally, the goal in an energy optimization of task scheduling problem is to find a feasible schedule with the minimum energy consumption. For a feasible schedule of the task scheduling problem, the execution time of all the tasks sequence fulfilling the precedence constraint are adjusted for reducing the energy consumption. Several symbols are introduced as follows:

- ec_i: the energy consumption of the task at_i
- dt_i: the deadline of the task at_i
- et_{ij}: the execution time of task at_i which is assigned on processor ap_j
- prt_i: the priority of task at_i
- ef_i: the earliest finish time of task at_i
- lf_i: the latest finish time of task at_i
- ppt_i: the directly predecessor of task at_i
- spt_i: the successor of task at_i on the processor
- st_i: the directly successor of task at_i
- slt_i: the slack of task at_i
- ot_i: is the order task at_i in the schedule
- pt_i: the processor which execute task at_i
- Sch_k: kth schedule which is composed of ot_i and pr_i

- v_{imax}: the maximum voltage of task at_i
- v_{is}: the scaling voltage of task at_i
- EC_k: is the initial energy consumption of kth schedule
- AEC_i: is the adjusted energy consumption of kth schedule after using DVS strategy
- MS_i: the makespan of the schedule, i.e., the finish time of the last task at_m

Consider the feasibility of the solution in real time systems, the execution of the task cannot be interrupted, and the finish time of each task should be less than its deadline as $ef_i <= dt_i$.

3 Hybrid BBBC algorithm

3.1 BBBC algorithm

BBBC optimization includes Big Bang procedure and Big Crunch procedure. The Big Bang procedure is a random phase that simulates the energy dissipation in nature, one of the theories of the evolution of the universe. The following Big Crunch procedure is a contraction phase that calculates a center of mass for the population viewed as gravitational attraction [17]. In this way, the disorder solutions are generated over the entire search, and then the solutions convergence to a local or global optimum point iteratively. Big Bang and Big Crunch phases are firstly obtained random candidates distributed over the center of mass or the minimum candidate, and then compute the center of the candidates. The pseudocode of BBBC is given as follows:

Begin
k=1, random candidates are generated as the kth generation.
While stopping criteria not met **Do**
Pc, Center of mass of candidates, is obtained by Big crunch procedure.
k+1th generation is generated around Pc by Big bang procedure.
Evaluate k+1th generation.
k++.
End while
End

3.2 Population initialization

The scheduling situation of all the tasks is obtained according to a routing pattern and an ordering pattern alternatively. The first pattern demonstrates the assignment of the tasks, and the second pattern guarantees the execution order of the tasks on the heterogeneous processors. A candidate is represented as a schedule, Sch={OT,PT} that each task and processor is assigned a fixed ID. A random solution is obtained as the combination of two random orders of (1,2,..,m) and (1,2,..,n) respectively. The solutions to task scheduling problem can be represented as a population which is composed by a set of alternatives or candidates. At the beginning of the evolution, some list algorithms as HEFT are utilized in a hybrid way to generating various initial solutions with relative good quality [18]. The HEFT algorithm [19] is known as good trade off between schedule length and search effort as its time complexity is low as (O(nlogn+(e+n)p)).

To generate the initially promising schedule, the tasks are assigned onto processor with minimum earliest finish time. The insertion technique is used to insert the task into the earliest available time on its assigned processor. The remaining tasks are selected according to the descending order of the priority.

3.3 Objective function computation

For a feasible schedule of the task scheduling problem, the objective of the problem is to minimize the energy consumption. The energy consumption is obtained by extending the execution time of the tasks based on using DVS strategy on their slacks. The slack of task at_i, slt_i, equals difference between ef_i and

lf_i. Lf_m of the last task at_m is initialized as its deadline, and lf_i (i=1,2,..,*m 1*) is calculated recursively from the last task as shown in Eq.4:

$$lf_i = \min\{ lf_{st_i} - et_{ij}, lf_{spt_{ij}} - et_{ij} \}, j \in pt_i \tag{4}$$

Here method for minimizing total dynamic energy dissipation are based on variable voltage systems (nearly continuous range of possible supply voltages), that can be adjusted to multi voltage systems (small and limited number of potential supply voltages) easily [20]. AEC_i of task ta_i is obtained by the optimization strategy according to the PE power consumption and supply voltage optimization as Eq.5:

$$AEC_i(et_{ij} + \Delta t) = EC_i(t) \cdot vt_{is}^2 / vt_{i\max}^2 \tag{5}$$

where $\Delta t = \min\{slt_i\}/C$,C is a constant, and v_{is} is computed as Eq.6:

$$V_{is} = V_{it} + V_{ic}/2 * d_i + \sqrt{(V_{it} + V_{ic}/2 * d_i)^2 - V_{it}^2} \tag{6}$$

where $d_i = (et_{ij} + \Delta t)/et_{ij}$, V_{ic} is a constant and is computed as Eq.7:

$$V_{ic} = (V_{i\max} - V_{it})^2 / V_{i\max} \tag{7}$$

The task with high energy saving is adjusted firstly according the slack with satisfying deadline and precedence constraints. The procedure of extending execution time of the task is repeated until no slack existing.

3.4 Modified BC phase

The mass is calculated as the inverse of the energy consumption and makespan value respectively as the population is divided into two groups. With viewing ot_i as the position of task at_i, the convergence operator of the Big Crunch phase generates the center of mass (Pco_i, Pcp_i). $Pco_i(E)$ is computed according to Eq. 8:

$$Pco_i(E) = (\sum_{j=1}^{Num} ot_i / E_j) / (\sum_{j=1}^{Num} 1 / E_j), i = 1, \cdots, m \tag{8}$$

where *Num* is the population size of the feasible solutions.

$Pcp_i(E)$ is computed as Eq. 9:

$$Pcp_i(E) = (\sum_{j=1}^{G} pt_i / E_j) / (\sum_{j=1}^{G} 1 / E_j), i = 1, \cdots, m \tag{9}$$

where *E* equals makespan and energy consumption respectively for each group. The fit candidates with minimum makespan and energy consumption respectively are also viewed as the starting points in the Big Bang phase.

3.5 Modified BB phase

New shcedule (ot_i, pt_i) is randomly generated around the starting point obtained by BC phase.
Firstly pt_i is computed as Eq. 10:

$$pt_i = (Pcp_i + \alpha_i \cdot \gamma_i) \bmod m, i = 1, \cdots, n \tag{10}$$

where α_i is a random real less than 1. r_i is the standard deviation of a normal distribution, and is computed as Eq. 11:

$$\gamma_i = speed1 \cdot (n-1) / iter \tag{11}$$

where *iter* represents the number of iterations, *speed*1is an parameter to control convergence speed of the search.

Secondly the set of ot_i is obtained by in ascending order to sorting pt_i which is computed as Eq.12:

$$pt_i = (Pcp_i + \alpha_i \cdot \beta_i) \bmod n, i = 1, \cdots, n \tag{12}$$

where β_i is computed as Eq.13:

$$\beta_i = speed_i \cdot (pt_{\max} - pt_{\min}) / iter \tag{13}$$

where pt_{max} and pt_{min} are the upper and lower bounds of the variable pt_i, and $speed_i$ is an parameter acting as *speed*. It is noted that the set of ot_i maybe infeasible as it disobeys the precedence constraint. An adjust strategy need to recalculated pt_i recursively from the task as Eq.14:

$$pt_i = \max_{j \in ppt_i} \{pt_j + cc_i, pt_i\}, i = 1, \cdots, n \tag{14}$$

where cc_i is a constance equal to 1.

3.6 Chaos based BB phase

Meanwhile, to guarantee the diversity of the initial solution, a chaos strategy is applied to randomly order the tasks and generate their assigned machines. The non repetition is obtained by mathematics formula defined by chaos strategy that has been successfully applied in various research areas as physics and engineering [21]. Chaotic based functions have utilized to help heuristic algorithms such as imperialistic competitive algorithm [22], harmony method [23], charged system search strategy [24] to trade off between exploration and exploitation. Chaos strategy is incorporated with BBBC algorithm to improve the diversity among candidates and away from premature convergence. In this paper, logistic and sinusoidal chaotic map function are utilized to replace parameter cc_i and $speed_i$ in Eq. 13, and 14 respectively as Eq. 15 and 16.

$$cc_{i+1} = a1 \cdot cc_i (1 - cc_i) \tag{15}$$

$$speed_{i+1} = \alpha 2 \cdot speed_i^2 \sin(\pi speed_i) \tag{16}$$

where $a1$ is a constant equals to 4, *cc* is initialized as a real number between 0 and 1, and cannot equals 0, 0.25, 0.5, 0.75, and 1. $A2$ is a constant equals to 2.3, *speed* is initialized as 0.7.

The Hybrid BBBC (HBBBC) algorithm is described as follows:

Input: A set of tasks to be assigned on heterogeneous processors.

Output: A task schedule with adjusted execution voltage.

1: the population composed of *Num* candidates is initialized by HEFT and random strategy;
2: compute the makespan, slack energy consumption of each candidate;
3: new energy consumption of each candidate is obtained by using Eq. 5;
4: **while** the stopping criteria are not met **do**
5: the feasible candidates are selected as each task satisfying the precedence and deadline constraint;
6: the center of mass (Pco_i, Pcp_i) is computed as Eq. 8 and 9 with *E=MS*;
7: the center of mass (Pco_i, Pcp_i) is computed as Eq. 8 and 9 with *E=EC*;
8: new candidates are generated by Eq. 10 and 12 with the starting point (Pco_i, Pcp_i);
9: the adjust strategy described as Eq.14 guarantees the feasibility of the candidates;

10: Chaos BB strategy changes candidates by modifying the parameters in Eq. 13 and 14;
11: compute the makespan and energy consumption of each candidate;
12: new energy consumption of each candidate is obtained by using Eq. 5;
13: **end while**

4 Experiment and Result

The extensive experiments are implemented to evaluate HBBBC algorithm which was programmed in java and run on Intel Core TM processors with 2 GHz speed with 4GB of memory. HEFT scheduling algorithm that proved to perform well for the task scheduling problem and our previously proposed ACO [25] are also implemented for verifying our algorithm, the energy saving of HBBBC is enabled by the exploitation of the DVS technique.

4.1 Experimental settings

Each method is run 10 times with different initial control variables for 100 generations. The best results of each run are presented. The performance of HBBBC was thoroughly evaluated with two test cases: randomly and benchmarks as three real world parallel applications: Fast Fourier Transformation [26], the Ludecomposition [27] and the Laplace equation solver [28]. A random tasking scheduling problem generator is designed to generate various task graphs with different characteristics.

Specifically, *m* is generated between 10 and 200. The ratio of the computational and communication value of the tasks is randomly generated as less 4 or larger than 1 that represents the application with larger computation or communication costs respectively. The maximum and minimum execution time of each task is defined to describe difference between the fastest processor and the slowest processor in a given system.

4.2 Results

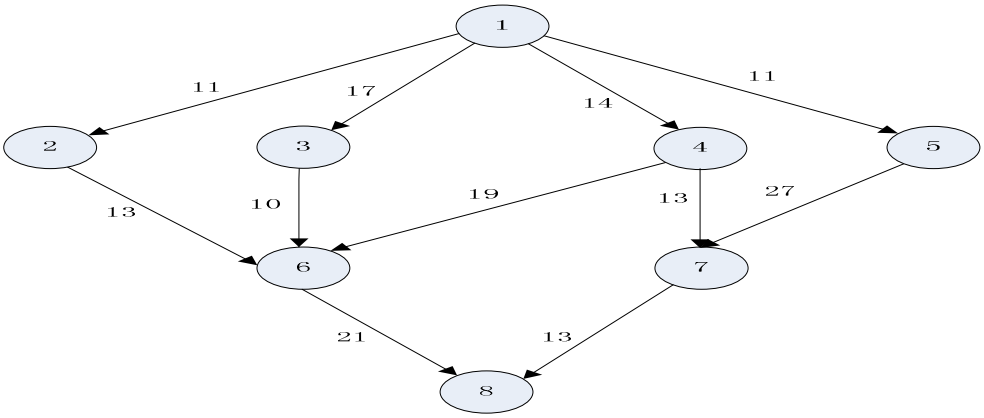

Figure 1. Example of a task graph with 8 tasks.

Table 1. Computation cost of Fig 1.

at	1	2	3	4	5	6	7	8
P1	11(1)	10(17)	9(23)	11(24)	15(35)	12(37)	10(61)	11(39)
P2	13(21)	15(24)	12(31)	16(27)	11(41)	9(43)	14(23)	15(41)
P3	9(3)	11(2)	14(2)	10(20)	19(37)	5(25)	13(66)	10(40)

Here, test firstly is implemented on a benchmark set. Figure 1 demonstrates a DAG with nine tasks and 13 edges labelled the communication times that remain the same on the all communication edges. Figure 2 and Figure 4 show the schedules of the tasks in Figure 1 by HEFT and HBBBC strategies on three processors respectively based on the computation cost (energy) described in Table 1. Figure 3 and Figure 5 show the energy saving by DVS strategy based on schedules shown in Figure 2 and Figure 4. If the deadline is extended to 150, the energy consumption is decreased from the 2855.0 to 393.19257 by using the DVS strategy on the schedule obtained by HEFT strategy. And the energy consumption is decreased from 1723.0 to 152.40613 by using the DVS strategy on the schedule obtained by HBBBC strategy.

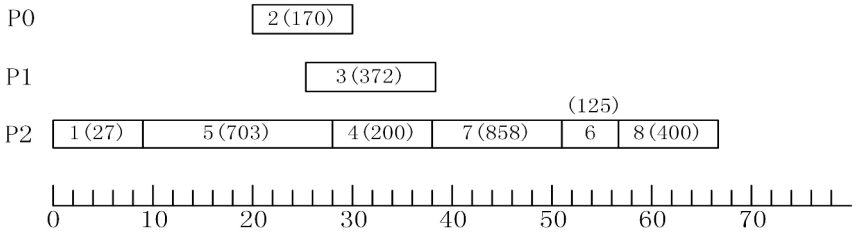

Figure 2. The schedule of Fig. 1 by HEFT.

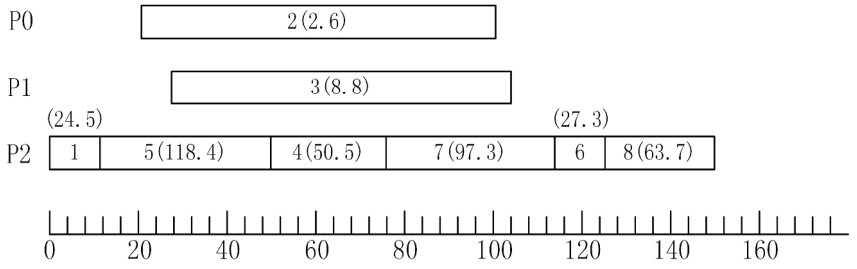

Figure 3. The energy saving of the schedule in Fig 2.

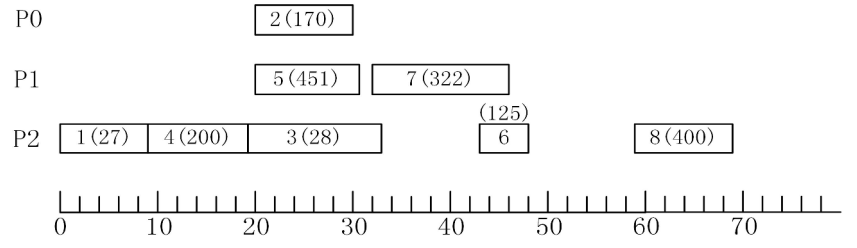

Figure 4. The schedule of Fig 1 by HBBBC.

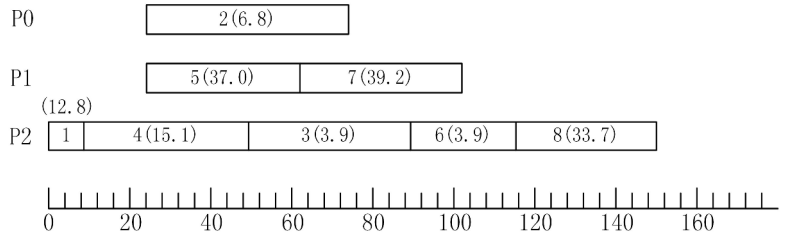

Figure 5. The energy saving of the schedule in Fig 4.

Table 2 presents the energy saving results obtained from our extensive comparative evaluation study among HBBBC, HEFT and our previously proposed ACO algorithms. Simulations have been conducted using the simulator we developed as part of this study as HEFT is energy unconscious and is not directly applicable to DVS problem.

Table 2. Comparative results

Algorithm over DAG set	HBBBC over ACO	HBBBBC over HEFT
FFT	21%	22%
Laplace	15%	25%
LU	13%	19%
Random	20%	26%

5 Conclusion

Since most traditional algorithms only focus on makespan, we presented HBBBC to investigate the energy consumption and task scheduling altogether. The energy will be saved by adjusting slack and execution time obtained by schedule algorithm. Our algorithm hybridizes HBBBC algorithm and dynamic scaling strategy to minimize the energy consumption of the tasks on the heterogeneous distributed mobile system. An adjust strategy is used to guarantee the solutions meeting the precedence constraints, and the feasible solutions are selected to generate around the centric mass in BC phase. The centric mass is treated as the starting point for generate new random solutions around it in BB phase. In essence, the energy saving of dynamic scaling strategy is incorporated with the scheduling strategy based on BBBC and chaos strategy. Chaos map functions are applied to trade off the exploration and exploitation. The computational results on a set of random and specific instances testify the performance of HBBBC. Our study provides promising results showing the significance and potential of hybridizing the task scheduling and dynamic scaling strategy in the reduction of energy consumption.

References

1. T. Ishihara and H. Yasuura, Voltage scheduling problem for dynamically variable voltage processors, *Proceedings of International Symposium on Low Power Electronics and Design*, 197-202 (1998)
2. A. Manzak and C. Chakrabarti, Variable voltage task scheduling for minimizing energy or minimizing power, *Islped Proceedings of the International Symposium on Low Power Electronics and Design*, **6** (2), 3239-3242 (2000)
3. M.S. Al Hashimi, Considering power variations of DVS processing elements for energy minimisation in distributed systems, International Symposium on System Synthesis, 250-255 (2001)
4. T. Burd, T. Pering, A. Stratakos, and R. Brodersen, A dynamic voltage scaled microprocessor system, IEEE International Solid state Circuits Conference, **35** (11), 1571-1580 (2000)
5. F. Yao, A. Demers, and S. Shenker, A scheduling model for reduced CPU energy, Foundations of Computer Science Annual Symposium, **23** (25), 374-382 (1995)
6. A. Qadi, S. Goddard, and S. Farritor, A dynamic voltage scaling algorithm for sporadic tasks, IEEE Real time Systems Symposium, 52-62 (2004)
7. C. Scordino and G. Lipari, A resource reservation algorithm for power aware scheduling of periodic and aperiodic real time tasks, Computers IEEE Transactions, **55** (12), 1509-1522 (2007)
8. P. Pillai and K. Shin, Real time dynamic voltage scaling for low power embedded operating systems, Acm Sigops Operating Systems Review, **35** (5), 89-102 (2001)
9. S. Saewong and R. Rajkumar, Practical voltage scaling for fixed priority RT systems, IEEE Real time and Embedded Technology and Applications Symposium, 106-114 (2010)

10. Y. Liu and A. Mok, An integrated approach for applying dynamic voltage scaling to hard real time systems, IEEE Real time and Embedded Technology and Applications Symposium, 116-123 (2003)
11. H. Aydin, V. Devadas, and D. Zhu, System level energy management for periodic realtime tasks, *Proceedings of IEEE 27th Real Time Systems Symposium*, 313-322 (2006)
12. N. Sinha, R. Chakrabarti, and P. Chattopadhyay, Evolutionary programming techniques for economic load dispatch, IEEE Transactions on Evolutionary Computation, **7** (1), 83-94 (2003)
13. V. Bilolikar, K. Jain, and M. Sharma, An adaptive crossover genetic algorithm with simulated annealing for multi mode resource constrained project scheduling with discounted cash flows, International Journal of Operational Research, **25** (1), 28-46 (2016)
14. A. Meng, Z. Li, H. Yin, S. Chen, and Z. Guo, Accelerating particle swarm optimization using crisscross search, Information Sciences, **329**, 52-72 (2015)
15. O. Erol and I. Eksin, A new optimization method: big bang-big crunch, Advances in Engineering Software, **37** (2), 106-111 (2006)
16. B. Alatas, Uniform big bang-chaotic big crunch optimization, Communications in Nonlinear Science and Numerical Simulation, September, **16** (9), 3696-3703 (2011)
17. C. Rao and G. Yesuratnam, Big bang and big crunch (BB BC) and firefly optimization (FFO): application and comparison to optimal power flow with continuous and discrete control variables, American Journal of Nursing, **104** (9), 26-26 (2004)
18. H. Arabnejad and J. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Transactions on Parallel and Distributed Systems, **25** (3), 682-694 (2014)
19. H. Topcuouglu, S. Hariri, and M.Y. Wu, Performance effective and low complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Dist. Systems, **13** (3), 260-274 (2002)
20. M. Schmitz and B. Al, Iterative schedule optimization for voltage scalable distributed embedded systems, Acm Trans. Embedded Comput. Syst, **3** (1), 182-217 (2004)
21. M. Tavazoei and M. Haeri, Comparison of different one dimensional maps as chaotic search pattern in chaos optimization algorithms, Applied Mathematics and Computation, **187** (2), 1076-1085 (2007)
22. S. Talatahari, B. Azar, R. Sheikholeslami, and A. Gandomi, Mperialist competitive algorithm combined with chaos for global optimization, Communications in Nonlinear Science & Numerical Simulation, **17** (3), 1312-1319 (2012)
23. Q. Pan, L. Wang, and L. Gao, A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers, Applied Soft Computing, **11** (8), 5270-5280 (2011)
24. S. Talatahari, A. Kaveh, and R. Sheikholeslami, An efficient charged system search using chaos for global optimization problems, International Journal of Optimization in Civil Engineering, **1** (2), 305-325 (2011)
25. Y. Kang, An ant colony system for dynamic voltage scaling problem in heterogeous system, Lecture Notes in Electrical Engineering, **277**, 73-81 (2013)
26. R.E. Lord, J.S. Kowalik, and S.P. Kumar, Solving linear algebraic equations on an mimd computer, J. ACM, **30** (1), 103-117 (1983)
27. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press (1990)
28. M.Y. Wu and D.D. Gajski, Hypertool: a programming aid for message passing systems, IEEE Trans. Parallel and Distributed Systems, **1** (3), 330-343 (1990)