Real-time task scheduling in heterogeneous multiprocessor systems using artificial bee colony

Mohammad Shokouhifar[1], Ali Jalali[2]
Department of Electrical and Computer Engineering
Shahid Beheshti University
Tehran, Iran
[1]m.shokouhifar@sbu.ac.ir, [2]a_jalali@sbu.ac.ir

Abstract—**Scheduling of real-time tasks in multiprocessor systems is a NP-hard problem. Recently, swarm intelligence algorithms have been efficiently applied for this problem. Real-time tasks can be classified into hard real-time tasks and soft real-time tasks. The aim of hard real-time task scheduling algorithms is to meet all tasks deadline constraints. However, slight violation is not critical, in the case of soft real-time tasks. In this paper, a new algorithm based on artificial bee colony (ABC) is proposed for scheduling of soft real-time tasks. In this method, a hybrid neighborhood search mechanism is introduced to improve the convergence of ABC. Experimental results demonstrate the effectiveness of proposed algorithm for scheduling of soft real-time tasks in heterogeneous multiprocessor systems.**

Keywords-heterogeneous multiprocessor systems; real-time task scheduling; local search; global search; artificial bee colony;

I. INTRODUCTION

The multiprocessor computing environment is well suited to meet the computational demands of diverse tasks [1]. Real-time tasks can be classified into two categories: hard real-time tasks and soft real-time tasks. In hard real-time systems (e.g. patient monitoring, nuclear plant control, etc), the violation of timing constraints of the tasks is not acceptable. On the other hand, slight violation of timing constraints is not critical in the case of soft real-time systems (e.g. telephone switching, image processing, etc). The usefulness of a soft real-time system decreases over time after the deadline expires without causing any damage to the controlled environment [2].

In multiprocessor platforms, the objective of scheduling is not only to optimize an execution order of tasks, but also to determine the specific processor to be used for each task. In homogeneous multiprocessor systems, all processors are identical. The problem becomes more difficult, in the case of heterogeneous multiprocessors. Additional complexity appears from the fact that each task needs different execution times upon each processor. For example, an intensive mathematical calculation may execute much faster on a floating-point coprocessor than a digital signal processor [1].

Typically, Rate Monotonic (RM) and Earliest Deadline First (EDF) are applied for task scheduling in hard real-time uniprocessor systems [3], which guarantee the optimality of the solution. However, these algorithms have some drawbacks in coping with soft real-time systems with overloaded situation. In soft real-time systems, the objective is to minimize the total tardiness. Rate Regulating Proportional Share (RRPS) [4] and Modified Proportional Share (MPS) [5] have been proposed for task scheduling in soft real-time systems. However, these algorithms also cannot cope under an overloaded situation and are restricted only for uniprocessor systems.

Real-time task scheduling in multiprocessor systems is more difficult than that in uniprocessor systems. It is a NP-hard problem [6]. Recently, swarm intelligence algorithms have been efficiently applied for scheduling problems. Mitra et al. presented a genetic algorithm (GA) implementation to solve the scheduling problem for non-preemptive tasks with precedence and deadline constraints in a multiprocessor environment [7]. Lin et al. proposed a hybrid GA, where different operators are applied for scheduling of non-preemptive tasks in soft real-time systems [8]. Monnier et al. introduced a GA for scheduling of real-time non-preemptive tasks [9]. In these algorithms [7-9], only one specific objective function (e.g. cost, completion time, total tardiness, etc) is considered to optimize the assignment problem.

Also, some multi-objective solutions based on GA have been introduced. Oh et al. proposed a multi-objective GA for scheduling of non-preemptive tasks in multiprocessor platforms [10]. Also, a static scheduling [11] and a dynamic scheduling [12] using GA for scheduling of real-time tasks on heterogeneous systems are proposed. Dhodhi et al. [13] also presented a GA for task scheduling on heterogeneous systems, where a new encoding has been used to represent the feasible solutions. It is remarkable that these GA-based multi-objective algorithms [10-13] were implemented for the general tasks without considering time constraints. Yoo et al. [14] proposed a hybrid multi-objective algorithm based on GA and Simulated Annealing (SA) for scheduling of soft real-time tasks. In this method, the convergence of GA was improved by using the probability of SA as the criterion to accept the new solutions. They also proposed a new encoding and decoding method in GA [15] for regulation of the total number of active processors. The presented fitness function combines the adaptive weight approach (AWA) that utilizes some useful information from the current population to adjust the weights within the fitness function [15].

In this paper, we propose a new algorithm based on multi-objective artificial bee colony (ABC) for scheduling of soft real-time tasks in heterogeneous multiprocessor systems. It is assumed that all tasks are non-preemptive and have precedence relations among them. In this method, a hybrid local and global mechanism is introduced for the neighborhood search in ABC to improve the convergence of the proposed algorithm. The objective of proposed algorithm is to minimize the total tardiness, total number of active processors and the completion time simultaneously.

The rest of the paper is organized as follows: In Section 2, the problem definitions are formulated for scheduling of soft real-time tasks in heterogeneous multiprocessor systems. Section 3 the new extended ABC algorithm is introduced, and the implementations of it for scheduling problem is described. Experimental results of the LGABC and comparison with other scheduling algorithms are illustrated in Section 4. Finally, Section 5 provides the conclusion and suggestions for future works on this problem.

II. SCHEDULING PROBLEM DEFINITIONS

In this paper, the problem of scheduling soft real-time non-preemptive tasks in heterogeneous multiprocessor systems is considered. It is assumed that all tasks have timing constrained. The precedence relations among the tasks can be considered from the task relations graph. The problem is optimized in a way that simultaneously minimizes the total tardiness (f_1), the number of active processors (f_2), and the completion time (f_3), under the following conditions:

1. System is a heterogeneous multiprocessor system.
2. All tasks are non-preemptive.
3. Tasks have soft deadlines.
4. Tasks have precedence relations among them.
5. Every processor can process only one task at a time.
6. Every task is processed on one processor at a time.
7. Completion time of each task on each processor is known.
8. Deadline of each task is known.

The problem of scheduling non-preemptive soft real-time tasks in the heterogeneous multiprocessor system to minimize the objectives f_1, f_2 and f_3 is formulated as follows:

$$Min\{f = w_1 f_1 + w_2 f_2 + w_3 f_3\} \tag{1}$$

$$f_1 = \sum_{i=1}^{N} max\{0, \sum_{m=1}^{M}(t_i^S + c_i^m - d_i)x_i^m\} \tag{2}$$

$$f_2 = \sum_{m=1}^{M} min\{1, \sum_{i=1}^{N} x_i^m\} \tag{3}$$

$$f_3 = \max_i\{t_i^F\} \tag{4}$$

Subject to:

$$x_i^m = \begin{cases} 1 & if\ p_m\ is\ selected\ for\ \tau_i \\ 0 & otherwise \end{cases} \tag{5}$$

$$\sum_{m=1}^{M} x_i^m = 1 \tag{6}$$

$$t_i^S \geq t_i^E \tag{7}$$

$$t_i^F = t_i^S + \sum_{m=1}^{M} c_i^m x_i^m \tag{8}$$

$$t_i^E \geq t_j^E + \sum_{m=1}^{M} c_j^m x_j^m, \quad \tau_j \in pre(\tau_i) \tag{9}$$

$$t_i^E = \begin{cases} 0 & if\ pre(\tau_i) \in \{\emptyset\} \\ \max_j\{t_j^E + \sum_{m=1}^{M} c_j^m x_j^m\} & \tau_j \in pre(\tau_i) \end{cases} \tag{10}$$

$$e_{ij} = \begin{cases} 1 & if\ \tau_j \in pre(\tau_i) \\ 0 & otherwise \end{cases} \tag{11}$$

In above equations, the parameters and notations can be defined as follows:

N	: total number of tasks
M	: total number of processors
i,j	: task index, $i,j=1,2,...,N$
m	: processor index, $m=1,2,...,M$
$G=(T,E)$	: task graph
$T=\{\tau_i\}$	: a set of N tasks
$P=\{p_m\}$	: a set of M processors
$E=\{e_{ij}\}$	: a matrix of directed edges among tasks ($i \neq j$)
c_i^m	: computation time of task τ_i on processor p_m
d_i	: deadline of task τ_i
t_i^E	: earliest possible start time of task τ_i
t_i^S	: real start time of task τ_i
t_i^F	: finish time of task τ_i
$pre(\tau_i)$	: set of all predecessor tasks of task τ_i
$suc(\tau_i)$	: set of all successor tasks of task τ_i

Equation (1) is the proposed multi-objective function to be minimized. Equation (2) means to minimize the total tardiness of tasks (f_1). Equation (3) means to minimize the number of active processors (f_2). Equation (4) means to minimize the completion time (f_3). The constraint conditions are shown from (5) to (11). Equation (6) means that every task is processed on one processor at a time. Equation (7) means that the task can be started after its earliest possible start time. Eq. (10) defines the earliest possible start time of the task τ_i is calculated by the maximum finish time of all predecessor tasks of the task τ_i.

III. ARTIFICIAL BEE COLONY (ABC) ALGORITHM

Artificial Bee Colony (ABC) is a recently proposed swarm intelligence algorithm inspired by the natural behavior of honey bees in their search process for the best food source, in case of the nectar quality and the position. It was first proposed by Karaboga and Basturk in 2006 [16]. In ABC algorithm, the search space is simulated as the foraging environment with N-dimensional, where N is the total number of optimization variables within the problem. Each point in the search space corresponds to a food source position (a feasible solution), and

it's nectar amount corresponds to the fitness of the solution explored by the artificial bee.

In ABC algorithm, an artificial bee colony consists of three kinds of bees: employed bees, onlooker bees and scout bees [17]. Employed bees exploit the specific food sources they have explored in the previous iteration. Then, they come back to the hive and give the quality information of the food sources to the onlookers within the hive. Each onlooker receives the information about the food sources and chooses a good food source to exploit depending on the information of nectar quality. The more nectar the food source contains, the larger probability the onlooker will chooses it. The employed bee whose food source has been abandoned becomes a scout bee. It is controlled by a parameter called "limit". Scout bees explore the whole search space randomly. The process of selecting the employed bees by onlookers is similar to the natural selection in evolutionary algorithms, like the roulette wheel selection in GA. Also, the neighborhood search strategy in ABC can be compared with the mutation process in GA [18].

IV. Proposed Scheduling Algorithm

A. Representation of the feasible solutions

In heterogeneous multiprocessor platforms, the objective of scheduling is not only to optimize an execution order of tasks, but also to determine the specific processor to be used for each task. The first is an ordering problem, and the last is an assignment problem. Recently, authors have applied ABC to a popular ordering problem (Traveling Salesman Problem) [19] and an assignment problem (Multiple Knapsack Problem) [20]. The idea of employing ABC algorithm for the scheduling problem in this paper is inspired by our previous works using ABC for ordering and assignment problems.

In order to solve the scheduling problem using ABC, a mechanism must be employed to formulate the problem into ABC algorithm. In this paper, a hybrid structure is used to represent a feasible solution, which is partitioned into two parts. The first part determines the hybrid overall execution order of tasks (ordering part), and the second part denotes the processor number to which the task is assigned (assignment part). Therefore, the length of each part is the total number of tasks. The ordering part should satisfy the precedence relationship with respect to the given task graph. For example, a feasible solution for a dataset with 9 tasks and 3 processors can be shown in Fig. 1. According to Fig. 1, the corresponding scheduling on heterogeneous processors is as follows:

p_1: $\tau_2 \rightarrow \tau_4$

p_2: $\tau_7 \rightarrow \tau_1 \rightarrow \tau_9 \rightarrow \tau_5$

p_3: $\tau_8 \rightarrow \tau_6 \rightarrow \tau_3$

	1	2	3	4	5	6	7	8	9
Ordering Part :	**7**	**2**	**4**	**1**	**9**	**5**	**8**	**6**	**3**
Assignment Part :	**2**	**1**	**3**	**1**	**2**	**3**	**2**	**3**	**2**

Figure 1. An Example for representation of a feasible solution (a bee).

B. Construction of initial population

In ABC algorithm, half of the colony consists of employed bees and the other half are onlooker bees. At the initialization phase, the food sources are randomly explored by employed bees, within the range of the boundaries of the variables. In this way, the ordering part is randomly generated by respecting the precedence relationship of the tasks. On the other hand, any task τ_i must do not present before the tasks within it's $pre(\tau_i)$ set. In order to do that, at first, the tasks that have not any predecessor task are ordered randomly and placed at the beginning the ordering part. Then, all the predecessor tasks of them were selected and ordered randomly after the pre-ordered tasks. This process continues until all tasks are ordered by respecting the precedence relationship of the tasks. In the assignment part, a processor is selected randomly for each task to be processed on that processor.

C. Cost function

After all bees complete their search process, they come back to the hive and share their information about the quality of their food sources with the others. So, the quality of each solution is evaluated according to the corresponding cost achieved by (1). In this way, the fitness (nectar) of k^{th} bee is calculated as follows:

$$Fitness_k = \frac{1}{f_k} \tag{12}$$

where, $Nectar_k$ and f_k are respectively the nectar amount and the corresponding cost of the k^{th} bee.

D. Generation of the new population

In this part, each bee constructs a new solution, in order to update the current population. It has three phases:

1) Employed bees phase

In this phase, each of the employed bees is moved onto her previously visited food source environment, to explore a new food source in the neighborhood of the present one. If the nectar amount of the new solution is higher, the bee forgets the previous one and memorizes the new one.

2) Onlooker bees phase

After all employed bees construct their feasible solutions, they come back into the hive and share the quality information of the food sources with the onlookers within the hive [17]. The more nectar the food source contains, the larger probability the onlooker will choose it. The probability of an employed bee to be selected by onlookers is computed as follows:

$$P_i = \frac{(Fitness_i)^\alpha}{\sum_{j=1}^{PN}(Fitness_j)^\alpha} \tag{13}$$

where, P_i is the probability of the i^{th} employed bee to be chosen by onlookers. α is a constant parameter, that adjusts the selection type. The bigger value of α, the larger probability the onlooker will choose the employed bees with more fitness. After an onlooker selects an employed bee by (13), she then goes onto the food source area of the employed bee, to explore a new food source in the neighborhood of that.

3) Scout bees phase

As mentioned above, the employed bee whose food source has been abandoned will become a scout bee. It can be controlled by a parameter called *limit*. Then, the scout bee caries out random searching the whole search space to discover a new solution. It is performed like the random search strategy used for construction of the initial population.

E. The proposed hybrid neighborhood search strategy

In this paper, a hybrid local and global search strategy is proposed to improve the convergence of ABC algorithm. In this method, some neighborhood search operators are used for exploration phase of employed bees and onlooker bees. Each operator can improve a kind of trapped local points. In order to do that, Swap, Exchange, Relocation and Or-opt operators are applied [21]. The Relocation and Or-opt operators are used in the first part (ordering part) of the solution. The swap is applied in the second part (assignment part). And the Exchange is used in both parts, called "Exchange-1" and "Exchange-2" respectively. In both employed phase and onlooker phase, in order to explore the neighborhood are of a bee, the probability of applying each operator is 20%. The neighborhood search using these operators can be shown in Fig. 2. The overall flowchart of proposed algorithm can be seen in Fig. 3.

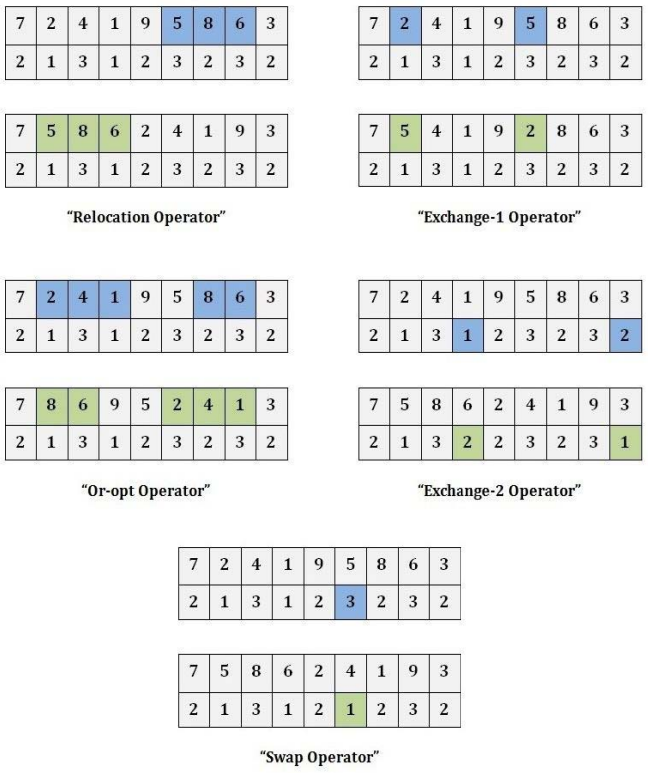

Figure 2. An Example of neighborhood search operators.

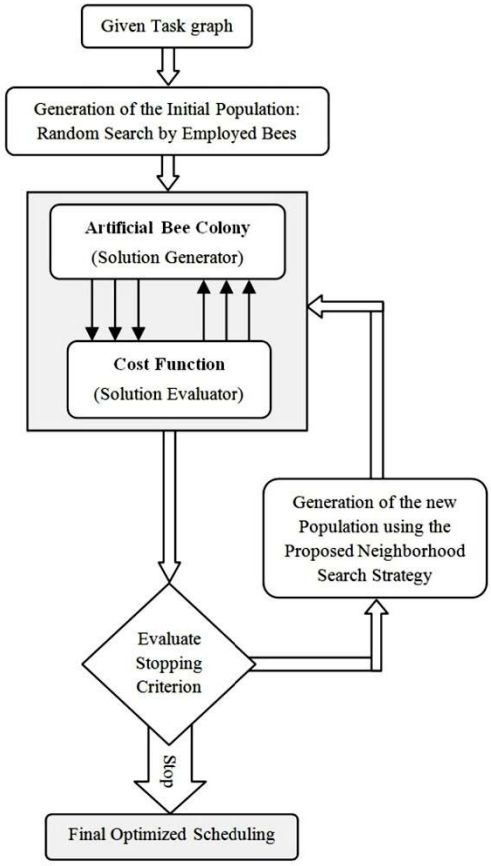

Figure 3. The overall flowchart of the proposed scheduling algorithm.

V. EXPERIMENTAL RESULTS

A. Tuning the parameters of ABC algorithm

All of the experiments were implemented on a PC with 2.53^{GHZ} processor and 4^{GB} memory running on windows 8. Determination of the parameters of ABC is an important issue on the efficiency of the proposed algorithm. For tuning the parameters of ABC algorithm, different values were tested, and the best values are considered. In our experiments, we set α=3 in (13). The population of bees was equal to 50, on the other hands the colony of artificial bees consists of 25 employed bees and 25 onlooker bees. The maximum number of iterations were set to $(10 \times N)$, where N is the number of tasks within the task graph. Finally, the parameters of the proposed algorithm can be summarized in Table 1.

TABLE I. TUNING THE PARAMETERS OF ABC ALGORITHM.

Parameter	***Value***
Maximum iteration	10×N
Population	50
Number of employed Bees	25
Number of onlooker Bees	25
α (Equation 13)	3
Limit parameter	0.6
Probability of applying each of the 5 neighborhood search operator	20% (uniform)
Maximum size of selected part in "Relocation"	5 tasks
Maximum size of each parts in "Or-opt"	3 tasks

B. Datasets for simulation

To validate the proposed ABC-based scheduling algorithm, several numerical tests [14] are performed. A task graph with 10 tasks (Fig. 4) and a task graph with 50 tasks (Fig. 5) were implemented. The details of dataset with 10 tasks can be summarized in Table 2. We compared the proposed algorithm with three GA-based algorithms named Monnier-GA by Monnier et al. [9], Oh-GA by Oh et al. [10] and Yoo-GA by Yoo et al. [14], respectively. All algorithms were simulated in the same situations in the same dataset.

TABLE II. DATASET DETAILS OF THE PROBLEM 1, WITH 10 TASKS [14].

I	$suc(\tau_i)$	d_i	C_i^m			
			C_i^1	C_i^2	C_i^3	C_i^4
1	8	19	5	3	11	8
2	6	9	6	5	4	13
3	4,5	18	11	8	6	7
4	6,7,8	37	10	13	5	6
5	6,10	38	10	13	8	11
6	9	37	2	11	11	3
7	---	44	3	10	11	8
8	---	30	4	12	10	5
9	10	37	6	9	7	10
10	---	58	11	12	6	4

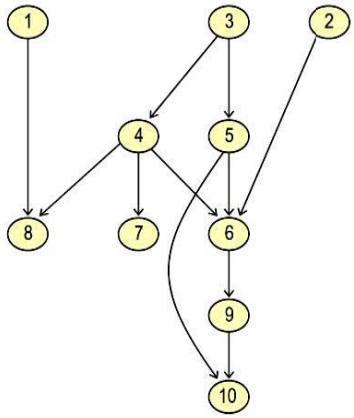

Figure 4. The task graph with 10 tasks (Problem 1) [14].

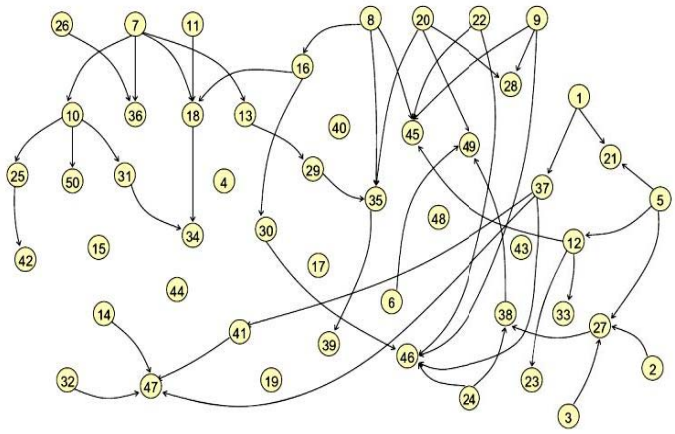

Figure 5. The task graph with 50 tasks (Problem 2) [14].

C. Simulation results

The comparisons of the obtained results with the two other scheduling algorithms can be shown in Table 3 and Table 4. Table 3 represents the comparison of the tardiness with the same number of active processors, and Table 4 represents the comparison of several terms without tardiness. In Table 3, the tardiness of the proposed algorithm is smaller than those of the other two algorithms. In Table 4, the total computation time of the proposed algorithm is a little bit longer than those of the other two algorithms. However, the number of utilized processors is fewer (2 processors) than those of the other two algorithms (3 processors). The variance of processor utilization rate by the proposed algorithm is more desirable than those of the others.

TABLE III. COMPARISON OF THE TOTAL TARDINESS ON PROBLEM 1 (FIGURE 4).

Algorithm	*Total number of active processors*		
	1	*2*	*3*
Monnier-GA [9]	35	19	0
Oh-GA [10]	36	13	0
Yoo-GA [14]	25	0	0
Proposed Algorithm	25	0	0

TABLE IV. COMPARISON OF THE RESULTS WITH NO TARDINESS ON PROBLEM 1 (FIGURE 4).

		Monnier-GA [9]	Oh-GA [10]	Yoo-GA [14]	Proposed Algorithm
Number of Active Processors		3	3	2	2
Computation Time		44	42	45	45
Utilization of Processors	P_1	0.59	0.57	0.51	100
	P_2	0.68	0.59	100	0.51
	P_3	0.27	0.47	---	---
	Average	0.51	0.54	0.75	0.75

TABLE V. COMPARISON OF THE TOTAL TARDINESS ON PROBLEM 2 (FIGURE 5).

Algorithm	*Total number of active processors*					
	5	*10*	*13*	*15*	*16*	*17*
Monnier-GA [9]	135	78	---	25	13	0
Oh-GA [10]	121	59	---	19	0	0
Yoo-GA [14]	60	22	---	0	0	0
Proposed Algorithm	60	17	0	0	0	0

TABLE VI. COMPARISON OF THE RESULTS WITH NO TARDINESS ON PROBLEM 2 (FIGURE 5).

	Monnier-GA [9]	Oh-GA [10]	Yoo-GA [14]	Proposed Algorithm
Number of Active Processors	17	16	15	13
Computation Time	43	46	47	49
Average Utilization of Processors	0.45	0.47	0.49	0.54

VI. Conclusion and Future Works

In this paper, a new algorithm based on multi-objective artificial bee colony (ABC) was proposed for scheduling of soft real-time tasks in heterogeneous multiprocessor systems. This algorithm was designed for non-preemptive tasks with the precedence relationship between them. The objective of the proposed algorithm is to minimize the total tardiness, the total number of active processors and the completion time simultaneously. In this method, the convergence of ABC is improved by introducing a new hybrid neighborhood search strategy. From the simulation results, the results of the proposed ABC-based algorithm are better than that of the other algorithms. The number of utilized processors in the proposed algorithm is fewer than those of the other two algorithms. Also, the variance of processor utilization rate by the proposed algorithm is more desirable than those of the others.

We plan to design a hybrid algorithm for scheduling of real-time tasks on heterogeneous multiprocessor systems using swarm intelligence algorithms and local search algorithms like simulated annealing (SA) and variable neighborhood search (VNS). In this way, we plan to introduce other neighborhood search strategies in our algorithm.

References

[1] H. Chen, A. M. K. Cheng, Y. W. Kuo, "Assigning real-time tasks to heterogeneous processors by applying ant colony optimization," J. Parallel Distrib. Comput., vol. 71, pp. 132–142, 2011.

[2] C. M. Krishna, G. S. Kang, Real-time system. NewYork: McGraw-Hill, 1997.

[3] G. Bernat, A. Burns, A. Liamosi, Weakly hard real-time systems. IEEE Transactions on Computer Systems, vol. 50, no. 4, pp. 308–21, 2001.

[4] M. H. Kim, H. G. Lee, J. W. Lee, A proportional-share scheduler for multimedia applications. In: Proceedings of the multimedia computing and systems, pp.484-491, 1997.

[5] M. R. Yoo, A scheduling algorithm for multimedia process. PhD dissertation, University of YeoungNam, Korea, 2002.

[6] F. Yalaoui, C. Chu, "Parallel machine scheduling to minimize total tardiness," International Journal of Production Economics, vol. 76, no. 3, pp. 265-279, 2002.

[7] H. Mitra, P. Ramanathan, "A genetic approach for scheduling non-preemptive tasks with precedence and deadline constraints," In: Proceedings of the 26th Hawaii international conference on system sciences, pp. 556–564, 1993.

[8] M. Lin, L. Yang, "Hybrid genetic algorithms for scheduling partially ordered tasks in a multi-processor environment," In: Proceedings of the sixth international conference on real-time computer systems and applications, pp. 382–387, 1999.

[9] Y. Monnier, J. P. Beauvais, A. M. A. Deplanche, "Genetic algorithm for scheduling tasks in a real-time distributed system," In: Proceedings of the 24th euromicro conference, pp. 708–714, 1998.

[10] J. Oh, C. Wu, "Genetic-algorithm-based real-time task scheduling with multiple goals," Journal of Systems and Software, vol. 71, no. 3, pp. 245-258, 2004.

[11] M. D. Theys, T. D. Braun, H. J. Siegal, A. A. Maciejewski, Y. K. Kwok, "Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach," In: Zomaya AY, Ercal F, Olariu S, editors. Solutions to parallel and distributed computing problems. New York: Wiley, pp. 135–178 [chapter 6], 2001.

[12] A. J. Page, T. J. Naughton, "Dynamic task scheduling using genetic algorithm for heterogeneous distributed computing," In: Proceedings of the 19th IEEE international parallel and distributed processing symposium, pp. 189-201, 2005.

[13] M. K. Dhodhi, I. Ahmad, A. Yatama, I. Ahmad, "An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems," Journal of Parallel and Distributed Computing, vol. 62, pp. 1338–1361, 2002.

[14] M. Yoo, M. Gen, "Scheduling algorithm for real-time tasks using multiobjective hybrid genetic algorithm in heterogeneous multiprocessors system," Journal of Computers & Operations Research, vol. 34, pp. 3084–3098, 2007.

[15] M. Yoo, "Real-time task scheduling by multiobjective genetic algorithm," Journal of Systems and Software, vol. 82, pp. 619–628, 2009.

[16] D. Karaboga, B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Journal of global optimization, vol. 39, no. 3, pp. 459-471, 2007.

[17] D. Karaboga, B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," Applied Soft Computing, pp. 687–697, 2008.

[18] M. Shokouhifar, F. Farokhi, "An artificial bee colony optimization for feature subset selection using supervised fuzzy c_means algorithm," 3rd International Conference on Information Security and Artificial Intelligent (ISAI), pp. 427-432, December 2010.

[19] Sh. Sabet, M. Shokouhifar, F. Farokhi, "A hybrid mutation-based artificial bee colony for traveling salesman problem," 4th International Conference on Electronics Computer Technology (ICECT 2012), pp. 348–352, 2012.

[20] Sh. Sabet, M. Shokouhifar, F. Farokhi, "A discrete artificial bee colony for multiple knapsack problem," *Int. J. Reasoning-based Intelligent Systems*, vol. 5, no. 2, pp.88–95, 2013.

[21] T. Caric, H. Gold, Vehicle Routing Problem. In-The, Croatian branch of I-Tech Education and Publishing KG, Vienna, Austria, pp. 19-25, 2008.