Journal of Communication and Computer 13 (2016) 103-115
doi:10.17265/1548-7709/2016.03.001

Real-time Task Scheduling in Heterogeneous Multiprocessors System Using Hybrid Genetic Algorithm

Myungryun Yoo

Department of Computer Science Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, Tokyo, Japan

Abstract: The real-time multiprocessor scheduling problem is one of the NP-hard problems. Furthermore, there are no papers which are concerned to heterogeneous multiprocessors system. This paper proposes a new real-time task scheduling algorithm using hGA (hybrid genetic algorithm) on heterogeneous multiprocessor environment. In solution algorithms, the GA (genetic algorithm) and the SA (simulated annealing) are cooperatively used. In this method, the convergence of GA is improved by introducing the probability of SA as the criterion for acceptance of new trial solution. The objective of proposed scheduling algorithm is to minimize total tardiness. The effectiveness of the proposed algorithm is shown through simulation studies. In simulation studies, the results of proposed algorithm show better than that of other algorithms.

Key words: Genetic algorithm, real-time task scheduling, heterogeneous multiprocessor.

1. Introduction

In hard real-time system, tardiness can be catastrophic. The goal of hard real-time scheduling algorithms is to meet all tasks' deadlines, in other words, to keep the feasibility of scheduling through admission control. However, in the case of soft real-time systems, slight violence of deadlines is not so critical [1].

Traditionally, the performance criteria of algorithm for TSP (task scheduling problem) are throughput, utilization of processors, waiting time of tasks, etc. In hard real-time system, the performance of scheduling algorithm is measured by its ability to generate a feasible schedule for a set of real-time tasks. Typically, there is RM (rate monotonic) and EDF (earliest deadline first) derived scheduling algorithms for hard real-time system with uniprocessor [2, 3]. They guarantee the optimality in somewhat restricted environments. However, these algorithms have some drawbacks to cope with soft real-time system related resource utilization and pattern of degradation under the overloaded situation. The objective of scheduling task in soft real-time system is to minimize total tardiness. As the growing of soft real time applications, the necessity of scheduling algorithm for soft real-time system is on the increase and several researches for soft real time system are reported. rrPS (rate regulating proportional share) scheduling algorithm based on stride scheduler by Kim [4] and mPS (modified proportional share) scheduling algorithm by Yoo [5] are designed for tasks in soft real-time system. However, these algorithms also can not show the graceful degradation of performance under the overloaded situation and are restricted in uniprossor system. The optimal assignment of tasks to multiprocessor is, in almost all practical cases, an NP-hard problem [6]. Consequently various modern heuristics based algorithms have been proposed for practical reason.

Recently, several approaches GA are proposed. Mitra and Ramanathan proposed a GA for scheduling of nonpreemptive tasks with precedence and deadline

Corresponding author: Myungryun Yoo, Ph.D., research field: real time OS.

constraints [7]. Lin and Yang presented a hybrid GA, where different operators are applied at different stage of the lifetime, for scheduling partially ordered nonpreemptive tasks in a multiprocessor environment [8]. Monnier et al. presented a GA implementation to solve a real-time nonpreemptive task scheduling problem [9]. Oh and Wu presented a multiobjective GA for scheduling nonpreemptive tasks in soft real-time system with multiprocessors [10]. However, these algorithms assume that the performance of all processors is same.

In this paper, we propose a new scheduling algorithm for nonpreemptive tasks with precedence relationship in soft real-time heterogeneous multiprocessor system. In solution algorithms, the GA and the SA (simulated annealing) are cooperatively used [11]. In this method, the convergence of GA is improved by introducing the probability of SA [12] as the criterion for acceptance of new trial solution. However, it is hard to find the optimum solution by only applying the genetic operators. The objective of proposed scheduling algorithm is to minimize the total tardiness.

The rest of the paper is organized as follows: In Section 2, we explain sr-TSP (soft real-time task scheduling problem) in heterogeneous multiprocessors system and the problem is mathematically formulated. Section 3 introduces the GA combined with SA methods and describes implementations used for this problem. Then, the experimental results are illustrated and analyzed in Section 4. Finally, Section 5 provides discussion and suggestions for further work on this problem.

2. Soft Real-time Task Scheduling Problem and Mathematical Model

In this study, we consider the problem of scheduling the tasks with precedence and timing constrained task graph on a set of heterogeneous processors in a way that minimizes the total tardiness $F(\boldsymbol{x}, \boldsymbol{t}^S)$ under the following conditions:

(1)All tasks are nonpreemptive;

(2) Every processor processes only one task at a time;

(3) Every task is processed on one processor at a time;

(4) Only processing requirements are significant; memory, I/O, and other resource requirements are negligible.

The sr-TSP is formulated under the following assumptions: Computation time and deadline of each task are known. A time unit is artificial time unit. Soft real-time tasks scheduling problem in heterogeneous multiprocessors system to minimize the total tardiness is formulated as follows:

$$\min F(\boldsymbol{x}, \boldsymbol{t}^S) =$$

$$\sum_{i=1}^{N} \max\left\{0, \sum_{m=1}^{M} (t_i^S + c_{im} - d_i) \cdot x_{im}\right\} \quad (1)$$

$$\text{s.t.} \quad t_i^E \le t_i^S \le d_i, \quad \forall i \quad (2)$$

$$t_i^E \ge t_j^E + \sum_{m=1}^{M} c_{jm} \cdot x_{jm}, \quad \tau_j \in \text{pre}(\tau_i), \quad \forall i \quad (3)$$

$$\sum_{m=1}^{M} x_{im} = 1, \ \forall i \quad (4)$$

$$x_{im} \in \{0,1\}, \ \forall i, m \quad (5)$$

In above equations, notations are defined as follows:

-Indices

i, j : task index, $i, j = 1,2,\ldots,N$

m: processor index, $m = 1, 2,\ldots, M$

-Parameters

$G = (T, E)$: task graph

$T = \{\tau_1, \tau_2, \ldots, \tau_N\}$: a set of N tasks

$E = \{e_{ij}\}$, $i, j=1, 2,\ldots,N$, $I \neq j$: a set of directed edges among the tasks representing precedence relationship

τ_i : the I th task, $i = 1, 2,\ldots, N$

e_{ij} : precedence relationship between task τ_i and task τ_j

p_m: the m th processor, $m = 1, 2,\ldots,M$

c_{im}: computation time of task τ_i on processor p_m

d_i : deadline of task τ_i

pre*(τ_i): set of all predecessors of task τ_i

suc*(τ_i): set of all successors of task τ_i

pre(τ_i): set of immediate predecessors of task τ_i

suc(τ_i): set of immediate successors of task τ_i

t_i^E: earliest start time of task τ_i

$$t_i^E = \begin{cases} 0, & \text{if } \neg\exists \tau_j : (\tau_j, \tau_i) \in E \\ \max\limits_{\tau_j \in \text{pre*}(\tau_i)} \{t_j^E + \sum\limits_{m=1}^{M} c_{jm} \cdot x_{jm}\}, & \text{otherwise} \end{cases}, \quad \forall i \tag{6}$$

t_i^F: finish time of task τ_i

$$t_i^F = \min\{t_i^S + \sum_{m=1}^{M} c_{im} \cdot x_{im},\ d_i\},\ \forall i \tag{7}$$

-Decision Variables

t_i^S: real start time of task τ_i

$$x_{im} = \begin{cases} 1, & \text{if processor } p_m \text{ is selected for task } \tau_i \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Eq. (1) is the objective function in this scheduling problem. Eq. (1) means to minimize total tardiness of tasks. Constraints conditions are shown from Eq. (2) to Eq. (5). Eq. (2) means that task can be started after its earliest start time, begin its deadline. Eq. (3) defines the earliest start time of task based on precedence constraints. Eq. (4) means that every task is processed on one processor at a time. Fig. 1 represents the time chart of sr-TSP.

3. GA Approach Combined with SA

In this paper, solution algorithm is based on GA. Several new techniques are proposed in the encoding and decoding algorithm of genetic string and the genetic operations are introduced for discussion. They are explained in the following subsections.

3.1 Encoding and Decoding

A chromosome V_k, $k = 1,2,\ldots,$ *popSize*, represents one of all the possible mappings of all the tasks into the processors. Where *popSize* is the total number of chromosomes in a generation. A chromosome V_k is partitioned into two parts $u(\cdot)$, $v(\cdot)$. $u(\cdot)$ means scheduling order and $v(\cdot)$ means allocation information. The length of each part is the total number of tasks. The scheduling order part should be a topological order with respect to the given task graph that satisfies precedence relationship. The allocation information part denotes the processor to which task is allocated.

Encoding procedure for soft real-time task scheduling problem (sr-TSP) will be written as follows:

procedure: Encoding for sr-TSP

input: task graph data set, total number of processors M

output: $u(\cdot)$, $v(\cdot)$

begin

$l \leftarrow 1$, $W \leftarrow \phi$;

while ($T \neq \phi$)

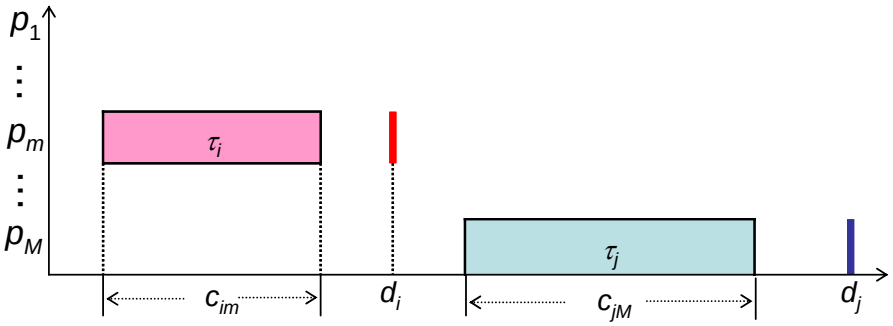


Fig. 1 The time chart of sr-TSP.

$W \leftarrow W \cup \arg\{\tau_i \mid \text{pre}^*(\tau_i) = \phi, i\}$;
$T \leftarrow T - \{\tau_i\}, i \in W$;
while ($W \neq \phi$)
$j \leftarrow$ random(W);
$u(l) \leftarrow j$;
$W \leftarrow W - \{j\}$;
$\text{pre}^*(\tau_i) \leftarrow \text{pre}^*(\tau_i) - \{\tau_j\}, i$;
$m \leftarrow$ random[1:M];
$v(l) \leftarrow m$;
$l \leftarrow l+1$;
end
end
output $u(\cdot)$, $v(\cdot)$;
end

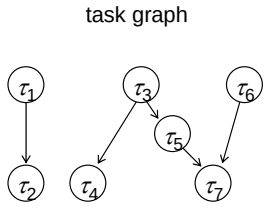
Where, W is temporary defined working data set for tasks without predecessors. In encoding procedure, feasible solutions are generated by respecting the precedence relationship of task and allocated processor is selected randomly.

Fig. 2 represents the example of this encoding procedure.

Decoding procedure will be written as follows:

procedure: Decoding for sr-TSP
input: task graph data set, chromosome $u(\cdot)$, $v(\cdot)$
output: schedule set S, total tardiness of tasks F
begin
$l \leftarrow 1$, $F \leftarrow 0$, $S \leftarrow \phi$;
while ($l \leq N$)
$i \leftarrow u(l)$;
$m \leftarrow v(l)$;
if (exist suitable idle time) **then** insert(i);
start(i);
update_idle();
$F \leftarrow F + \max\{0, (t_i^S + c_{im} - d_i)\}$;
$S \leftarrow S \cup \{(\tau_i, p_m: t_i^S - t_i^F)\}$;
$l \leftarrow l+1$;
end
output S, F
end

Where insert (i) means to insert τ_i at idle time if τ_i is computable in idle time. At start (i), the real start time of ith task t_i^S and the finish time of i th task t_i^F can be calculated. updata_idle () means that the list of idle time is updated if new idle time duration is occurred. The objective value $F(\boldsymbol{x}, \boldsymbol{t}^S)$ and schedule set S is generated through this procedure.



total number of processors M=3

chromosome V_k

l	1	2	3	4	5	6	7
$u(\cdot)$	3	6	1	4	5	2	7
$v(\cdot)$	1	2	3	1	3	3	1

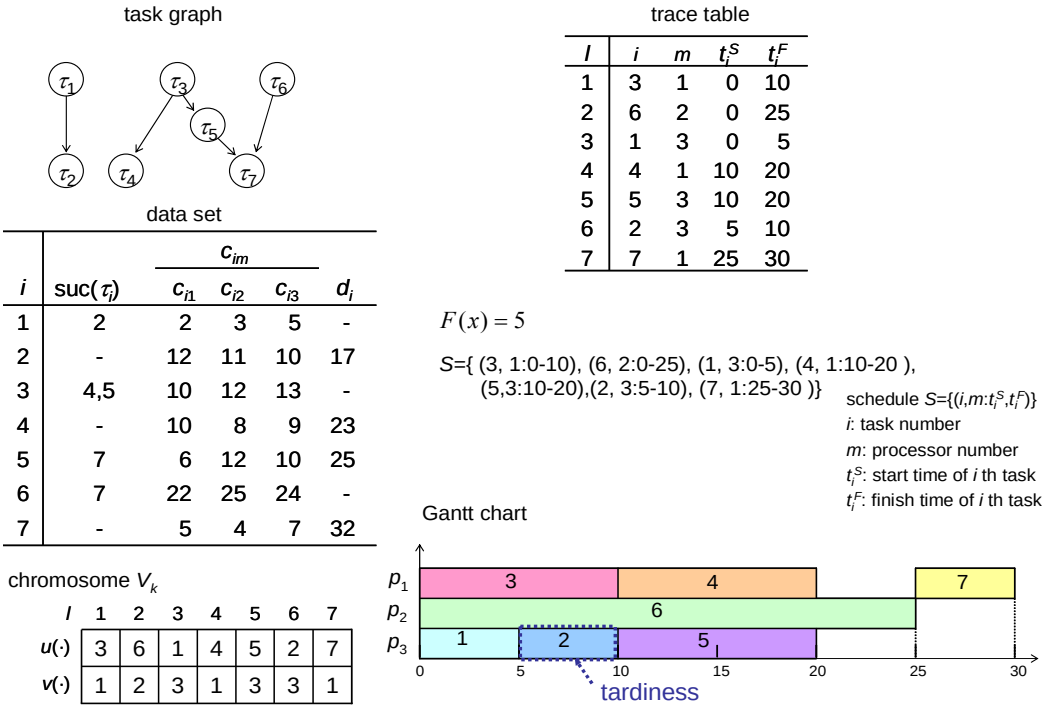
data set

i	suc(τ_i)	c_{im}			d_i
		c_{i1}	c_{i2}	c_{i3}	
1	2	2	3	5	-
2	-	12	11	10	17
3	4,5	10	12	13	-
4	-	10	8	9	23
5	7	6	12	10	25
6	7	22	25	24	-
7	-	5	4	7	32

trace table

l	m	j	W	T
			ϕ	$\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7\}$
1	1	3	{1,3,6}	$\{\tau_2, \tau_4, \tau_5, \tau_7\}$
2	2	6	{1,6}	$\{\tau_2, \tau_4, \tau_5, \tau_7\}$
3	3	1	{1}	$\{\tau_2, \tau_4, \tau_5, \tau_7\}$
			ϕ	$\{\tau_2, \tau_4, \tau_5, \tau_7\}$
4	1	4	{2,4,5}	$\{\tau_7\}$
5	3	5	{2,5}	$\{\tau_7\}$
6	3	2	{2}	$\{\tau_7\}$
			ϕ	$\{\tau_7\}$
7	1	7	{7}	ϕ
			ϕ	ϕ

Fig. 2 The example of encoding procedure.



i	suc(τ_i)	c_{i1}	c_{i2}	c_{i3}	d_i
1	2	2	3	5	-
2	-	12	11	10	17
3	4,5	10	12	13	-
4	-	10	8	9	23
5	7	6	12	10	25
6	7	22	25	24	-
7	-	5	4	7	32

l	i	m	t_i^S	t_i^F
1	3	1	0	10
2	6	2	0	25
3	1	3	0	5
4	4	1	10	20
5	5	3	10	20
6	2	3	5	10
7	7	1	25	30

l	1	2	3	4	5	6	7
u(·)	3	6	1	4	5	2	7
v(·)	1	2	3	1	3	3	1

Fig. 3 The example of decoding procedure.

Fig. 3 represents the example of decoding procedure with chromosome in Fig. 2.

3.2 Evolution Function and Selection

The fitness function is essentially the objective function for the problem. It provides a means of evaluating the search node and it also controls the selection process [13, 14].

The fitness function used for our algorithm is based on the $F(\boldsymbol{x}, \boldsymbol{t}^S)$ of the schedule. Because we use the roulette wheel selection, we convert the minimization problem to maximization problem, that is, the used evaluation function is then

$$eval(V_k) = 1 / F(\boldsymbol{x}, \boldsymbol{t}^S), \quad \forall k \tag{9}$$

Selection is the main way GA mimics evolution in natural systems: fitter an individual is, the highest is its probability to be selected. For selection, the commonly strategies called roulette wheel selection [9, 15] has been used.

3.3 GA Operators

We use one-cut crossover. This operator creates two new chromosomes (proto-offspring) by mating two chromosomes (the parent). The one-cut crossover procedure will be written as follows:

procedure: One-cut Crossover

input: parent chromosomes $u_1(\cdot)$, $v_1(\cdot)$, $u_2(\cdot)$, $v_2(\cdot)$

output: proto-offspring chromosomes $u_1'(\cdot)$, $v_1'(\cdot)$, $u_2'(\cdot)$, $v_2'(\cdot)$

begin

$r \leftarrow$ random[1,N];

$u_1'(\cdot) \leftarrow u_1(\cdot)$;

$v_1'(\cdot) \leftarrow v_1[1{:}r]$ // $v_2[r{+}1{:}N]$;

$u_2'(\cdot) \leftarrow u_2(\cdot)$;

$v_2'(\cdot) \leftarrow v_2[1{:}r]$ // $v_1[r{+}1{:}N]$;

output proto-offspring chromosomes $u_1'(\cdot)$, $v_1'(\cdot)$, $u_2'(\cdot)$, $v_2'(\cdot)$;

end

where $u'(\cdot)$, $v'(\cdot)$ are proto-offspring chromosome and $(\cdot)_1//(\cdot)_2$ means to append $(\cdot)_2$ after $(\cdot)_1$. Fig. 4 represents the example of one-cut crossover procedure.

For another GA operator, mutation, we use the classical one-bit altering mutation [16].

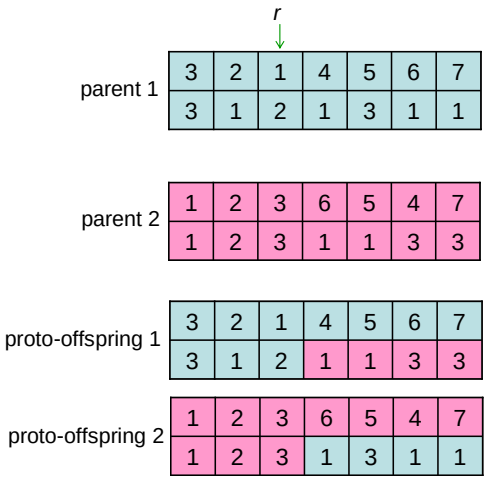


Fig. 4 The example of one-cut crossover.

3.4 Improving of Convergence by the Probability of SA

The convergence speed to local optimum of the GA can be improved by adopting the probability of SA (simulated annealing) [11, 12]. Even though the fitness function value of newly produced strings is lower than those of current strings, the newly produced ones are fully accepted in early stages of searching process. However, in later stages, a string with lower fitness function value is seldom accepted. The procedure of improved GA by the probability of SA will be written as follows:

procedure: Improving of GA chromosome by the probability of SA

input: parent chromosome V, proto-offspring chromosomes V',

temperature T, cooling rate of SA ρ

output: offspring chromosomes V''

begin

$r \leftarrow$ random[0,1];

$\Delta E \leftarrow eval(V')$-eval($V$);

if ($\Delta E > 0$ || $r <$Exp($\Delta E/T$))

$V'' \leftarrow V'$;

else

$V'' \leftarrow V$;

$T \leftarrow T$ xρ;

output offspring chromosomes V''

end

In this procedure, V and V' mean parent chromosome and proto-offspring chromosome. V'' means offspring chromosome which is produced by this procedure. The T means the temperature and the ρ means the cooling rate of SA.

3.5 Reproduction and Population Replacement

During reproduction and replacement steps, proto-offspring chromosomes are created by mating, with probability p_C, pairs of parents selected in the current population. And then chromosomes are mutated with probability p_M, randomly using one of the mutation operators [13]. The offspring chromosomes are produced by the probability of SA. Then new population is built through evaluating chromosomes and selecting.

This iterative evolution process is stopped as soon as one solution is found. However, we limit the number of offspring produced to *maxGen*, in order to avoid prohibitive calculation time, and to ensure that the GA will stop when treating an infeasible problem. Consequently, our proposed hGA (hybrid genetic algorithm) obeys to the following algorithm:

procedure: sr-TSP by hGA+SA

input: task graph data set

output: best schedule set S

begin

$t \leftarrow 0$;

initialize $P(t)$ by encoding routine;

fitness $eval(P)$ by decoding routine;

while (**not** termination condition) **do**

one-cut crossover $P(t)$ to yield $C'(t)$;

altering mutation $P(t)$ to yield $C'(t)$;

improving GA chromosome by the probability of SA;

fitness $eval(P, C)$ by by decoding routine;

select $P(t+1)$ from $P(t)$ and $C(t)$;

$t \leftarrow t+1$;

end

output best schedule set S;

end

4. Validation

To validate proposed hGA, several numerical tests are performed. We compared proposed hGA with Monnier's algorithm and proposed GA which is not combined with SA. The Monnier's algorithm is concerned to homogeneous multiprocessors system and the proposed hGA is designed for heterogeneous multiprocessors system. As there are no algorithms which are concerned to heterogeneous multiprocessors system, we compared proposed hGA with Monnier's algorithm in heterogeneous multiprocessors system. The Monnier's algorithm is proposed by Monnier, Beauvais and Deplanche [9]. This algorithm based on simple GA uses linear fitness normalization technique for evaluating chromosomes. The linear fitness normalization technique is effective to increase competition between similar chromosomes. However this method is limited in special problem with similar chromosomes. And in this algorithm, insertion method is not used. In other words, although there is idle time, task can not be executed in idle time. Numerical tests are performed with randomly generated task graph. We use P-Method [17] for generation task graph. The P-Method of generating a random task graph is based on the probabilistic construction of an adjacency matrix of a task graph. Element a_{ij} of the matrix is defined as 1 if there is a precedence relationship from τ_i to τ_j; otherwise, a_{ij} is zero. An adjacency matrix is constructed with all its lower triangular and diagonal elements set to zero. Each of the remaining upper triangular elements of the matrix is examined individually as part of a Bernoulli process with parameter ε, which represents the probability of a success. For each element, when the Bernoulli trial is a success, then the element is assigned a value of one; for a failure the element is given a value of zero. The parameter ε can be considered to be the sparsity of the task graph. With this method, a probability parameter of $\varepsilon = 1$ creates a totally sequential task graph, and $\varepsilon = 0$ creates an inherently parallel one. Values of ε that lie in between these two extremes generally produce task graphs that possess intermediate structures.

For tasks' computation time and deadline, we use random number based on exponential distribution and normal distribution as follows:

c_{im}^{E} = random value based on exponential distribution with mean 5

c_{im}^{N} = random value based on normal distribution with mean 5

r^{E} = random value based on exponential distribution with mean c_i^{E}

r^{N} = random value based on normal distribution with mean c_i^{N}

$$d_i^{E} = t_i^{E} + \max\{\, c_{im}^{E},\ \forall m\} + r^{E}$$

$$d_i^{N} = t_i^{E} + \max\{\, c_{im}^{N},\ \forall m\} + r^{N}$$

where, c_{im}^{E} and c_{im}^{N} are the computation time of ith task on mth processor based on exponential distribution and normal distribution respectively. d_i^{E} and d_i^{N} are the deadline of ith task based on exponential distribution and normal distribution respectively.

Numerical tests are performed with three task graph: the number of tasks 10, 50 and 100.

4.1 Example 1

Fig. 5 represents the task graph which used in example 1.

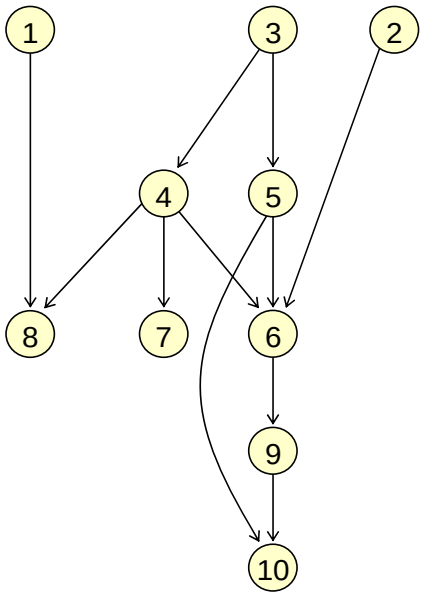


Fig. 5 Task graph with 10 tasks.

For generality of numerical test, Tables 1 and 2 are data set generated by exponential distribution and normal distribution of task graph in Fig. 5 respectively.

Tables 3 and 4 show the comparisons of results by three different scheduling algorithms based on data in Table 1. Tables 5 and 6 show the comparisons of results by three different scheduling algorithms based on data in Table 2. In Tables 3 and 5, $F(\boldsymbol{x}, \boldsymbol{t}^S)$ of each algorithm is compared in different number of processors. The results of proposed hGA are better than that of other algorithms. However, total number of processors without tardiness is same at all algorithms. In Tables 4 and 6, some terms such as makespan, computing time and the utilization of processors are compared on the total number of processors without tardiness. The computing time of proposed hGA is a little bit longer than those of the other two algorithms. However, this computation time can be an acceptable time for long term scheduling.

Table 1 Data set of task graph with 10 tasks (exponential distribution).

i	suc(τ_i)	c_{im}			d_i	i	suc(τ_i)	c_{im}			d_i
		c_{i1}	c_{i2}	c_{i3}				c_{i1}	c_{i2}	c_{i3}	
1	8	5	3	10	13	6	9	2	4	7	24
2	6	3	7	12	17	7	-	2	15	4	13
3	4,5	3	4	1	12	8	-	3	5	4	18
4	6,7,8	2	16	6	12	9	10	5	5	8	27
5	6,10	12	2	4	27	10	-	1	5	6	29

Table 2 Data set of task graph with 10 tasks (normal distribution).

i	suc(τ_i)	c_{im}				d_i	i	suc(τ_i)	c_{im}				d_i
		c_{i1}	c_{i2}	c_{i3}	c_{i4}				c_{i1}	c_{i2}	c_{i3}	c_{i4}	
1	8	5	3	11	8	19	6	9	2	11	11	3	37
2	6	6	5	4	13	9	7	-	3	10	11	8	44
3	4,5	11	8	6	7	18	8	-	4	12	10	5	30
4	6,7,8	10	13	5	6	37	9	10	6	9	7	10	37
5	6,10	10	13	8	11	38	10	-	11	12	6	4	58

Table 3 Comparison with 3 algorithms for $F(x, t^S)$ based on data set in Table 1 (exponential).

Total number of processors	Monnier's GA	Proposed GA	Proposed hGA
1	17	16	14
2	9	10	6
3	0	0	0

Table 4 Comparison with 3 algorithms without tardiness based on data set in Table 1 (exponential).

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors M	3	3	3
make span	29	27	26
Computing time (*msec*)	22	23	32
Average utilization of processors	0.517179	0.550725	0.568571

Table 5 Comparison with 3 algorithms for $F(x, t^S)$ based on data set in Table 2 (normal).

Total number of processors	Monnier's GA	Proposed GA	Proposed hGA
1	59	60	54
2	28	28	13
3	7	5	3
4	0	0	0

Table 6 Comparison with 3 algorithms without tardiness based on data set in Table 2 (normal).

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors M	4	4	4
Makespan	36	38	33
Computing times (*msec*)	32	32	47
Average utilization of processors	0.515152	0.519683	0.525556

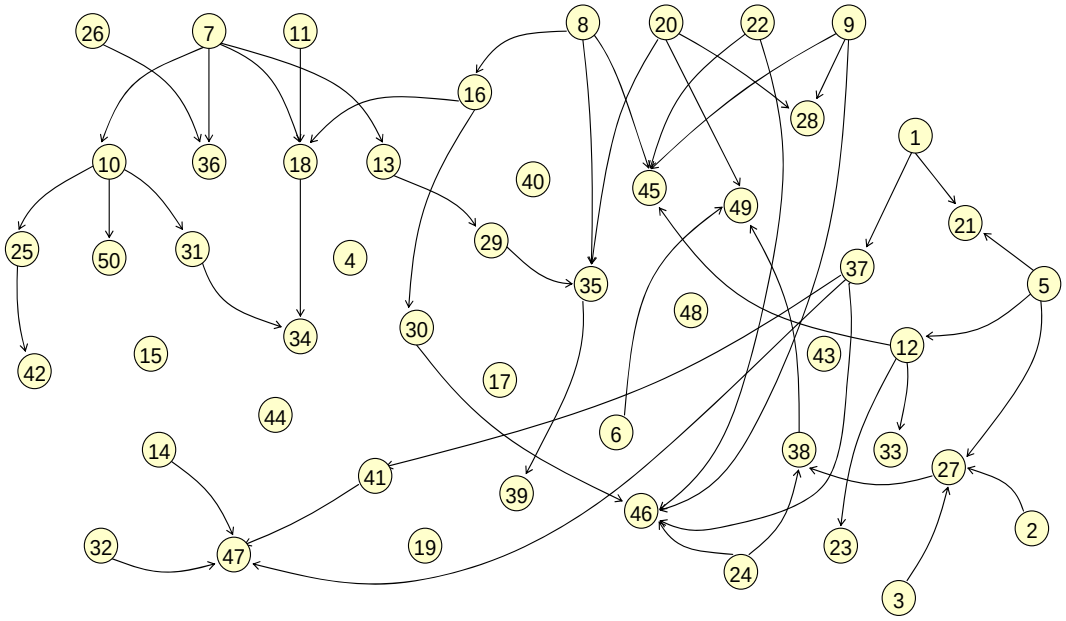


Fig. 6 Task graph with 50 tasks.

The average utilization of processors and makespan of hGA is more desirable than those of the others.

However, in these cases, total number of processors without tardiness is same at all algorithms. The distinction between proposed hGA and other algorithms is not appeared because the total number of tasks is very small.

4.2 Example 2

In this example, we use a task graph with 50 tasks. Fig. 6 represents the task graph of 50 tasks case.

The data set of tasks representing time constraints is omitted. Figs. 7 and 8 show that the comparison with 3 algorithms for $F(\boldsymbol{x}, \boldsymbol{t}^S)$ based on data in exponential distribution and normal distribution respectively. In Fig. 7 and 8, $F(\boldsymbol{x}, \boldsymbol{t}^S)$ of proposed hGA is smaller than that of each algorithms.

In Tables 7 and 8, some terms such as makespan, computing time and the utilization of processors are compared on the total number of processors without tardiness. Total number of processors without tardiness of proposed hGA is smaller than that of other algorithms. The computing time of proposed hGA is a little bit longer than those of the other two algorithms. However, this computation time can be an acceptable time for long term scheduling. The average utilization of processors of hGA is more desirable than that of other algorithms.

4.3 Example 3

In this example, we use a task graph with 100 tasks. Task graph and the data set of tasks representing time constraints are omitted. Figs. 9 and 10 show that the comparison with 3 algorithms for $F(\boldsymbol{x}, \boldsymbol{t}^S)$ based on data in exponential distribution and normal distribution respectively. In Figs. 9 and 10, $F(\boldsymbol{x}, \boldsymbol{t}^S)$ of proposed hGA is smaller that that of each algorithms.

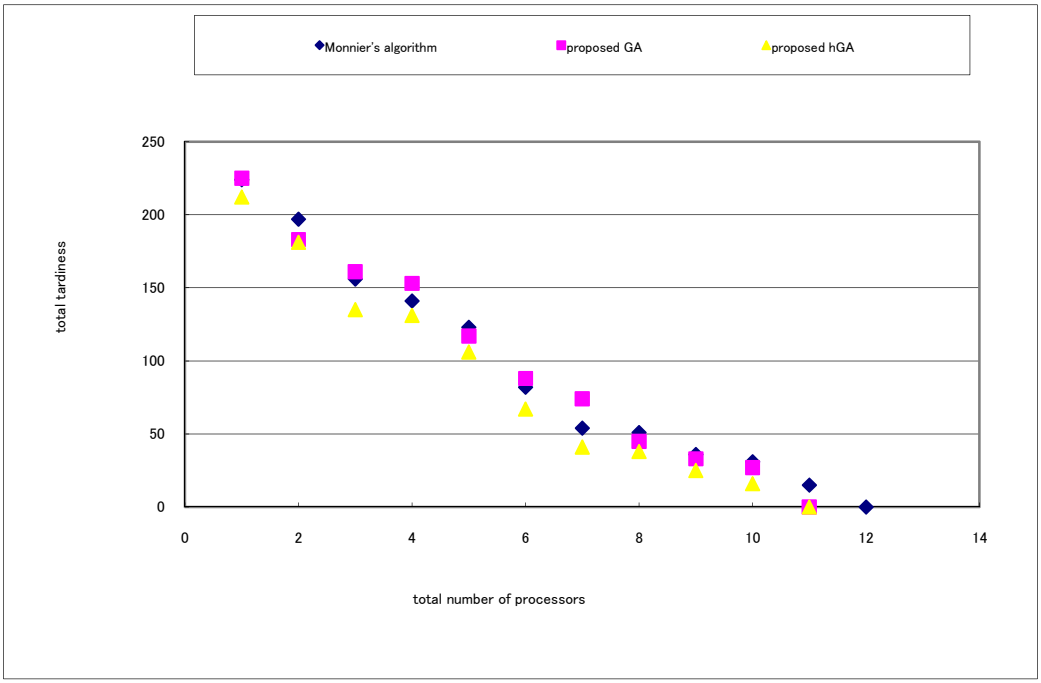


Fig. 7 Comparison with 3 algorithms for $F(x, t^S)$ based on exponential distribution.

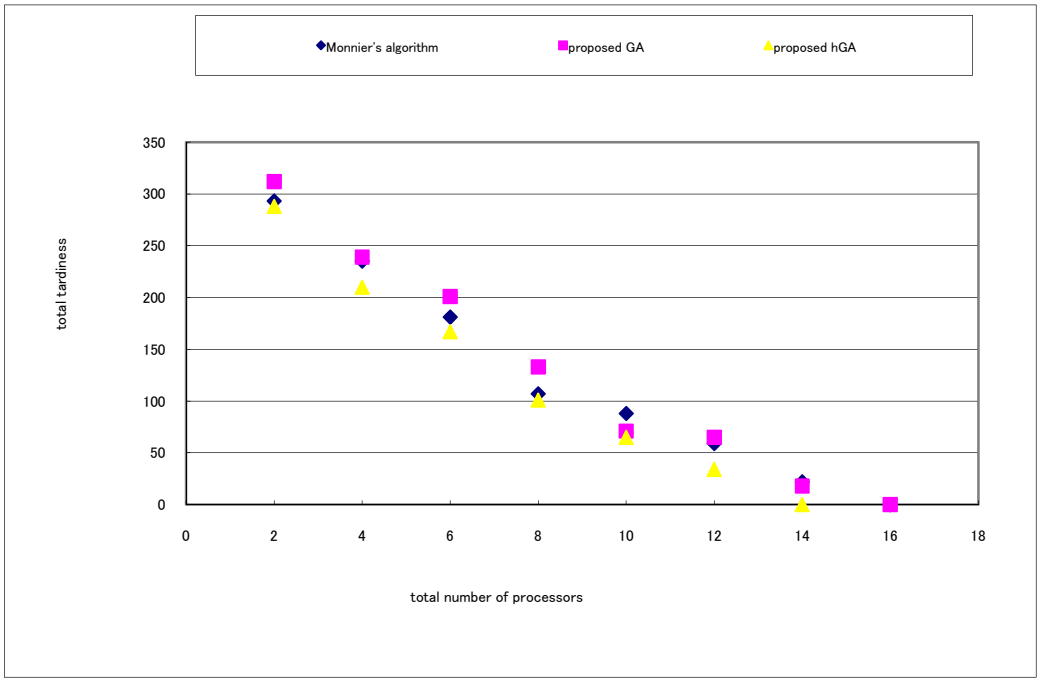


Fig. 8 Comparison with 3 algorithms for $F(x, t^S)$ based on normal distribution.

Table 7 Comparison with 3 algorithms without tardiness based on data in exponential distribution.

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors M	12	11	11
Makespan	35	37	39
Computing times (*msec*)	120	122	191
Average utilization of processors	0.518339	0.533981	0.532281

Table 8 Comparison with 3 algorithms without tardiness based on data in normal distribution.

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors M	16	16	14
makespan	47	50	49
Computing times (*msec*)	123	123	198
Average utilization of processors	0.500152	0.4909040	0.515035

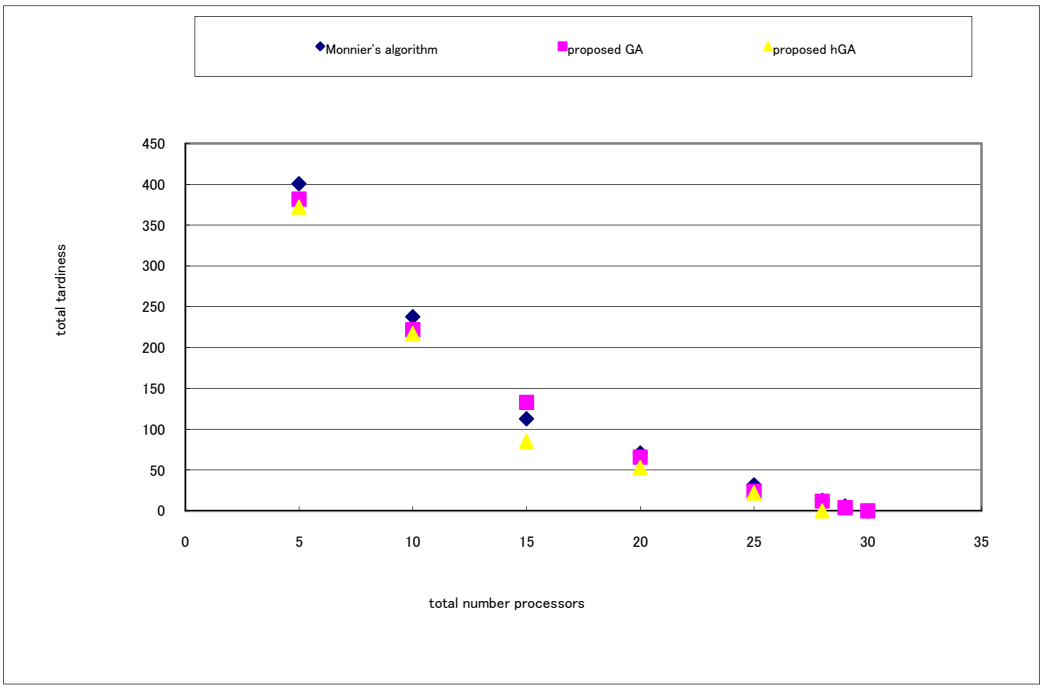


Fig. 9 Comparison with 3 algorithms for $F(x, t^S)$ based on exponential distribution

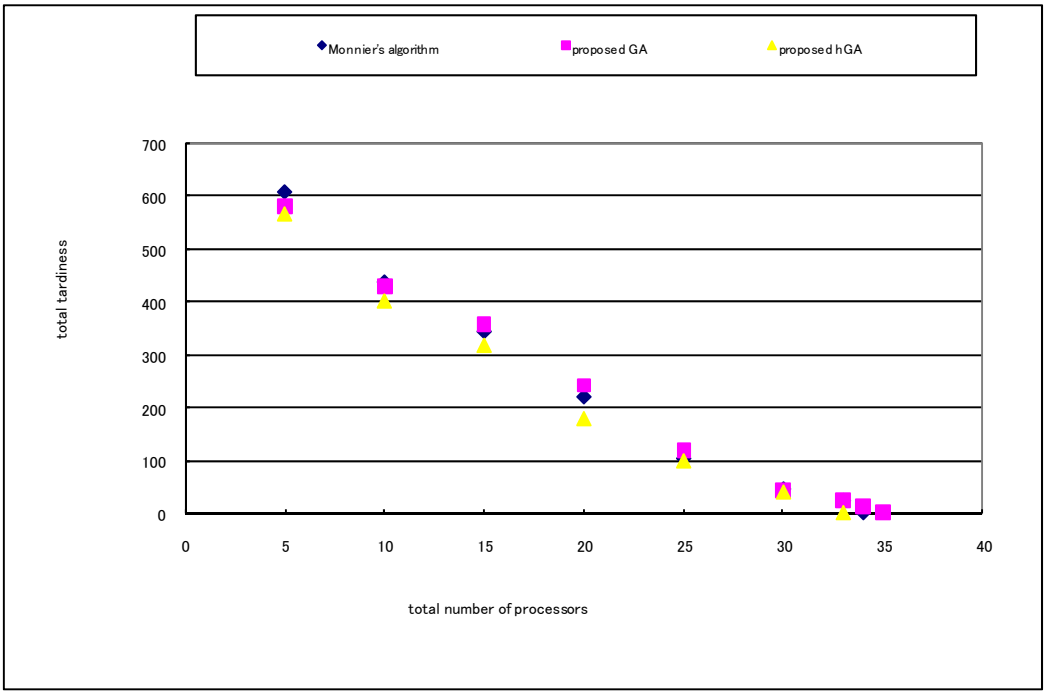


Fig. 10 Comparison with 3 algorithms for $F(x, t^S)$ based on normal distribution

Table 9 Comparison with 3 algorithms without tardiness based on data in exponential distribution.

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors *M*	30	30	28
makespan	103	101	106
Computing times (*msec*)	497	499	818
Average utilization of processors	0.433293	0.437284	0.477352

Table 10 Comparison with 3 algorithms without tardiness based on data in normal distribution.

Terms	Monnier's GA	Proposed GA	Proposed hGA
# of processors *M*	34	35	33
Makespan	132	130	137
Computing times (*msec*)	498	500	820
Average utilization of processors	0.523363	0.519823	0.527490

In Tables 9 and 10, some terms such as makespan, computing time and the utilization of processors are compared on the total number of processors without tardiness. Total number of processors without tardiness of proposed hGA is smaller than that of other algorithms and the average utilization of processors of hGA is more desirable than those of the others.

5. Conclusions

A new task scheduling algorithm is proposed in this paper. This algorithm is designed for non-preemptive tasks in soft real-time multiprocessor system with the communication time between processors and the precedence relationship between tasks. In solution algorithms, we proposed a method which combines the moGA (multiobjective genetic algorithm) and the SA. In this method, the convergence of GA is improved by introducing the probability of SA as the criterion for acceptance of new trial solution. The objective of proposed scheduling algorithm is to minimize the total tardiness and total number of processors used simultaneously. For these conflicting objectives, this paper combines AWA (adaptive weight approach). From the numerical results, the results of the proposed moGA are better than that of other algorithms.

This determines the next step of our study. We plan to design real-time tasks scheduling algorithm in heterogeneous multiprocessors system.

Reference

[1] Krishna, C. M., and Kang, G. S. 1997. *Real-Time System*. McGraw-Hill.

[2] Diaz, J. L., Garcia, D. F., and Lopez, J. M. 2004. "Minimum and Maximum Utilization Bounds for Multiprocessor Rate Monotonic Scheduling." *IEEE Transactions on Parallel and Distributed Systems* 15 (7): 642-53.

[3] Bernat, G., Burns, A., and Liamosi, A. 2001. "Weakly Hard Real-Time Systems." *IEEE Transactions on Computer Systems* 50 (4): 308-21.

[4] Kim, M. H., Lee, H. G., and Lee, J. W. 1997. "A Proportional-Share Scheduler for Multimedia Applications." In *Proceedings of Multimedia Computing and Systems*, 484-91.

[5] Yoo, M. R. 2002. "A Scheduling Algorithm for Multimedia Process." Ph.D. dissertation, University of Yeoung Nam, Korea.

[6] Yalaoui, F., and Chu, C. 2002. "Parallel Machine Scheduling to Minimize Total Tardiness." *International Journal of Production Economics* 76 (3): 265-79.

[7] Mitra, H., and Ramanathan, P. 1993. "A Genetic Approach for Scheduling Non-preemptive Tasks with Precedence and Deadline Constraints." In *Proceedings of the 26th Hawaii International Conference on System Sciences*, 556-64.

[8] Lin, M., and Yang, L. 1999. "Hybrid Genetic Algorithms for Scheduling Partially Ordered Tasks in a Multi-processor Environment." In *Proceedings of the 6th International Conference on Real-Time Computer Systems and Applications,* 382-7.

[9] Monnier, Y., Beauvais, J. P., and Deplanche, A. M. 1998. "A Genetic Algorithm for Scheduling Tasks in a Real-Time Distributed System." In *Proceedings of 24th Euromicro Conference,* 708-14.

[10] Oh, J., and Wu, C. 2004. "Genetic-Algorithm-Based

Real-time Task Scheduling with Multiple Goals." *Journal of Systems and Software* 71 (3): 245-58.

[11] Kim, H. C, Hayashi, Y., and Nara, K. 1997. "An Algorithm for Thermal Unit Maintenance Scheduling through Combined Use of GA, SA and TS." *IEEE Transactions on Power Systems* 12 (1): 329-35.

[12] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983. "Optimization by Simulated Annealing." *Science* 220 (4598): 671-80.

[13] Gen, M., and Cheng, R. 2000. *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons.

[14] Deb, K. 2001. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

[15] Gen, M., and Cheng, R. 1997. *Genetic Algorithms & Engineering Design*. John Wiley & Sons.

[16] Jackson, L. E., and Rouskas, G. N. 2003. "Optimal Quantization of Periodic Task Requests on Multiple Identical Processors." *IEEE Transactions on Parallel and Distributed Systems* 14 (8): 795-806.

[17] Al-Sharaeh, S., and Wells, B. E. 1996. "A Comparison of Heuristics for List Schedules Using the Box-method and P-method for Random Digraph Generation." In *Proceedings of the 28th Southeastern Symposium on System Theory,* 467-71.