Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 34 (2007) 3084–3098

computers & operations research

www.elsevier.com/locate/cor

Scheduling algorithm for real-time tasks using multiobjective hybrid genetic algorithm in heterogeneous multiprocessors system

Myungryun Yoo*, Mitsuo Gen

Graduate School of Information, Production & Systems, Waseda University, 2-7 Hibikino, Wakamatsuku, Kitakyushu, 808-0135 Japan

Available online 20 December 2005

Abstract

The scheduling problem for real-time tasks on multiprocessor is one of the NP-hard problems. This paper proposes a new scheduling algorithm for real-time tasks using multiobjective hybrid genetic algorithm (mohGA) on heterogeneous multiprocessor environment. In solution algorithms, the genetic algorithm (GA) and the simulated annealing (SA) 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 proposed algorithm has a multiobjective to minimize the total tardiness and completion time simultaneously. For these conflicting objectives, this paper combines adaptive weight approach (AWA) that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point.

The effectiveness of the proposed algorithm is shown through simulation studies. In simulation studies, the results of the proposed algorithm are better than that of other algorithms.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Multiobjective hybrid genetic algorithm; Scheduling for real-time tasks; Heterogeneous multiprocessor

1. Introduction

Real-time systems are characterized by computational activities with timing constraints and classified into two categories: hard real-time system and soft real-time system. In hard real-time system, the violation of timing constraints of a certain task should not be acceptable. The consequences of not executing a task before its deadline may lead to catastrophic consequences in certain environments i.e., in patient monitoring systems, nuclear plant control, etc. The goal of the scheduling algorithms in hard real-time system is to meet all tasks' deadlines, in other words, to keep the feasibility of scheduling through admission control. On the other hand, in the soft real-time system (e.g. telephone switching system, image processing, etc.), in which usefulness of results produced by a task decreases over time after the deadline expires without causing any damage to the controlled environment [1].

Traditionally, the performance criteria of an algorithm for a task scheduling problem (TSP) are throughput, utilization of processors, waiting time of tasks, etc. In a hard real-time system, the performance of the scheduling algorithm is measured by its ability to generate a feasible schedule for a set of real-time tasks. Typically, there is rate monotonic (RM) and earliest deadline first (EDF) derived scheduling algorithms for a hard real-time system with a uniprocessor [2,3].

* Corresponding author. Tel.: +81 80 5210 6409.
E-mail addresses: mryoo@toki.waseda.jp (M. Yoo), gen@waseda.jp (M. Gen).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.016

They guarantee the optimality in somewhat restricted environments. However, these algorithms have some drawbacks in coping with a soft real-time system related resource utilization and pattern of degradation under the overloaded situation. The objective of the scheduling task in soft real-time system is to minimize total tardiness. With the growth of soft real-time applications, the necessity for scheduling algorithms for soft real-time systems is on the increase and several researches for a soft real-time system are reported. Rate regulating proportional share (rrPS) scheduling algorithm, based on stride scheduler by Kim et al. [4] and modified proportional share (mPS) scheduling algorithm by Yoo [5] are designed for tasks in soft real-time systems. However, these algorithms also cannot show the graceful degradation of performance under an overloaded situation and are restricted in a uniprocessor system.

In a multiprocessor system, task scheduling is more difficult than that in a uniprocessor system. The optimal assignment of tasks to a multiprocessor is, in almost all practical cases, an NP-hard problem [6]. Consequently various modern heuristics based algorithms have been proposed for practical reasons.

Recently, several approaches of the genetic algorithm (GA) are proposed. Mitra and Ramanathan proposed a GA for scheduling of nonpreemptive tasks with precedence and deadline constraints [7]. Lin and Yang presented a hybrid GA, where different operators are applied at a 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 scheduling problem for real-time nonpreemptive tasks [9]. However, these algorithms have only one objective such as minimizing cost, completion time or total tardiness. Oh and Wu presented a multiobjective GA for scheduling nonpreemptive tasks in a soft real-time system with multiprocessors [10]. However, this algorithm did not refer to conflict between objectives, the so called Pareto optimum, and assume that the performance of all processors is the same. Theys et al. presented a static scheduling algorithm using GA on a heterogeneous system [11]. And, Page and Naughton presented a dynamic scheduling algorithm using GA on a heterogeneous system [12]. Dhodhi et al. presented a new encoding method of GA for task scheduling on a heterogeneous system [13]. However, these algorithms are designed for general tasks without time constraints.

In this paper, we propose a new scheduling algorithm for nonpreemptive tasks with a precedence relationship in a soft real-time heterogeneous multiprocessor system. In solution algorithms, the multiobjective genetic algorithm (mohGA) and the simulated annealing (SA) are cooperatively used [14]. In this method, the convergence of GA is improved by introducing the probability of SA [15] as the criterion for acceptance of a 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 and the completion time simultaneously. For these conflicting objectives, this paper combines adaptive weight approach (AWA) that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point [16].

The rest of the paper is organized as follows: In Section 2, we explain a scheduling problem for soft real-time tasks (SP-srt) 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. Scheduling problem for soft real-time tasks and mathematical model

In this study, we consider the problem of scheduling the tasks with precedence and a timing constrained task graph on a set of heterogeneous processors in a way that minimizes the total tardiness f_1 and completion time f_2 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.

The scheduling problem for soft real-time tasks (SP-srt) is formulated under the following assumptions: computation time and deadline of each task are known. A time unit is an artificial time unit. The scheduling problem for soft real-time tasks (SP-srt) in the heterogeneous multiprocessor system to minimize the total tardiness and completion time is formulated as follows:

$$\min f_1 = \sum_{i=1}^{N} \max \left\{ 0, \sum_{m=1}^{M} (t_i^{\mathrm{S}} + c_{im} - d_i) x_{im} \right\}, \tag{1}$$

$$\min f_2 = \max_i \{t_i^{\mathrm{F}}\}, \tag{2}$$

$$\text{s.t.} \quad t_i^{\mathrm{E}} \leqslant t_i^{\mathrm{S}} \ \ \forall i, \tag{3}$$

$$t_i^{\mathrm{E}} \geqslant t_j^{\mathrm{E}} + \sum_{m=1}^{M} c_{jm} x_{jm}, \quad \tau_j \in \mathrm{pre}(\tau_i) \ \ \forall i, \tag{4}$$

$$\sum_{m=1}^{M} x_{im} = 1 \ \ \forall i, \tag{5}$$

$$x_{im} \in \{0, 1\} \ \ \forall i, m. \tag{6}$$

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

N	: total number of tasks
M	: total number of processors used
τ_i	: the ith task, $i = 1, 2, \ldots, N$
e_{ij}	: precedence relationship between task τ_i and task τ_j

$$e_{ij} = \begin{cases} 1 & \text{if there are precedence relationship between task } \tau_i \text{ and task } \tau_j, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

c_{im}	:computation time of task τ_i on mth processor
d_i	: deadline of task τ_i
$\mathrm{pre}{*}(\tau_i)$	: set of all predecessors of task τ_i
$\mathrm{suc}{*}(\tau_i)$	: set of all successors of task τ_i
$\mathrm{pre}(\tau_i)$	: set of immediate predecessors of task τ_i
$\mathrm{suc}(\tau_i)$	: set of immediate successors of task τ_i
t_i^{E}	: earliest start time of task τ_i

$$t_i^{\mathrm{E}} = \begin{cases} 0 & \text{if } \neg\exists\tau_j : e_{ji} \in E \\ \max_{\tau_j \in \mathrm{pre}*(\tau_i)} \left\{ t_j^{\mathrm{E}} + \sum_{m=1}^{M} c_{jm} x_{jm} \right\} & \text{otherwise} \end{cases} \quad \forall i. \tag{8}$$

t_i^{F} : finish time of task τ_i

$$t_i^{\mathrm{F}} = \min\{t_i^{\mathrm{S}} + c'_{im}, d_i\} \quad \forall i, \tag{9}$$

where $c'_{im} = c_{im} | x_{im} \ \ \forall i$.

Decision variables:

t_i^{S} : real start time of task τ_i

$$x_{im} = \begin{cases} 1 & \text{if processor } p_m \text{ is selected for task } \tau_i, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Eqs. (1) and (2) are the objective function in this scheduling problem. Eq. (1) means to minimize total tardiness of tasks and Eq. (2) means to minimize completion time. Constraint conditions are shown from Eqs. (3) to (6). Eq. (3) means that the task can be started after its earliest start time begins its deadline. Eq. (4) defines the earliest start time of the task based on precedence constraints. Eq. (5) means that every task is processed on one processor at a time. In a homogeneous system, the computation time of a task is the same on all processors. However, the computation time of a task is dependent on the processor. Therefore, we have one more decision variable in a heterogeneous system but more than that in a homogeneous system.

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 the precedence relationship. The allocation information part denotes the processor to which the task is allocated.

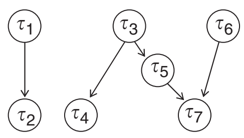
Encoding procedure for the scheduling problem for soft real-time tasks (SP-srt) will be written as follows:

```
procedure: Encoding for SP-srt
input: task graph data set, total number of processors $M$
output: $u(\cdot), v(\cdot)$
begin
    $l \leftarrow 1, w \leftarrow \phi$;
    while $(T \neq \phi)$
        $W \leftarrow W \cup \arg\{\tau_i | \text{pre}^*(\tau_i) = \phi, \forall i\}$;
        $T \leftarrow T - \{\tau_i\}, i \in W$;
        while $(W \neq \phi)$
            $j \leftarrow \text{random}(W)$;
            $u(l) \leftarrow j$;
            $W \leftarrow W - \{j\}$;
            $\text{pre}^*(\tau_i) \leftarrow \text{pre}^*(\tau_i) - \{\tau_j\}, \forall i$;
            $m \leftarrow \text{random}[1:M]$;
            $v(l) \leftarrow m$;
            $l \leftarrow l + 1$;
        end
    end
    output $u(\cdot), v(\cdot)$;
end
```

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

Fig. 1 represents the example of this encoding procedure.

task graph



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

		cim			
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

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. 1. Example of encoding procedure.

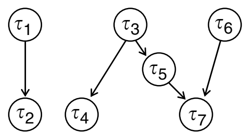
Decoding procedure will be written as follows:

```
procedure: Decoding for SP-srt
input: task graph data set,
            chromosome $u(\cdot)$, $v(\cdot)$
output: schedule set $S$,
            total tardiness of tasks $f_1$
            makespan $f_2$
begin
    $l \leftarrow 1$, $F \leftarrow 0$, $S \leftarrow \phi$;
    while ($l \leqslant N$)
        $i \leftarrow u(l)$;
        $m \leftarrow v(l)$;
        if (exist suitable idle time) then
            insert ($i$);
        start($i$);
        update_idle();
        $f_1 \leftarrow f_1 + \max\{0, (t_i^{\mathrm{S}} + c_{im} - d_i)\}$;
        $S \leftarrow S \cup \{(i, m : t_i^{\mathrm{S}} - t_j^{\mathrm{F}})\}$;
        $l \leftarrow l + 1$;
    end
    $f_2 \leftarrow \max\{t_i^{\mathrm{F}}\}$
    output $S$, $f_1$, $f_2$
end
```

Here 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 ith 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_1 and f_2 and schedule set S is generated through this procedure.

Fig. 2 represents the example of a decoding procedure with chromosome in Fig. 1.

task graph



data set

i	suc(τ_i)	c_{i1}	c_{i2}	c_{i3}	d_i
		cim			
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	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

$f_1 = 5$

$f_2 = 30$

S={ (3,1:0-10), (6, 2:0-25),(1,3:0-5), (4,1:10-20), (5,3:10-20),(2,3:5-10), (7, 1:25-30)}

schedule S={(i,m:t_i^S-t_i^F)}
i: task number
m: processor number
t_i^S: start time of i th task
t_i^F: finish time of i th task

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

Gantt chart

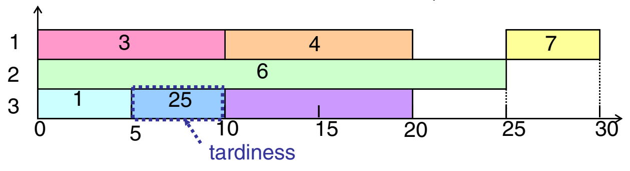


Fig. 2. Example of decoding procedure.

3.2. Evolution function and selection

The multiobjective optimization problems have been receiving growing interest from researchers with various backgrounds since early 1960. Recently, GAs have been received considerable attention as a novel approach to multiobjective optimization problems, resulting in a fresh body of research and applications known as genetic multiobjective optimizations [17,18].

In this paper, we combine the AWA [16] that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point.

For the examined solutions at each generation, we define two extreme points (maximum: $\boldsymbol{f}^{+}$, minimum: $\boldsymbol{f}^{-}$)

$$\boldsymbol{f}^{+} = \{f_1^{\max}, f_2^{\max}\}, \tag{11}$$

$$\boldsymbol{f}^{-} = \{f_1^{\min}, f_2^{\min}\}, \tag{12}$$

where $f_q^{\max}$ and $f_q^{\min}$ are the maximal and minimal values for the qth objective as defined by the following equations:

$$f_q^{\max} = \max_k\{f_q(V_k)\}, \quad q = 1, 2, \tag{13}$$

$$f_q^{\min} = \min_k\{f_q(V_k)\}, \quad q = 1, 2. \tag{14}$$

The equation driven above is a hyper plane defined by the following extreme points in current solutions

$$\begin{array}{ll} [f_1^{\max} & f_2^{\min}], \\ [f_1^{\min} & f_2^{\max}]. \end{array} \tag{15}$$

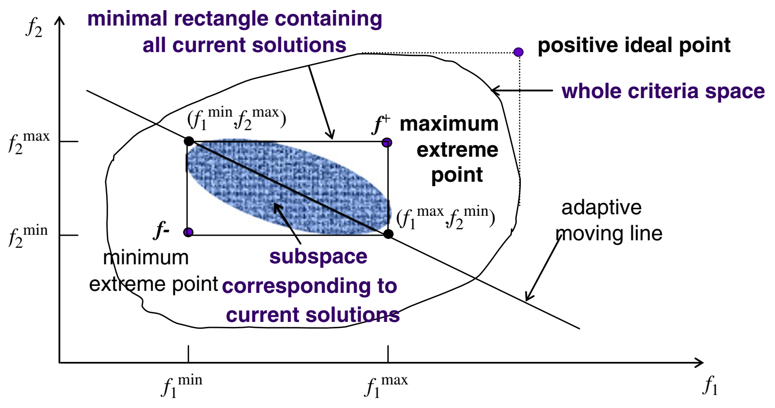


Fig. 3. Adaptive weights and adaptive hyper plane.

Adaptive moving line defined by the extreme points $(f_1^{\max}, f_2^{\min})$ and $(f_1^{\min}, f_2^{\max})$ are shown in Fig. 3. The weighted-sum objective function for a given chromosome V_k is given by the following equation:

$$F(V_k) = \sum_{q=1}^{2} w_q f_q(V_k) = \sum_{q=1}^{2} \frac{f_q(V_k)}{f_q^{\max} - f_q^{\min}}, \tag{16}$$

where w_q is adaptive weight for objective q:

$$w_q = \frac{1}{f_q^{\max} - f_q^{\min}}, \quad q = 1, 2. \tag{17}$$

The evaluation function is designed as follows:

$$eval(V_k) = 1/F(V_k) = \frac{1}{\sum_{q=1}^{2} \frac{f_q(V_k)}{f_q^{\max} - f_q^{\min}}}. \tag{18}$$

For selection, the commonly used strategy called roulette wheel selection [9,19] 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, $N$];
    $u_1'(\cdot) \leftarrow u_1(\cdot)$;
    $v_1'(\cdot) \leftarrow v_1[1:r]//v_2[r+1:N]$;
```

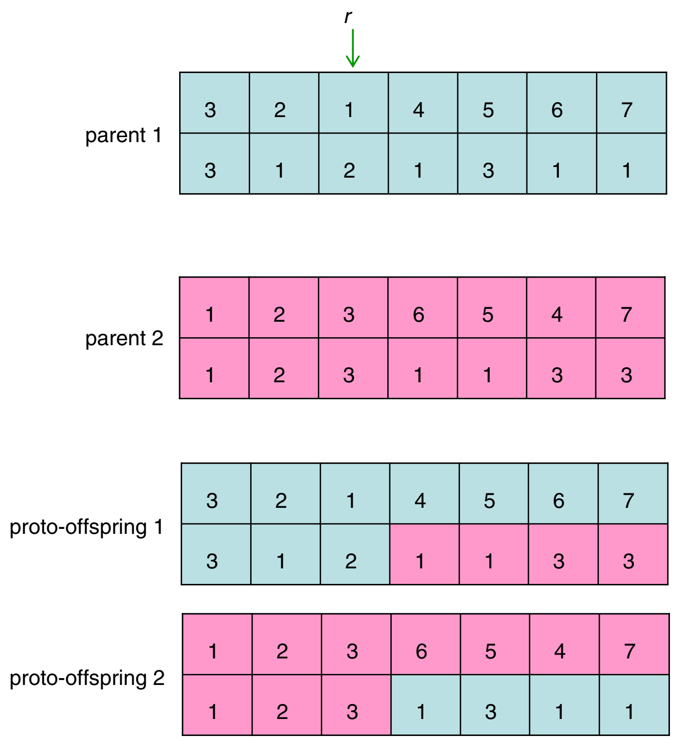


Fig. 4. Example of one-cut crossover.

```
    $u'_2(\cdot) \leftarrow u_2(\cdot)$;
    $v'_2(\cdot) \leftarrow v_2[1:r]//v_1[r+1:N]$;
    **output** proto-offspring chromosomes
            $u'_1(\cdot), v'_1(\cdot), u'_2(\cdot), v'_2(\cdot)$;
end
```

Here $u'(\cdot)$, $v'(\cdot)$ 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 [20].

3.4. Improving of convergence by the probability of SA

The convergence speed to the local optimum of the GA can be improved by adopting the probability of SA. Simulated annealing means the simulation of the annealing process of metal. If the temperature is lowered carefully from a high temperature in the annealing process, the melted metal will produce the crystal at 0 K. Kirkpatrick developed an algorithm that finds the optimal solution by substituting the random movement of the solution for the fluctuation of a particle in the system in the annealing process and making the objective function value correspond to the energy of the system, which decreases (involving the temporary increase by Boltzman's probability) with the descent of temperature [14,15]. 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 the early stages of the searching process. However, in later stages, a string with a 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 by the robability of SA
input: parent chromosome $V$
       proto-offspring chromosomes $V$
       temperature $T_0$
```

```
        cooling rate of SA $\rho$
output: offspring chromosomes $V'$
Begin
    $T_t \leftarrow T_{t-1} \times \rho$;
    $r \leftarrow$ random[0, 1];
    $\Delta E \leftarrow eval(V)$-eval$(V)$;
    if $(\Delta E > 0 \| r < \text{Exp}(\Delta E/T))$
            $V' \leftarrow V$;
    else
            $V' \leftarrow V$;
    output offspring chromosomes $V'$
end
```

In this procedure, V and V' are mean parent chromosome and proto-offspring chromosome. V'' means offspring chromosome which is produced by this procedure. 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} [16]. The adoption of newly created offspring produced by crossover is decided by probability of SA. Superior offspring to its parents adopted, and will survive next generation. However, inferior offspring to its parents adopted conditionally by the criterion of Boltzman's Probability ($r < \exp(\Delta E/T)$).Then the new population is built through evaluating chromosomes and selection.

The algorithm terminates when *maxGen* generations are completed. We use a fixed number of generations as the stopping criterion. Our proposed mohGA obeys the following algorithm:

```
algorithm: SP-srt by mohGA
input: task graph data set
output: best schedule set $S$
begin
    $t \leftarrow 0$;
    initialize $P(t)$ by encoding routine;
    evaluate $f_1, f_2$ of $P(t)$ by decoding routine;
    create Pareto $R(P)$;
    fitness $eval(P)$ by AWA;
    while (not termination condition) do
        one-cut crossover $P(t)$ to yield $C'(t)$;
        altering mutation $P(t)$ to yield $C'(t)$;
        improving of GA by the probability of SA;
        evaluate $f_1, f_2$ by $C(t)$ by decoding routine;
        update Pareto $R(P, C)$;
        fitness $eval(P, C)$ by AWA;
        select $P(t+1)$ from $P(t)$ and $C(t)$;
        $t \leftarrow t + 1$;
    end
    output best schedule set $S$;
end
```

In this algorithm, Fonseca and Fleming's Pareto ranking method is used for creating and updating Pareto set. In this method, all nondominated chromosomes are assigned to rank 1, and any other chromosomes are assigned a rank equal to the number of its dominating chromosomes plus one. This method proceeds by sorting the population according to the ranks, and ties may be broken by random choice. The Pareto set are preserved separately from the population pool. The best schedule set $S = \{i, m\colon t_i^{\text{S}} - t_i^{\text{F}}\}$ means the final scheduling list decoded from the best chromosome.

4. Validation

To validate proposed mohGA, several numerical tests are performed. We compared the proposed mohGA with Monnier's algorithm by Monnier et al. [9], Theys's algorithm by Theys et al. [11] and mohGA which is not combined with SA. The Monnier's algorithm is designed for soft real-time tasks on a homogeneous multiprocessors system and the Theys's algorithm is designed for general tasks on a heterogeneous multiprocessors system.

Numerical tests are performed with a randomly generated task graph. We use P-Method [21] for the 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 the tasks' computation time and deadline, we use random numbers based on exponential distribution and normal distribution as follows:

$$\begin{aligned}
c_{im}^{\mathrm{E}} &= \text{random value based on exponential distribution with mean 5}\\
c_{im}^{\mathrm{N}} &= \text{random value based on normal distribution with mean 5}\\
r^{\mathrm{E}} &= \text{random value based on exponential distribution with mean } c_i^{\mathrm{E}}\\
r^{\mathrm{N}} &= \text{random value based on normal distribution with mean } c_i^{\mathrm{N}}\\
d_i^{\mathrm{E}} &= t_i^{\mathrm{E}} + \max\{c_{im}^{\mathrm{E}}\ \forall m\} + r^{\mathrm{E}}\\
d_i^{\mathrm{N}} &= t_i^{\mathrm{E}} + \max\{c_{im}^{\mathrm{N}}\ \forall m\} + r^{\mathrm{N}},
\end{aligned}$$

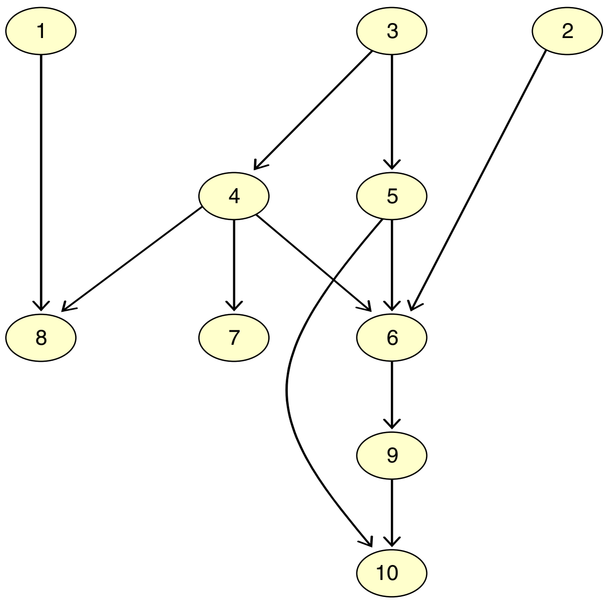


Fig. 5. Task graph with 10 tasks.

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

i	suc(τ_i)	C_{im}			d_i
		c_{i1}	c_{i2}	c_{i3}	
1	8	5	3	10	13
2	6	3	7	12	17
3	4,5	3	4	1	12
4	6,7,8	2	16	6	12
5	6,10	12	2	4	27
6	9	2	4	7	24
7	—	2	15	4	13
8	—	3	5	4	18
9	10	5	5	8	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
		c_{i1}	c_{i2}	c_{i3}	c_{i4}	
1	8	5	3	11	8	19
2	6	6	5	4	13	9
3	4,5	11	8	6	7	18
4	6,7,8	10	13	5	6	37
5	6,10	10	13	8	11	38
6	9	2	11	11	3	37
7	—	3	10	11	8	44
8	—	4	12	10	5	30
9	10	6	9	7	10	37
10	—	11	12	6	4	58

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

The parameters of GA were set to 0.7 for crossover (p_{C},), 0.3 for mutation (p_{M},), and 30 for population size (*popSize*). Probabilities for crossover are tested from 0.5 to 0.8, from 0.001 to 0.4 for mutation, with the increments 0.05 and 0.001, respectively. For population size, individuals from 20 to 200 are tested. Each combination of parameters is tested 20 times, respectively. The best combination of parameters is selected by an average performance of 20 runs. The parameters of SA were set to 500 for the temperature T and to 0.98 for the cooling rate ρ.

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

4.1. Example 1

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

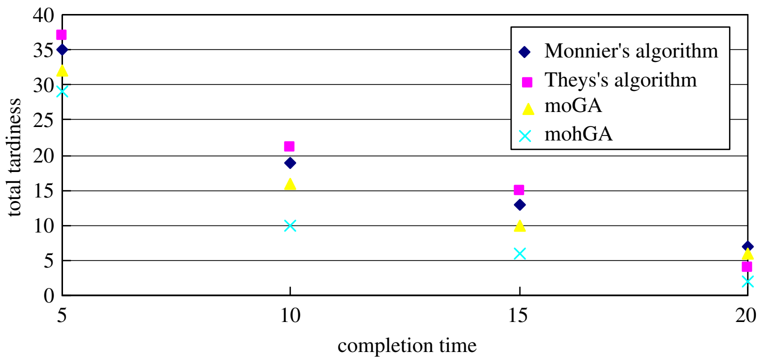


Fig. 6. Pareto solution in 10 tasks (exponential distribution).

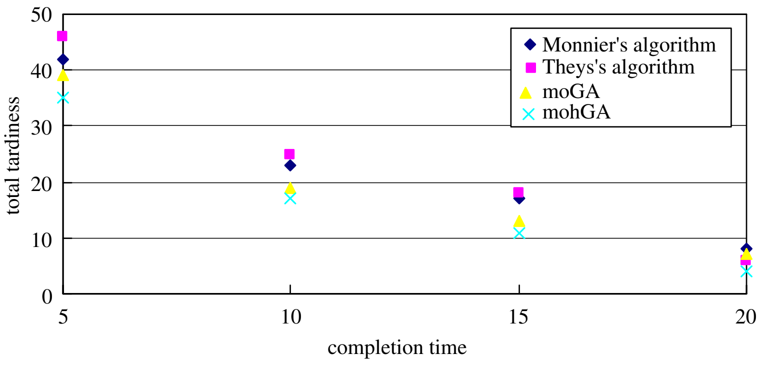


Fig. 7. Pareto solution in 10 tasks (normal distribution).

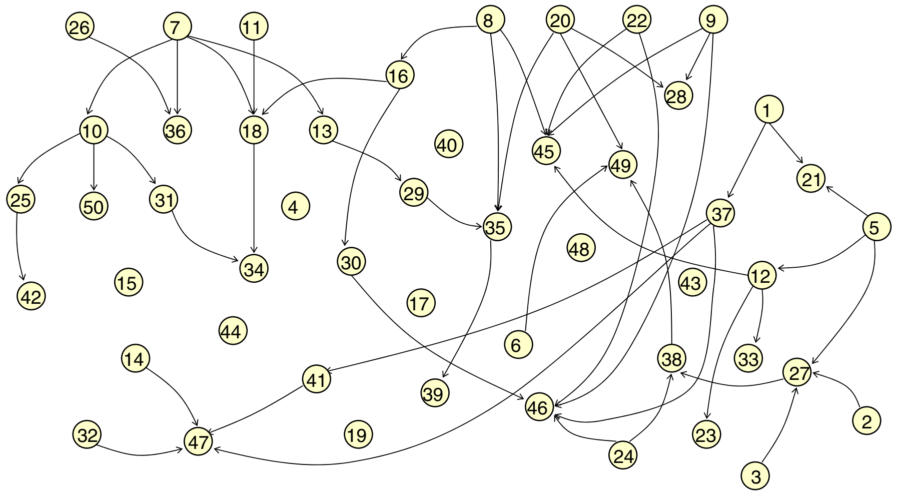


Fig. 8. Task graph with 50 tasks.

For generality of the numerical test, Tables 1 and 2 are data sets generated by exponential distribution and normal distribution of task graph in Fig. 5, respectively. This example is simulated in various processors. Tardiness became 0 in 3 and 4 processors, respectively. The data and the result are showed in this paper.

Figs. 6 and 7 represent the Pareto solution of proposed mohGA and that of other algorithms on 3 and 4 processors, respectively. In these figures, the Pareto solution curve by proposed mohGA is closer to the ideal point

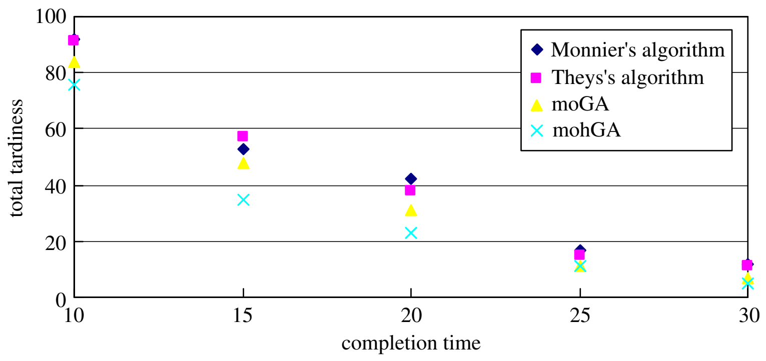


Fig. 9. Pareto solution in 50 tasks (exponential distribution).

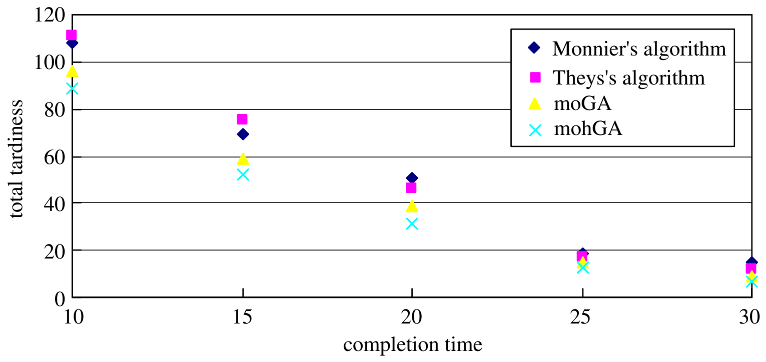


Fig. 10. Pareto solution in 50 tasks (normal distribution).

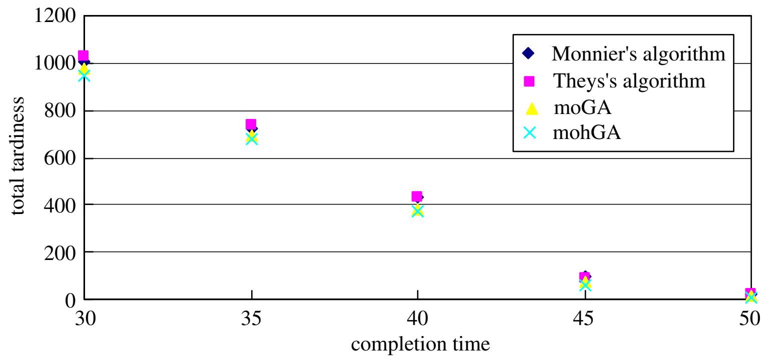


Fig. 11. Pareto solution in 1000 tasks (exponential distribution).

than that of other algorithms. The coordinate of the ideal point total tardiness is $0(f_1 = 0)$ and completion time is 1 $(f_2 = 1)$.

4.2. *Example 2*

In this example, we use a task graph with 50 tasks. Fig. 8 represents the task graph of the 50 tasks case.

Tardiness became 0 in 8 and 10 processors, respectively.

Figs. 9 and 10 represent the Pareto solution of the proposed mohGA and that of other algorithms in this example. In these figures, the Pareto solution curve by the proposed mohGA is closer to ideal point than that of other algorithms.

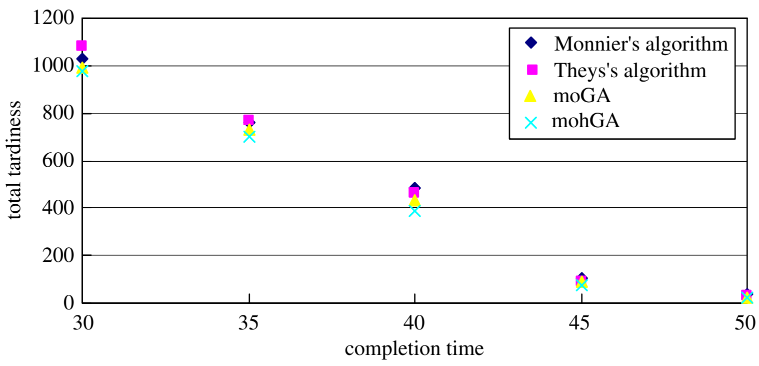


Fig. 12. Pareto solution in 1000 tasks (normal distribution).

4.3. *Example 3*

In this example, we use a task set with 1000 tasks. The data set of 1000 tasks and the task graph are omitted. Tardiness became 0 in 132 and 137 processors, respectively.

Figs. 11 and 12 represent the Pareto solution of the proposed mohGA and that of other algorithms in this example. In these figures, the Pareto solution curve proposed by mohGA is closer to ideal point than that of other algorithms.

5. Conclusions

A new task scheduling algorithm is proposed in this paper. This algorithm is designed for nonpreemptive tasks with the precedence relationship between tasks on a soft real-time heterogeneous multiprocessor system. The objective of the proposed scheduling algorithm is to minimize the total tardiness and completion time simultaneously. We use multiobjective hybrid genetic algorithm (mohGA) combined with simulated annealing (SA) for this problem. In this method, the convergence of genetic algorithm (GA) is improved by introducing the probability of SA as the criterion for acceptance of the new trial solution. From the numerical results, the results of the proposed mohGA are better than that of other algorithms.

We plan to design a scheduling algorithm for real-time tasks on heterogeneous multiprocessors system with communication time between processors.

Acknowledgements

This work was partly supported by Waseda University Grant for Special Research Projects 2004 and the Ministry of Education, Science and Culture, the Japanese Government: Grant-in-Aid for Scientific Research (no. 17510138).

References

[1] Krishna CM, Kang GS. Real-time system. New York: McGraw-Hill; 1997.

[2] Diaz JL, Garcia DF, Lopez JM. Minimum and maximum utilization bounds for multiprocessor rate monotonic scheduling. IEEE Transactions on Parallel and Distributed Systems 2004;15(7):642–53.

[3] Bernat G, Burns A, Liamosi A. Weakly hard real-time systems. IEEE Transactions on Computer Systems 2001;50(4):308–21.

[4] Kim MH, Lee HG, Lee JW. A proportional-share scheduler for multimedia applications. In: Proceedings of the multimedia computing and systems 1997. p. 484–91.

[5] Yoo MR. A scheduling algorithm for multimedia process. PhD dissertation, University of YeoungNam, Korea, 2002 [in Korean].

[6] Yalaoui F, Chu C. Parallel machine scheduling to minimize total tardiness. International Journal of Production Economics 2002;76(3):265–79.

[7] Mitra H, Ramanathan P. A genetic approach for scheduling non-preemptive tasks with precedence and deadline constraints. In: Proceedings of the 26th Hawaii international conference on system sciences, 1993. p. 556–64.

[8] Lin M, Yang L, 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, 1999. p. 382–87.

[9] Monnier Y, Beauvais JP, Deplanche AM. A genetic algorithm for scheduling tasks in a real-time distributed system. In: Proceedings of the 24th euromicro conference, 1998. p. 708–14.
[10] Oh J, Wu C. Genetic-algorithm-based real-time task scheduling with multiple goals. Journal of Systems and Software 2004;71(3):245–58.
[11] Theys MD, Braun TD, Siegal HJ, Maciejewski AA, Kwok YK. 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; 2001. p. 135–78 [chapter 6].
[12] Page AJ, Naughton TJ. Dynamic task scheduling using genetic algorithm for heterogeneous distributed computing. In: Proceedings of the 19th IEEE international parallel and distributed processing symposium, 2005. p. 189.1.
[13] Dhodhi MK, Ahmad I, Yatama A, Ahmad I. An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. Journal of Parallel and Distributed Computing 2002;62:1338–61.
[14] Kim HC, Hayashi Y, Nara K. An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS. IEEE Transactions on Power Systems 1997;12(1):329–35.
[15] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[16] Gen M, Cheng R. Genetic algorithms & engineering optimization. New York: Wiley; 2000.
[17] Deb K. Multi-objective optimization using evolutionary algorithms. New York: Wiley; 2001.
[18] Tsujimura Y, Takeno T. Effect of service time constraint in permutation representation method for vehicle routing problem. In: Proceeding of the fourth Asia–Pacific conference evolution and learning, 2002. p. 100–4.
[19] Gen M, Cheng R. Genetic algorithms & engineering design. New York: Wiley; 1997.
[20] Jackson LE, Rouskas GN. Optimal quantization of periodic task requests on multiple Identical processors. IEEE Transactions on Parallel and Distributed Systems 2003;14(8):795–806.
[21] Al-Sharaeh S, Wells BE. 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, 1996. p. 467–71.