Multiobjective GA for Real Time Task Scheduling

Myungryun Yoo, and Takanori Yokoyama

***Abstract*—This paper proposes a new real time task scheduling algorithm with GA. The proposed algorithm has multiobjective to minimize the total tardiness and total number of processors used 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.**

***Index Terms*— adaptive weight approach, genetic algorithm, multiobjective, simulated annealing**

I. Introduction

REAL-time systems are classified into two categories: hard real-time system and soft real-time system. 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].

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 Rate Monotonic (RM) and Earliest Deadline First (EDF) 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. As the growing of soft real time applications, the necessity of scheduling algorithm for soft real-time system is on increase. Several researches for soft real time system are reported [4], [5]. However, these algorithms also can not show the graceful degradation of performance under the overloaded situation. Furthermore, 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.

This work supported in part by JSPSKAKENHI Grant Number 15K00084.

Myungryun Yoo is with the Department of Computer Science, Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, Tokyo, Japan (corresponding author to provide phone: +81-3-5707-0104; e-mail: yoo@cs.tcu.ac.jp).

Takanori Yokoyama is with the Department of Computer Science, Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, Tokyo, Japan (e-mail: yokoyama@cs.tcu.ac.jp)

Recently, several approaches with single objective using Genetic Algorithm (GA) are proposed [7]-[9]. These algorithms have only one objective such as minimizing cost, end time, total tardiness.

In this paper, we propose a new scheduling algorithm for nonpreemptive tasks in soft real-time multiprocessor system with the communication time between processors and the precedence relationship between tasks. 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 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 [10].

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

In this study, we consider the problem of scheduling the tasks of precedence and timing constrained task graph on a set of homogeneous processors with communication time in a way that simultaneously minimizes the number of processors used f_1 and the total tardiness 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.
4. Only processing requirements are significant; memory, I/O, and other resource requirements are negligible.

The soft real-time task scheduling problem (sr-TSP) is formulated under the following assumptions: Computation time and deadline of each task and the communication time between processors are known. A time unit is artificial time unit. Soft real-time tasks scheduling problem is formulated as follows:

$$\min f_1 = M \tag{1}$$

$$\min f_2 = \sum_{i=1}^{N} \max\left\{0,\ t_i^S + c_i - d_i\right\} \tag{2}$$

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

$$t_i^E \ge t_j^E + c_j + t_{ji}^C, \quad \tau_j \in \text{pre}(\tau_i), \quad \forall i \tag{4}$$

$$1 \le M \le N \tag{5}$$

In above equations, notations are defined as follows:

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- Indices

i, j : task index, i, j=1,2,…,N

m : processor index, m=1,2,…,M

- Parameters

G=(T, E) : task graph

T={τ_1, τ_2, …, τ_N} : a set of N tasks

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

τ_i : ith task, i=1,2,…, N

p_m: mth processor, m=1,2,…,M

c_i : computation time of task τ_i

d_i : deadline of task τ_i

g_{ij} : communication time between task τ_i and task τ_j

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 ith task

$$t_i^E = \begin{cases} 0, & \text{if } \neg\exists \tau_j : (\tau_j, \tau_i) \in E \\ \max\limits_{\tau_j \in \text{pre}^*(\tau_i)} \{t_j^E + c_j + t_{ji}^C\}, & \text{otherwise} \end{cases}, \ \forall i$$

t_i^L : latest start time of ith task

$$t_i^L = \begin{cases} d_i - c_i, & \text{if } \neg\exists \tau_j : (\tau_i, \tau_j) \in E \\ \min\{\min\limits_{\tau_j \in \text{suc}^*(\tau_i)} \{t_j^L - c_i - t_{ij}^C\}, d_i - c_i\}, & \text{otherwise} \end{cases}, \ \forall i$$

- Decision Variables

t_i^S : real start time of ith task

$$t_{ij}^C = \begin{cases} 0, & \text{if task } \tau_i \text{ and task } \tau_j \text{ are assigned to same processor} \\ g_{ij}, & \text{otherwise} \end{cases}$$

M : total number of processors used

Equation (1) and (2) are the objective function in this scheduling problem. In (1) means to minimize the total number of processors used and (2) means to minimize total tardiness of tasks. Constraints conditions are shown from (3) to (5). Equation (3) means that task can be started after its earliest start time, begin its deadline. Equation (4) defines the earliest start time of task based on precedence constraints. Equation (5) is nonnegative condition for the number of processors.

III. Proposed GA

In this paper, solution algorithm is based on genetic algorithm (GA). Several new techniques are proposed in the encoding and decoding algorithm of genetic string. ush out any text that may try to fill in next to the graphic.

A. Encoding and Decoding

A chromosome V_k, k=1,2,…,$popSize$, 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 denote the processor to which task is allocated.

Encoding procedure for soft real-time task scheduling problem (sr-TSP) is composed of two strategies: strategy I for $u(\cdot)$ and strategy II for $v(\cdot)$. Procedures will be written as follows:

procedure: Encoding Strategy I for sr-TSP
input: task graph data set
output: $u(\cdot)$
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 ($w \neq \phi$)
    $j \leftarrow$ random(w);
    $u(l) \leftarrow j$;
    $l \leftarrow l+1$;
    $w \leftarrow w - \{j\}$;
    pre*(τ_i) $\leftarrow$ pre*(τ_i) $- \{\tau_j\}, \forall i$;
  end
  end
  output $u(\cdot)$;
end

procedure: Encoding Strategy II for sr-TSP
input: task graph data set, $u(\cdot)$,

$$M = \begin{cases} M(k-1), & \text{if } 1 < k \leq popSize \\ |\text{subgraph}|, & \text{if } k = 1 \end{cases}, \ \alpha, \beta$$

output: $v(\cdot)$, M_k
begin
  $l \leftarrow 1$, $t_m \leftarrow 0$, $\forall m$, $idle \leftarrow 0$;
  while ($l \leq N$)
    $m \leftarrow$ random[1, M];
    $i \leftarrow u(l)$;
    if ($t_m < t_i^E$) **then**
      $t_i^S \leftarrow t_i^E$;
      $idle \leftarrow idle + (t_i^S - t_m)$;
    end
    else $t_i^S \leftarrow t_m$;
    if ($t_i^S > t_i^L$) **then**
      if ($idle/c_i < \alpha$) **then**
      $M \leftarrow M+1$;
      $m \leftarrow M$;
      $idle \leftarrow idle + t_i^E$;
      $t_m \leftarrow t_i^E + c_i$;
      end
      else
      $idle \leftarrow \max\{0, (idle - c_i)\}$;
      end
    else
      $t_m \leftarrow t_i^S + c_i$;
      $v(l) \leftarrow m$;
      $l \leftarrow l+1$;
    end
    $idle \leftarrow idle + \sum(\max\{t_m\} - t_m)$;
  end
  while ($idle/\sum M \times \max\{t_m\} > \beta$)
    $M \leftarrow M - 1$;
    idle $\leftarrow$ idle- $idle/\sum M \times \max\{t_m\}$;
  end
  output $v(\cdot)$, M_k;
end

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The procedure of strategy I generate $u(\cdot)$ as satisfying precedence relationship between tasks.

In encoding strategy II procedure, the total number of processors is not fixed. Before generating $v(\cdot)$, the total number of processors M is defined and after generating $v(\cdot)$, M influence to $v(\cdot)$ of next chromosome. The M is changed during generating $v(\cdot)$. So, the M of output is different to the M of input. Where α, β is boundary constant to decide increasing the number of processor and decreasing the number of processor respectively.

Decoding procedure is will be written as follows:

procedure: Decoding for sr-TSP
input: task graph data set, chromosome $u(\cdot)$, $v(\cdot)$
output: schedule set S, the total number of processor used f_1, total tardiness of tasks f_2
begin
$l \leftarrow 1$, $t_m \leftarrow 0$, $\forall m$, $idle_m \leftarrow \phi$, $\forall m$, $f_1 \leftarrow 0$, $f_2 \leftarrow 0$,
$S \leftarrow \phi$,
while ($l \le N$)
$i \leftarrow u(l)$;
$m \leftarrow v(l)$;
if ($t_m = 0$) **then** $f_1 \leftarrow f_1 + 1$;
if (exist suitable idle time) **then** insert(i);
start(i);
update_idle();
$f_2 \leftarrow f_2 + \max\{0, (t_i^S + c_i - d_i)\}$;
$S \leftarrow S \cup \{(i, m: t_i^S - t_i^F)\}$;
$l \leftarrow l + 1$;
end
output S, f_1, f_2 ;
end

where insert(i) means to insert τ_i at idle time if τ_i is computable in idle time, start(i) means to assigne τ_i to maximum finish time of all assigned task to p_m, add_idle() means to add idle time to idle time list if idle time is occurred. I^S means the start time of idle duration, I^F means the end time of idle duration, $idel_m$ means the list of idle time and t_m means the maximum finish time of all assigned task to p_m.

The values of two objective f_1 and f_2 are calculated through the procedure of decoding.

We use one-cut crossover for GA paerator.

B. 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 [11].

In this paper, we combine Adaptive Weight Approach (AWA) [10] 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: $\boldsymbol{f}^-$)

$$\boldsymbol{f}^+ = \{f_1^{\max}, f_2^{\max}\} \tag{6}$$

$$\boldsymbol{f}^- = \{f_1^{\min}, f_2^{\min}\} \tag{7}$$

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

$$f_q^{\max} = \max_k \{f_q(V_k)\},\ q = 1,2 \tag{8}$$

$$f_q^{\min} = \min_k \{f_q(V_k)\},\ q = 1,2 \tag{9}$$

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

$$\begin{bmatrix} f_1^{\max} & f_2^{\min} \\ f_1^{\min} & f_2^{\max} \end{bmatrix} \tag{10}$$

Adaptive moving line defined by the extreme points ($f_1^{\max}$, $f_2^{\min}$) and ($f_1^{\min}$, $f_2^{\max}$) are shown Figure 1.

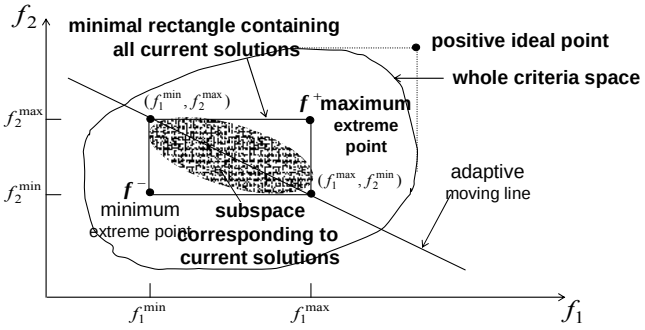

Fig. 1. Adaptive weights and adaptive hyper plane

The weighted-sum objective function for a given chromosome V_k is given by the following equation:

$$F(V_k) = \sum_{q=1}^{2} w_q f_q(V_k) = \sum_{q=1}^{2} \frac{f_q(V_k)}{f_q^{\max} - f_q^{\min}} \tag{11}$$

where w_q is adaptive weight for objective q :

$$w_q = \frac{1}{f_q^{\max} - f_q^{\min}},\quad q = 1,2 \tag{12}$$

The evaluation function is designed as follows:

$$eval(V_k) = 1 / F(V_k) = \frac{1}{\sum_{q=1}^{2} \frac{f_q(V_k)}{f_q^{\max} - f_q^{\min}}} \tag{13}$$

For selection, the commonly strategy called roulette wheel selection [9], [12] has been used.

C. GA operators

We use one-cut crossover. This operator creates two new chromosomes (the offspring) by mating two chromosomes (the parent). We crossover only part $v(\cdot)$ in chromosome to

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

maintain the topological order with respect to the given task graph that satisfies precedence relations.

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

D. Reproduction and Population Replacement

During reproduction and replacement steps, 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. Then new population is built through evaluating chromosomes and selecting.

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

procedure; sr-TSP by moGA
input: task graph data set
output: best schedule set *S*
begin
t ←0;
initialize *P*(*t*) by encoding strategy I and strategy II;
evaluate f_1, f_2 of *P*(*t*) by decoding routine;
create Pareto *E*(*P*);
fitness *eval*(*P*) by AWA;
while (**not** termination condition) **do**
crossover *P*(*t*) to yield *C*(*t*)
by one-cut crossover;
mutation *P*(*t*) to yield *C*(*t*)
by altering mutation;
evaluate f_1, f_2 by *C*(*t*) by decoding routine;
update Pareto *E*(*P*, *C*);
fitness *eval*(*P, C*) by AWA;
select *P*(*t* +1) from *P*(*t*) and *C*(*t*);
t ← *t* +1;
end
output best schedule set *S*;
end

Let *P*(*t*) and *C*(*t*) be parents and offspring in current generation *t*. Firstly, *P*(*t*) is initialized by encoding algorithm and each objective function is calculated. Then Pareto solution set is initialized by nondominated solution. The offspring is generated by crossover operation and mutation operation, and each objective function is calculated also. A new generation is formed by selecting, according to the fitness values, some of parents and offspring and rejecting others so as to keep the population size constant. The Pareto solution set is updated during these process.

IV. Validation

To validate proposed moGA (multiobjective GA), several numerical tests are performed. We compared proposed moGA with Oh-Wu's algorithm by Oh and Wu and Monnier-GA by Monnier et al [9]. Numerical tests are performed with randomly generated task graph. We use P-Method [13] for generation task graph. Numerical tests are performed with three task graph: the number of tasks 10, 50 and 100. Task graph and data set is omitted.

Table 1, 2 and 3 shows the comparisons of results by three different scheduling algorithms.

TABLE I
COMPARISON IN NO-TARDINESS FOR 10 TASKS

Terms	Monnier GA	Oh-Wu's algorithm	Proposed moGA
# of processor M	3	4	3
Makespan	27	29	28
CPU Times (*msec*)	14	13	19
Average utilization of processors	0.550725	0.487179	0.578571

TABLE II
COMPARISON IN NO-TARDINESS FOR 50 TASKS

Terms	Monnier GA	Oh-Wu's algorithm	Proposed moGA
# of processor M	22	19	16
Makespan	29	30	27
CPU Times (*msec*)	118	122	128
Average utilization of processors	0.368339	0.412281	0.543981

TABLE III
COMPARISON IN NO-TARDINESS FOR 100 TASKS

Terms	Monnier GA	Oh-Wu's algorithm	Proposed moGA
# of processor M	38	37	32
Makespan	149	157	163
CPU Times (*msec*)	497	511	518
Average utilization of processors	0.447582	0.453392	0.567352

In table 1, 2 and 3, there is no tardiness inclusively. The computing time of proposed moGA is a little bit longer than those of the other two. However, the number of used processors is fewer than those of the other two algorithms. The average utilization of processors by moGA is most desirable than those of the others.

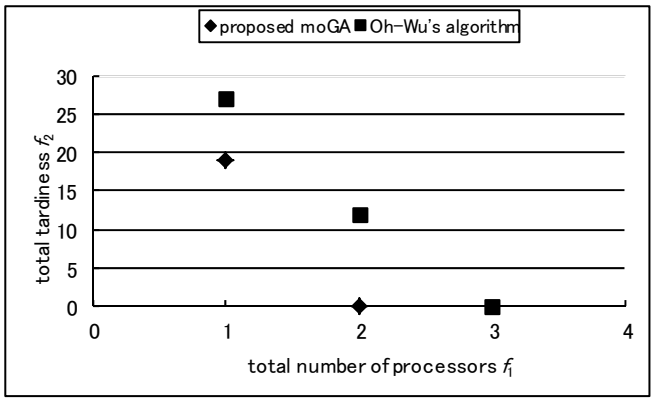

Fig. 2. Pareto solution in 10 task

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Figure 2, 3 and 4 represent the Pareto solution of proposed moGA and those of Oh-Wu's algorithm. In these Figure, the Pareto solution curve by proposed moGA is closer to ideal point than that of Oh-Wu's algorithm.

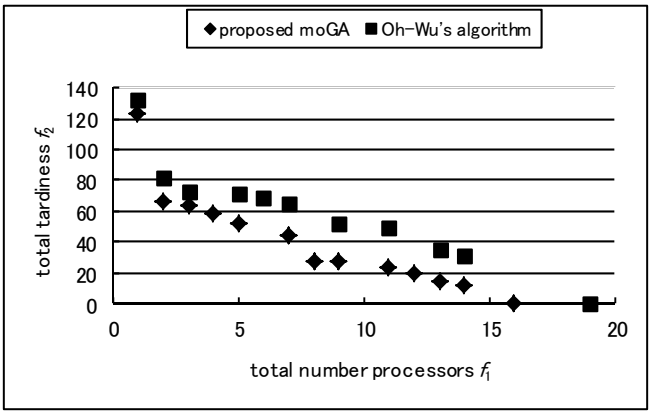

Fig. 3. Pareto solution in 50 tasks

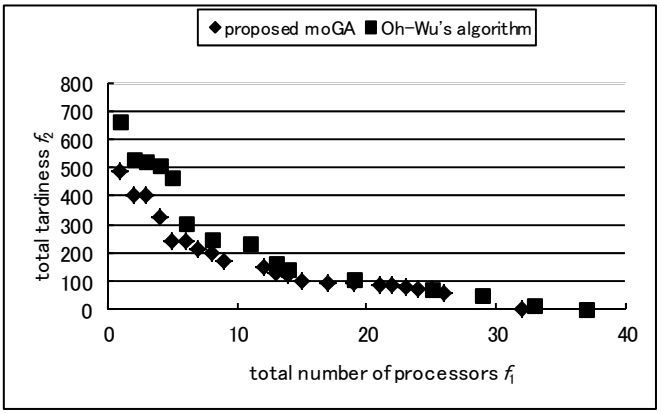

Fig. 4. Pareto solution in 100 tasks

V. Conclusions

A new task scheduling algorithm is proposed in this paper. This algorithm is designed for nonpreemptive tasks in soft real-time multiprocessor system with the communication time between processors and the precedence relationship between tasks. 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 Adaptive Weight Approach (AWA). 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.

References

[1] C. M. Krishna, and G. S. Kang, *Real-Time System*. McGraw-Hill 1997.
[2] J. L. Diaz, D. F. Garcia, and J. M. Lopez, "Minimum and Maximum Utilization Bounds for Multiprocessor Rate Monotonic Scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no.7, pp. 642-653, 2004.
[3] G. Bernat, A. Burns, and A. Liamosi., "Weakly Hard Real-Time Systems," *Transactions on Computer Systems*, vol. 50, no. 4, pp. 308-321, 2001.
[4] M. H. Kim, H. G. Lee, and J. W. Lee, "A Proportional-Share Scheduler for Multimedia Applications," *Proc. of Multimedia Computing and Systems*, pp. 484-491, 1997.
[5] M. R. Yoo, "A Scheduling Algorithm for Multimedia Process," *Ph. D. dissertation, University of YeoungNam*, *Korea*, 2002.
[6] F. Yalaoui, and 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, and P. Ramanathan, "A Genetic Approach for Scheduling Non-preemptive Tasks with Precedence and Deadline Constraints," *Proc. of the 26th Hawaii International Conference on System Sciences*, pp. 556-564, 1993.
[8] M. Lin, and L. Yang, "Hybrid Genetic Algorithms for Scheduling Partially Ordered Tasks in A Multi-processor Environment," *Proc. of the 6th International Conference on Real-Time Computer Systems and Applications*, pp. 382-387, 1999.
[9] Y. Monnier, J. P. Beauvais, and A. M. Deplanche, "A Genetic Algorithm for Scheduling Tasks in a Real-Time Distributed System," *Proc. of 24th Euromicro Conference*, pp. 708-714, 1998.
[10] M. Gen, and R. Cheng, *Genetic Algorithms & Engineering Optimization*, John Wiley & Sons, 2000.
[11] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.
[12] M. Gen, and R. Cheng, *Genetic Algorithms & Engineering Design*, John Wiley & Sons, 1997.
[13] S. Al-Sharaeh, and B. E. Wells, "A Comparison of Heuristics for List Schedules using The Box-method and P-method for Random Digraph Generation," *Proc. of the 28th Southeastern Symposium on System Theory*, pp. 467-471, 1996.

ISBN: 978-988-19253-8-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)