

# A PARALLEL SEARCHING SCHEME FOR MULTIPROCESSOR SYSTEMS AND ITS APPLICATION TO COMBINATORIAL PROBLEMS

Masaharu Imai  
School of Information Eng.  
Toyohashi University of  
Technology  
Toyohashi 440, Japan

Yuuji Yoshida  
Computation Center  
Nagoya University  
Nagoya 464, Japan

Teruo Fukumura  
Faculty of Engineering  
Nagoya University  
Nagoya 464, Japan

In this paper, we propose a parallelized computational scheme called *Parallelized Depth-First Algorithm* (PDFA) for Branch-and-Bound (B&B) method that works on multiprocessor systems. It is shown theoretically that the space requirement of PDFA on  $p$  processing units is at most  $p$  times as much as that of the sequential B&B algorithm with the depth-first search function. Moreover, from our experimental results through simulation, it is known that the computation time of PDFA on  $p$  processing units can be reduced to less than  $1/p$  that of the sequential B&B algorithm with the depth-first search function. We name this reduction effect in the computation time *Acceleration Effect*.

## 1. INTRODUCTION

Branch-and-Bound (B&B) method is one of the most general technique to solve combinatorial optimization problems; in particular, it is remarkable that B&B method is closely related to the problem solving methods used in Artificial Intelligence.[3]

In this paper, we propose a parallelized computation scheme called *Parallelized Depth-First Algorithm* (PDFA) for B&B method that works on multiprocessor systems. Next, we show that the space requirement of PDFA on  $p$  processing units is at most  $p$  times as much as that of the sequential B&B algorithm with the depth-first search function. And, we also show that the computation time of PDFA on  $p$  processing units can be reduced to less than  $1/p$  that of the sequential B&B algorithm using the depth-first search function.

## 2. DEFINITIONS AND NOTATIONS

The multiprocessor system on which PDFA is assumed to be implemented consists of the following components (see Fig. 1):

- (i)  $p$  Processing Units (PUs) with Private Memory Unit (PMU) each;
  - (ii) A shared Common Memory Unit (CMU).
- In this model, PMU# $i$  is restricted to be accessible only by PU# $i$ , for each  $i$ ,  $i = 1, 2, \dots, p$ . On the other hand, CMU is accessible by any PU. Moreover, we also assume:
- (i) Each PU computes independently of other PUs;
  - (ii) The communication between PUs is carried out only through CMU.

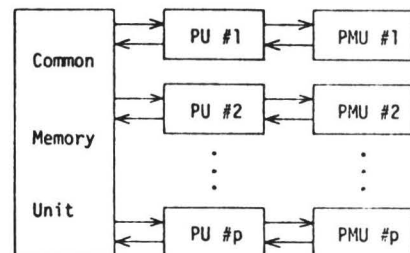


Fig. 1 Multiprocessor System

PU: Processing Unit  
PMU: Private Memory Unit

In the followings, let  $P_0$  denote a minimization problem with  $n$   $k$ -valued variables. The goal of the problem is to find out one of the optimum solutions of  $P_0$  and its objective value. Let  $I$  be the current set of open subproblems generated by a B&B algorithm. For each subproblem  $P_i$  in  $I$ , we define the functions  $cost$ ,  $lb$ ,  $h$  and  $depth$  as follows:

- (i) Let  $cost(P_i)$  be the optimal value of the objective function for  $P_i$ .
- (ii) Since the actual value of  $cost(P_i)$  cannot be known before  $P_i$  has been completely solved, we use  $lb(P_i)$  as its lower bound under the assumption that  $lb(P_i)$  should be less than or equal to  $cost(P_i)$ , for all  $P_i$  in  $I$ .
- (iii) When some additional information on  $cost(P_i)$  is available, we can construct a more powerful estimation than  $lb$ . We denote it  $h(P_i)$ . Without loss of generality, we assume  $h(P_i) \neq h(P_j)$  if  $i \neq j$ . Function  $h$  is usually called *heuristic function*.

- (ix) Finally, let  $depth(P_i)$  denote the number of variables whose values are already assigned in  $P_i$ .

There are several kind of search functions used in B&B method. One of the most popularly used search functions is the depth-first search function  $s_D$ . It selects the subproblem  $P_i$  that has the smallest  $h$  value among those with the largest depth in  $I$ .

### 3. PARALLELIZED DEPTH-FIRST ALGORITHM

In PDFA described below, following variables are stored in CMU for common use:

- $I$ : The set of open subproblems generated through PDFA;  
 $X_{opt}$ : The currently best solution (a candidate of the optimal solution);  
 $C_{opt}$ : The cost of  $X_{opt}$ , i.e.,  $cost(X_{opt})$ ;  
 $act$ : The number of "active" PUs, where "active" means that the PU is processing a subproblem.

All accesses to any of these variables should be controlled under mutual exclusion.

PDFA is described as follows:

- Phase 1: Set  $I \leftarrow \{P_0\}$ ,  $X_{opt} \leftarrow \phi$ , and  $act \leftarrow 0$ .  
Phase 2: Do Procedure P in parallel on each PU.  
 Algorithm terminates when no PU is active.

Procedure P is described as follows:

- Step 1: If  $I = \phi$  and  $act = 0$  then halt.  
 Otherwise, if  $I = \phi$  and  $act > 0$  then repeat this step, else go to Step 2.  
Step 2: Set  $act \leftarrow act + 1$ ,  $P_i \leftarrow s_D(I)$ , and  $I \leftarrow I - \{P_i\}$ .  
Step 3: If  $P_i$  is terminated then go to Step 5.  
Step 4: If  $lb(P_i) < C_{opt}$  then decompose  $P_i$  into  $k$  subproblems and add them to  $I$ .  
 Go to Step 6.  
Step 5: If  $cost(P_i) < C_{opt}$  then set  $X_{opt} \leftarrow P_i$  and  $C_{opt} \leftarrow cost(P_i)$ .  
Step 6: Set  $act \leftarrow act - 1$  and go to Step 1.

In the followings,  $p$ -PDFA denotes PDFA with  $p$  PUs. It is obvious that 1-PDFA is identical to the sequential B&B algorithm using the depth-first search function.

### 4. SOME PROPERTIES OF PDFA

In this section, we discuss the space- and time-requirements of PDFA.

#### 4.1 SPACE REQUIREMENT

Let  $M_p$  be the largest value of all  $|I|$ 's obtained during the computation of  $p$ -PDFA to solve  $P_0$ .

Then, the following theorem holds.

Theorem 1:  $M_p \leq p \times \{(k-1) \times n + 1\}$

According to Theorem 1, the memory space needed to implement  $p$ -PDFA is only  $O(p \times k \times n)$ , which is just  $p$  times that of the sequential B&B algorithm using the depth-first search function. It should be also noted here that the space requirement of the sequential B&B algorithm using any other search functions such as the best-bound, the breadth-first, or the heuristic are  $O(k^n)$ .

#### 4.2 TIME REQUIREMENT

Let  $K$  be the number of subproblems processed by  $p$ -PDFA to solve  $P_0$ . It is reasonable to employ  $K$  to estimate the time requirement of  $p$ -PDFA. And, let  $T$  be the computation time needed to solve  $P_0$  by  $p$ -PDFA. We assume that  $T$  satisfies  $T_p = c \times K_p / p$ , for some constant  $c$ .

One of the most significant results to be mentioned in this paper is:

$T_p$  can be reduced to less than  $T_1/p$ , if  $p$  is properly selected.

The reason is briefly stated as follows:

- (i) Let  $C$  be the cost of the first feasible solution of  $P_0$  found by  $p$ -PDFA. Then, it is obvious that as  $p$  increases,  $C$  converges to the optimal objective value of  $P_0$ .
- (ii) The sooner a good feasible solution is found, the lesser  $K_p$  becomes. And accordingly,  $T_p$  will become less than  $T_1/p$ .
- (iii) But if  $p$  increases too much,  $K_p$  will also increase, since  $p$ -PDFA may process unnecessary subproblems to find an optimal solution.

We name this reduction effect in the computation time of  $p$ -PDFA compared to  $T_1$ , *Acceleration Effect*.

### 5. EXPERIMENT

Simulation experiment of  $p$ -PDFA was performed for a model of randomly generated 0-1 combinatorial optimization problems.[1] The purpose of this experiment is to know how *Acceleration Effect* appears in PDFA. In this simulation, we assumed that every subproblem can be processed in uniform time, for simplicity.

The problems were generated as follows. First, we set:

- (i)  $cost(P_0) = 1.0$  and  $lb(P_0) = 0.0$ .  
 And the following values were randomly generated:
- (ii) Let  $P_j$  and  $P_k$  be two different sons of  $P_i$ .

Then, we set:  $\text{cost}(P_j) = \text{cost}(P_i)$  and  $\text{cost}(P_k) = \text{cost}(P_i) + \alpha$ , where  $\alpha$  is a non-negative random number taken from the exponential distribution,  $\lambda \exp(-\lambda \alpha)$  with  $\lambda = 0.2$ .

- (iii) Let  $P_j$  be a son of  $P_i$ . Then,  $\text{lb}(P_j)$  was taken from uniform random numbers which distribute between  $[\text{lb}(P_i), \text{lb}(P_i) + 2 \times (\text{cost}(P_j) - \text{lb}(P_i)) / (n - \text{depth}(P_i))]$ .
- (iv) Let  $\sigma$  be a parameter which denotes the accuracy of the heuristic function  $h$ . Then,  $h(P_i)$  was taken from the normal distribution,  $N(\text{cost}(P_i), ((n - \text{depth}(P_i)) \times \sigma)^2)$ . Thus, a small  $\sigma$  implies an accurate  $h$ .

The results are summarized in Fig. 2, in which  $n = 30$ . Each point in this figure represents the average of 20 different test runs. And  $\max(K^*, (n+1) \times p)$  (shown by the dot-dash-line) denotes a theoretical lower bound of  $K_p$ . Where  $K^*$  is the number of subproblems whose  $\text{lb}$  values are less than  $\text{cost}(P_0)$ .

From this figure, we know:

- (i)  $K$  can be reduced to less than  $K_1$ , if  $p$  is properly selected. Thus, *Acceleration Effect* appears.
- (ii) *Acceleration Effect* appears more drastically when poorer heuristics are used.

## 6. CONCLUSION

PDFA is an attractive parallel algorithm for multiprocessor systems because of its small space requirement and especially of *Acceleration Effect*. *Acceleration Effect* appears through the parallelized B&B algorithm using the depth-first or the heuristic search function.

A similar effect as *Acceleration Effect* was also observed in the experiments on the  $\text{Cm}^*$  multiprocessor system, which solved integer programming problems by B&B method.[2]

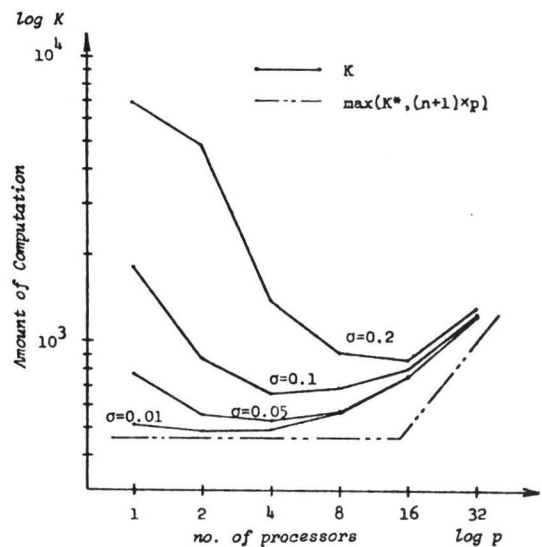


Fig. 2 Amount of Computation of PDFA

## ACKNOWLEDGEMENT

Authors would like to thank Prof. N. Honda of Nagoya University for his helpful suggestions and discussions.

## REFERENCES

- [1] Ibaraki, T. "Theoretical Consideration on Computational Performance of Branch-and-Bound Algorithms" presented at IX Int. Symp. on Math. Prog. (1976).
- [2] Fuller, S.H. et al. "Multi-Microprocessors: An Overview and Working Example" In Proc. of IEEE 66:2 (1978) 216-228.
- [3] Nilsson, N.J. Problem-Solving Methods in Artificial Intelligence. Mcgraw-Hill (1971).
- [4] Imai, M. et al. "A Parallelized Branch-and-Bound Algorithm - Implementation and Efficiency -" Trans. IECE of Japan 62D:6 to be appeared.