Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080425

A Parallel Branch and Bound Algorithm for Solving Large Scale Integer Programming Problems

Mahmoud M. Ismail[1], *Osama abd el-raoof*[2,*] *and Waiel F. Abd EL-Wahed*[2]

[1] Operations Research Department, Zagazig University, EL-Sharkia, Egypt
[2] Operations Research and DSS Department, Menofia University, Shebien El-koum, Egypt

Received: 26 Jul. 2013, Revised: 28 Oct. 2013, Accepted: 29 Oct. 2013
Published online: 1 Jul. 2014

Abstract: Branch and Bound technique is commonly used for intelligent search in finding a set of integer solutions within a space of interest. The corresponding binary tree structure provides a natural parallelism allowing concurrent evaluation of subproblems using parallel computing technology. While the master-worker paradigm is successfully used in many parallel applications as a common framework to implement parallel applications, it has drawbacks when a large number of computing resources are connected via WAN. A supervisor-master-sub-master-worker algorithm has been proposed. From the solved benchmark example this algorithm proved to provide a considerable save of time. Results show that a consistently better efficiency can be achieved in solving integer equations, providing reduction of time. The hierarchical supervisor-master-sub-master-worker algorithm sustains good performance revealed from the knapsack problem solved as a benchmark example.

Keywords: Branch and bound, parallel processing, and Integer Programming

1 Introduction

Combinatorial Optimization is the process of finding one or more of best solutions in a well-defined discrete problem space [24]. Such problems occur in almost all fields of management (e.g. finance, marketing, production, scheduling, inventory control, facility location and layout, data-base management), as well as in many engineering disciplines. Branch-and-bound algorithms are general methods applicable to various combinatorial optimization problems that belong to the class of NP-hard problems [4]. These algorithms are search-based techniques that enumerate the entire solution space. Parallelization is an appropriate method for accelerating the enumeration process. Since the algorithms is usually time consuming in the evaluation of a subproblem, high-level parallelism of such algorithms is implemented, in which case all the existing subproblems are parallelized simultaneously provided that an adequate number of processors is available. Even though there are several criteria for classifying parallel branch-and-bound algorithms [5], [7], [9] and [19], the most useful criterion is the search tree management. The search tree is managed with a single subproblem pool in the central control case (central control scheme) or with multiple subproblem pools in the distributed control case (distributed control scheme). Parallelization with a single subproblem pool usually achieves higher efficiency until a large number of processors are used [11] and [20].

Parallel Branch and Bound algorithm for Integer programming has been studied since the early eighties [2], [8], [16], [17], [22], [23] and [25]. Parallel computers in general and distributed multiprocessor computers in particular are increasingly accepted as platforms which provide enhanced computational power in a cost effective way. Developing parallel search techniques for these platforms has two alternative motivations, namely, speeding up the solution time for a given model and to increase the size of the solvable IP problems; these are known as speed-up and scale-up in the parallel algorithm literature. In developing Parallel Branch and Bound (PB&B) algorithms, the broad aim is to reduce the execution time in relation to the number of processors used and to solve large size problems.

The paper is organized such that the next section 2 provides a brief overview of integer programming. Section 3 describes the basics of Parallel Branch and Bound algorithm. Section 4 describes the method of the

* Corresponding author e-mail: osamaabd@hotmail.com

proposed parallel branch and bound algorithm used. Section 5 discusses Solving Integer Programming Problem by the Proposed Parallel Branch and Bound Algorithm. Section 6 discusses the computational results. In section 7, a conclusion is introduced.

2 Integer Programming

It is often impossible to represent certain features of many real-world problems using only linear constraints and continuous variables. In modeling a real world problem, it is often necessary to represent discrete activities by variables which are restricted to take only integer values.

The general mathematical form of integer programming problems is:

Maximize

$$Z = \sum_{i=1}^{n} c_i x_i. \tag{1}$$

Subject to.

$$\sum_{i=1}^{n} a_{ji}\, x_i \leq b_j \ , (j = 1,2,3,...,m), \tag{2}$$

$$x_i \geq 0,\ x_i \ integers \ and \ (i = 1,2,3,...,n). \tag{3}$$

Such problems are called linear integer-programming problems. It is said to be mixed integer program when some, but not all, variables are restricted to be integer, and is called a pure integer program when all decision variables are integers. If the constraints are of network nature, then an integer solution can be obtained by ignoring the integrality restrictions and solving the resulting linear program. Otherwise, variables will be fractional in the linear-programming solution, and further measures must be taken to determine the integer-programming solution.

Note, that MIPs in maximization form can be transformed to minimization form by multiplying the objective function vector by -1. Similarly, "$\geq$" constraints can be multiplied by -1 to obtain "$\leq$" constraints. Equations can be replaced by two opposite inequalities [1].

We consider the 0-1 integer programming problem:

Maximize

$$Z = \sum_{i=1}^{n} c_i x_i. \tag{4}$$

Subject to.

$$\sum_{i=1}^{n} a_{ji}\, x_i \leq b_j \ , (j = 1,2,3,...,m), \tag{5}$$

$$x_j = 0 \ or \ 1 \ , (i = 1,2,3,...,n). \tag{6}$$

We will restrict our attention to the case where a is 0-1, and where b is integer. Also with these restrictions, the problem is NP-hard, and several well known NP-hard problems, such as the set partitioning, covering and packing problems, are conveniently stated in this way. Common solution methods for ILP are based on solving LP, which can usually be done efficiently. If LP happens to give a 0-1 solution, this is also an optimal solution to ILP, and for certain common and nontrivial subclasses of 0-1 problems, LP always have an integer solution. For more difficult problems particular instances may also be easy in this sense. If however the LP solution has many noninteger values, very little information about the solution to the 0-1 problem is obtained in this way, and typically techniques such as branch and bound have to be used to resolve the solution to integrality. This can work very well for small problems and also for larger problems with special structure, but nevertheless strongly limits the range and size of problems that can be solved. For a full presentation of existing methods, see for example [10] and [21].

Branch and bound method is the basic workhorse technique for solving integer and discrete programming problems. The method is based on the observation that the enumeration of integer solutions has a tree structure. The main idea of the branch and bound algorithm is to find an optimal solution and to prove its optimality by successively partitioning the feasible set of the solution, or the original problem, into subproblems of smaller size. These subproblems are investigated by computing lower/upper bounds of the objective function. These lower/upper bounds are used to avoid exhaustive search of the solution space.

3 Parallel Branch and Bound

The branch and bound is the divide and conquer method. We divide a large problem into a few smaller ones. (This is the "branch" part). The conquering part is done by estimate how good a solution we can get for each smaller problems (to do this, we may have to divide the problem further, until we get a problem that we can handle), that is the "bound" part [3].

The branch and bound algorithm is able to be parallelized by distributing computation of subproblems on multiple computing nodes. Parallel branch and bound algorithms with the master-worker algorithm, where a single master process dispatches tasks to multiple worker processes, have been proposed in many literatures [3], [18]. In master-worker algorithm, a single master process dispatches subproblems, which correspond to leaf nodes on the search tree, to multiple worker processes and receives the computed results from the worker processes. The computed results contain the best upper bound of the objective function, and subproblems that have generated by branching and have not been pruned on a worker process. Also, the parallel algorithm with the hierarchical master-worker paradigm is proposed to improve

performance on large-scale computing environment [13] and [28].

3.1 Previous work

Carrying out a literature revealed that the master-worker used in large scale size problems suffers from communication overhead, and bottleneck on a Single Master Process.

3.1.1 Communication overhead

Communication overhead between a master process and worker processes affects the performance of an application significantly. Communication occurs when a master process dispatches a task to a worker process and a worker process returns computed results to a master process. Performance degradation occurs when communication overhead is relatively large compared with execution time of a single task. Thus, the impact of communication overhead on performance could be significant. In the supervisor-master-sub-master-worker algorithm, the traffics between supervisor-masters processes, master-submasters processes, and submaster-worker processes are reduced.

3.1.2 Bottleneck on a Single Master Process

The performance of a master process could be a bottleneck of application performance if the master process controls too many worker processes. A master process continuously communicates with all worker processes to find an idle worker process, to dispatch new tasks and to receive computed results. A master process needs to perform these procedures in very frequent manner. Thus, if a master process controls too many worker processes (in large scale problems), procedures for computation and I/O on a master process degrades performance.

4 Proposed Parallel Branch and Bound

This section describes the proposed parallel branch and bound algorithm to solve the large-scale integer programming Problems with the hierarchical supervisor-master-sub-master-worker paradigm, where a supervisor process controls multiple process sets, each of which is composed of a master processes. Each master composed of sub-master processes and each sub-master composed of worker processes. A set of master-sub-master-worker processes performs a parallel branch and bound method for a subset of a search tree, that is, a master process dispatches subproblems to multiple sub-masters, each sub-master process dispatches subproblems to worker processes and receives computed results from the worker processes. A supervisor process performs load balancing among master processes by delivering subproblems to master processes. A Master process performs load balancing among sub-master processes by delivering subproblems to sub-master processes. Also, a supervisor process, master processes, and sub-master processes gather the best upper bound of the objective function, which is computed on each worker process, and updates the current best upper bound on all worker processes hierarchically.

The updating of the current best upper bound improves the performance of the application. Figure 4.1 shows an overview of the proposed algorithm. On the figure, Z_{Wk}, Z_{Mi}, Z_{SMj}, and Z denote the current best upper bound of the objective function stored on a worker process W_k, a master process M_i, a sub-master process S_{Mj} and a supervisor process, respectively. Here, k = 1,2 ... is the number of worker processes in a set of a master and worker processes and i = 1,2 ... is the number of master processes, and j = 1,2 ... is the number of sub-master processes.

The supervisor, master, sub-master, and worker processes in the proposed algorithm are described in details in the rest of this section.

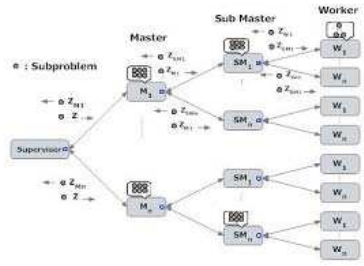

Fig. 1: Proposed Hierarchical Supervisor-Master-Sub master-Worker Paradigm

4.1 Supervisor Process

A supervisor process achieves load balancing; distribute the equal number of subproblems to master processes, and shares the best upper bound of the objective function among all processes by performing the following steps:

1. A supervisor process receives the computed results from the master process M_i.
2. A supervisor process compares Z_{Mi} with the current best upper bound Z. If Z_{Mi} is less than Z a supervisor process updates Z to the value of Z_{Mi} and requests all master processes to update the current best upper bound on the master processes with the updated Z.

3.A supervisor process computes the lowest lower bound of the objective function by comparing the lowest lower bounds computed on master processes, and examines the optimality of the problem, if the problem is feasible and optimal, a supervisor process requests all master processes to terminate computation.

4.2 Master Process

A master process, M_i, achieves load balancing; distribute the equal number of subproblems to sub-master processes. It repeats the following steps until it receives a request from a supervisor process to terminate the computation.

1.M_i receives new subproblems and the current best upper bound of the objective function stored on supervisor, or Z, from supervisor.
2.M_i compares the best upper bound stored on the supervisor process Z, with the current Z_{Mi}. If Z is less than Z_{Mi}, M_i updates Z_{Mi} to the value of Z.
3.M_i sends the results to a supervisor process containing the number of subproblems assigned to M_i, Z_{Mi} the current lowest lower bound of the objective function, and the solution of the objective function.
4.M_i searches for idle sub-master processes. If M_i finds an idle sub-master process, S_{Mj}, it performs the following steps:
   – M_i receives computed results containing subproblems generated on S_{Mj}, Z_{SMj} and the solution of the objective function, from S_{Mj}.
   – M_i prunes subproblems which their lower bounds exceed Z_{Mi}.
   – M_i compares the current Z_{Mi} and Z_{SMj}. If Z_{SMj} is less than Z_{Mi}, S_{Mj} updates Z_{Mi} to the value of Z_{SMj}.
   – M_i dispatches a new subproblem and sends Z_{Mi} to S_{Mj}.

4.3 Sub-Master Process

A sub-master process performs a parallel branch and bound method with worker processes and achieves load balancing in cooperation with a master process. A sub-master process, SMj, repeats the following steps until it receives a request to terminate the computation from a master process.

1.S_{Mj} receives new subproblems and the current best upper bound of the objective function stored on M_i, or Z_{Mi}, from M_i.
2.S_{Mj} compares the best upper bound stored on the master process, or Z_{Mi}, with the current Z_{SMj}. If Z_{Mi} is less than Z_{SMj}, S_{Mj} updates Z_{SMj} to the value of Z_{Mi}.
3.S_{Mj} sends the results to a master process. The results contain the number of subproblems assigned to S_{Mj}, Z_{SMj} the current lowest lower bound of the objective function, and the solution of the objective function.
4.S_{Mj} searches for idle worker processes. If S_{Mj} finds an idle worker process, W_k, it performs the following steps:
   – S_{Mj} receives computed results, which contains subproblems generated on W_k, Z_{Wk} and the solution of the objective function from W_k.
   – S_{Mj} prunes subproblems which their lower bounds exceed Z_{SMj}.
   – S_{Mj} compares the current Z_{SMj} and Z_{Wk}. If Z_{Wk} is less than Z_{SMj}, S_{Mj} updates Z_{SMj} to the value of Z_{Wk}.
   – S_{Mj} dispatches a new subproblem and sends Z_{SMj} to W_k.

4.4 Worker Process

A worker process, W_k, performs the following steps whenever it is dispatched a subproblem from a sub-master process, S_{Mj}.

1.W_k receives a subproblem and the current best upper bound of the objective function stored on S_{Mj}, or Z_{SMj}, from S_{Mj}.
2.W_k applies branch and bound technique on the subproblem and generates a tree of subproblems.
3.W_k computes the lower/upper bound of the objective function for subproblems on the tree. For each subproblem on the tree, if the computed upper bound is less than the current best upper bound stored on W_k, or Z_{Wk}, W_k updates Z_{Wk} to the lower value.
4.W_k prunes subproblems which their lower bounds exceed Z_{Wk}.
5.W_k returns the computed results, or subproblems that have not been pruned, Z_{Wk} and the solution of the objective function to S_{Mj}.

4.5 Waiting nodes organizing

Each worker keeps a local list of waiting nodes on the basis of node evaluation criterion. The list of waiting nodes is the most important item, which is requested by the sub-master. Moreover, the nodes which are higher up in the sorted list are more likely to be transmitted first. Therefore, a procedure has been set up to keep the list of waiting nodes sorted considering their priority as determined by the given criterion. Whenever a worker produces two new nodes or receives a number of nodes from the sub-master, this procedure is called to place these nodes. The nodes are then stored according to the given priority in the local list. The following is a summary of the procedure by which the list of waiting nodes is organized.

–Sorting the list: If there are new nodes to add to the local list then insert them into the list of waiting nodes in their proper priority, according to the strategy chosen.
–Pruning nodes:
-If the list is empty then Exit.
-If the node satisfies the pruned conditions then prune this node.
-Otherwise: Pointing to the next node to be solved.
–Exit.

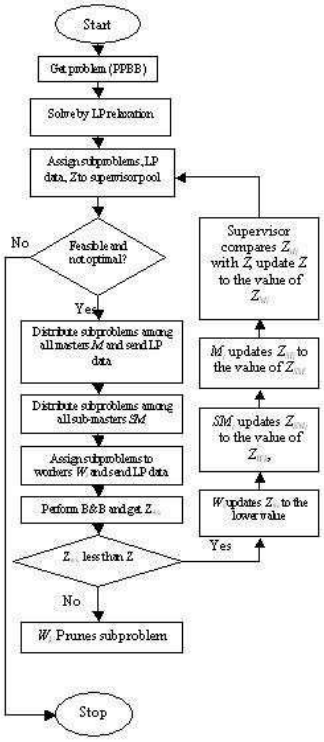

Fig. 2: Flowchart for proposed Parallel B&B

5 Solving Integer Programming Problem by the Proposed Parallel Branch and Bound Algorithm

benchmark tests the algorithm and compares results in many papers, so let us solve the following example by the proposed algorithm, Assuming that we have two masters.

Max

$$Z = 2x_1 - x_2 + 4x_3 + 7x_4 - 5x_5 + 12x_6 + 9x_7 - 4x_8 - x_9 + 2x_{10} \quad (7)$$

S.t.

$$3x_1 - x_2 + 2x_3 + 4x_6 - 3x_7 + 8x_8 + x_9 \geq 5 \quad (8)$$

$$x_1 - x_2 + 3x_4 + 7x_5 + 8x_6 + 5x_7 - x_8 - 7x_9 + 4x_10 \geq 12 \quad (9)$$

$$4x_2 + 7x_3 + x_4 + 2x_5 - 5x_6 + 3x_9 + 9x_10 \geq 1 \quad (10)$$

$$2x_1 + 4x_3 - x_4 + 4x_8 + 5x_9 + 3x_10 \geq 2 \quad (11)$$

$$x_j \text{ are integers } for\ j = 1, 2, 3, ..., 10 \quad (12)$$

The first step is to solve the problem using LP-relaxation and put the solution into the supervisor list, the initial solution is shown in Table 5.1.

Table 1: Initial solution using LP-relaxation

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
36.55	0	0	0	3.302	0	1.097	0.149	0	1.06	0

That is, the optimal value of the LP-relaxation is an upper bound z = 36.5513, and there are four non integer variables in the initial solution x_4, x_6, x_7, and x_9, the supervisor divides these variables among the masters using the loading balance. Let's deal with the non integer variables in the initial solution, supervisor sends variables x_7 and x_9 to 1^{st} master (M_1), and sends x_4 and x_6 to 2^{nd} master (M_2). M_1 divides its variables among sub-masters by sending x_9 to 1^{st} sub-master (S_{M1}), and sending x_7 to the 2^{nd} sub-master (S_{M2}). M_2 also divides its variables among sub-masters by sending x_6 to 1^{st} sub-master in M_2, and sending x_4 to the 2^{nd} sub-master in M_2.

S_{M1} in M_1 assigns the variable x_9 to the 1^{st} worker (W_1) which uses the branch and bound technique for the non integer variable x_9. The solution that is gained is shown in Table 5.2.

Table 2: Solution of B&B on x_9 variable

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
35.97	0	0	0	3.337	0	1.067	0.089	0	1	0

W_1 delivers the solution to S_{M1} in M_1 to add the new non integer variables x_4, x_6, and x_7 to its list, and then S_{M1} dispatches them to the workers by assigning x_4 to W_1, x_6 to W_2, and x_7 to W_3 to be solved, then the workers deliver the solutions to S_{M1} in M_1 to compare them and choose the best solution among them. At the same time, S_{M2} in M_1, S_{M1} in M_2, and S_{M2} in M_2 do the same steps on the non integer variables that have been assigned to each one until we obtain the integer solution. The first integer solution has been obtained in W_3 in S_{M1} in M_1 at the iteration number three, and it is shown in Table 5.3.

Table 3: First integer solution obtained

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
32	0	0	0	3	0	1	0	0	1	0

S_{M1} in M_1 compares the solutions of its workers and find that the best solution is the integer solution at W_3, and then it delivers this solution to M_1 which compares the solutions of S_{M1} and S_{M2} to find the best of them. Then it delivers the solution to the supervisor which compares the solutions of M_1 and M_2 to find that the integer solution at M_1 is the best solution. Supervisor broadcasts the best solution to masters, sub-masters, and workers to terminate the solution. The final (optimal) solution of this problem that has been broadcast by the supervisor is shown in Table 5.4.

Table 4: Final solution of the problem

z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
32	0	0	0	3	0	1	0	0	1	0

6 Computational Results

This section presents computational results of the proposed parallel branch and bound algorithm to solve Large-scale Knapsack Problems with the hierarchical supervisor-master-sub-master-worker paradigm. The computational results show performance comparison between the master-worker algorithm (MW) and the hierarchical supervisor- master-sub-master-worker algorithm (PPBB). Table 6.5, and Figure 6.3 present execution time to solve knapsack problems by MW and PPBB. A supervisor process, a master process, sub-master process and worker processes run on computers (PIV 3.0GHz, 1024MB RAM.) connected to LAN.

Table 5: Execution time of different instances of knapsack problems.

No. of Variables	PPBB		MW	
	Z	Time (ms)	Z	Time (ms)
10	241	111	241	362
50	569	826	569	1558
100	460	1542	460	7188
500	965	6284	965	26115
1000	4266	10109	4266	73645
2000	10242	18127	10242	97518

Computed results show that in small problems; with 10, and 50 variables, the execution times were approximately equal in both algorithms (PPBB, and MW). However moving to largest scale problems, the execution time of PPBB was obviously smaller than that in MW. It may be included from the results that as the number of variables , increases the difference between execution time of the two algorithms will assure the better performance of the PPBB algorithm with a clear reduction of execution time dealing with large-scale problems.

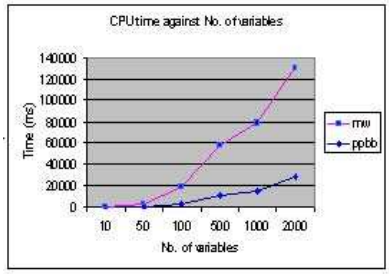

Fig. 3: Execution time of different instances of knapsack problems

From the previous results we can conclude from the authors view, the efficiency improvement between PPBB and MW as shown in Eq. (13).

$$\frac{T_k}{T_j} \times 100\%. \tag{13}$$

Where the T_k is the execution time of PPBB method, and T_j is the execution time of MW method. The results are shown in table 6.6.

Table 6: Efficiency Improvement in Time between PPBB and MW.

No. of variables	Efficiency Improvement in Time
10	326.1%
50	188.6%
100	466.1%
500	415.8%
1000	728.5%
2000	537.9%

It appears that the efficiency in time of PPBB is much greater than MW. Also, It appears that in large problem sizes the time efficiency in PPBB gets better than MW.

Conclusion

This paper proposed a parallel branch and bound algorithm to solve large-scale integer programming problems that parallelized with the hierarchical supervisor-master-sub-master-worker algorithm, and

compared its performance with conventional master-worker algorithm. The algorithm effectively overcomes the drawbacks of master-worker algorithm, such as communication overhead by putting frequent communication between a master process, a sub-master processes, and worker processes in tightly coupled computing resources. It also, overcomes bottleneck on a single master process by distributing work among multiple master processes. The algorithm also, increases the efficiency of the solution process, improves the performance scalability by distributing work among multiple master processes, and increases the diversification of solutions at the same time reducing the execution time in comparison with master-worker algorithm. The algorithm has been tested by solving a set of large-scale knapsack problems. The proposed algorithm is capable to provide a considerable reduction of time compared with other algorithms most obviously at lower scale problems.

References

[1] A. Fgenschuh and A. Martin. Computational integer programming and cutting planes. In K. Aardal, G. L. Nemhauser, and R.Weismantel, editors, Discrete Optimization, *Handbooks in Operations Research and Management Science*, **12**, 69–122 (2005).

[2] B. W. Wah, G. J. Li, C. F. Yu, Multiprocessing of Combinatorial Search Problems, *IEEE Comp.*, 93-108 (1985).

[3] Epperly, T. G. W, *Global Optimization of Nonconvex Nonlinear Programs Using Parallel Branch and Bound* , Ph.D. thesis. University of Wisconsin, Madison, WI, (1995).

[4] G. Cybenko, Dynamic load balancing for distributed memory multiprocessors, *Jornal of Parallel and Distributed Computing*, **7**, 279–301 (1989).

[5] G. Gendron and T. G. Crainic, Parallel branch-and-bound algorithms: survey and synthesis, *Operations Research*, **6**, 1042–1066 (1994).

[6] G. Mitra et al, *Parallel Computing*, **23**, 733–753 (1997).

[7] G. P. McKeown, V. J. Rayward-Smith, and S. A. Rush. Parallel branch-and-bound, *Parallel branch-and-bound*, Advanced topics in computer science, 111-150 (1992).

[8] G. P. McKeown, V. J. Rayward-Smith, S. A. Rush, H. J. Turpin, Using transputer network to solve B&B problems, *Proceedings of the Transputing 91 Conference*, **2**, 781–800 (1991).

[9] H. W. J. M. Trienekens, *Parallel Branch and Bound Algorithms* , Ph.D. thesis. Erasmus Universiteit, Rotterdam, (1990).

[10] Hu T. C., *Integer programming and network flows* , Addison Wesley, (1969).

[11] J. Eckstein. Control strategies for parallel branch-and-bound, *In Proceedings of Super-computing*, **94**, 41–48 (1994).

[12] J. Goux, S. Kulkarni, J. Linderoth, and M. Yoder, An enabling framework for master-worker applications on the computational grid, *IEEE Symposium on High Performance Distributed Computing*, **9**, (1994).

[13] K. Aida, W. Natsume and Y. Futakata, Distributed Computing with Hierarchical Master-worker Paradigm for Parallel Branch and Bound Algorithm, *IEEE/ACM International Symposium on Cluster Computing and the Grid*, **3**, (2003).

[14] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C Lee and H. Casanova, Overview of GridRPC, *A Remote Procedure Call API for Grid Computing, Grid Computing Grid 2002, LNCS2536*, (2002).

[15] M. O. Neary and P. Cappello, Advanced Eager Scheduling for Java-Based Adaptively Parallel Computing, *Proc. of the 2002 joint ACM-ISCOPE conference on Java Grande*, (2002).

[16] M. J. Quinn, Analysis and implementation of B&B algorithms on a hypercube multicomputer, *IIEEE Trans. Comput.*, **3**, 384–387 (1990).

[17] M. J. Quinn, N. Deo, An upper bound for the speed-up of parallel best-bound B&B algorithm, *BIT*, **26**, 35–43 (1986).

[18] Nemhauser G. L. and Wolsey L. A. , *Integer and Combinatorial Optimization*, Wiley, (1989).

[19] R. Corra. A parallel formulation for general branch-and-bound algorithms. In A.Ferreira and J.Rolim, editors, *Lecture Notes in Computer Science*, **980**, 395–409 (1995).

[20] R. Lling and B. Monien. Two strategies for solving the vertex cover problem on transputer network. In M.Raynal and J.C-.Bermond, editors, *Distributed Algorithms, Lecture Notes in Computer Science*, 160–170 (1989).

[21] Schrijver A., Theory of Linear and Integer Programming, *Wiley*, (1995).

[22] T. H. Lai, S. Sahni, Anomalies in parallel B&B algorithms, *Res. Contrib.*, **6**, 594–602 (1984).

[23] T. L. Cannon, K. L. Hoffman, Large scale O-l linear programming on distributed workstations, *Ann. Operations Res*, **22**, 181–217 (1990).

[24] Tomar, *Short course: Some applications of combinatorial optimization in telecommunications*, (2003).

[25] V. Kumar, G. Y. Ananth, *Parallel Algorithms for Discrete Optimisation Problems*, Department of Computer Science, University of Minnesota, Minneapolis, (1992).

[26] Y. Tanaka, M. Sato, M. Hirano, H. Nakada, and S. Sekiguchi, Performance evaluation of a firewall compliant globus-based wide-area cluster system, *Proc. of IEEE Symposium on High-Performance Distributed Computing,*, **9**, (2000).

[27] Yong Ching Lim, Fellow, IEEE, Y. Sun, and Ya Jun Yu, Student Member, IEEE, Design of Discrete-Coefficient FIR Filters on Loosely Connected Parallel Machines, *IEEE transactions on signal processing*, **50**, (2002).

[28] Yoshiaki Futakata, Tokyo Institute of Technology, Parallel Branch and Bound Algorithm with the Hierarchical Master-Worker Paradigm on the grid, *Presto, JST*, **2**, (2006).

Mahmoud Mohammed Ismail got his BS.C. in information system and technology in May 2004, Faculty of Computers and Informatics, Zagazig University. He is Teaching Assistant in decision Support Department, Faculty of Computers and Informatics, Zagazig University..

Osama Abdel-Raouf Abdel-Rahman Lecturer of Operation Research and decision support systems Faculty of Computers and Information Menoufia University got his B.Sc.In Electrical Engineering faculty of engineering Menoufia University, M.Sc.In Electrical Engineering faculty of engineering Menoufia in 2000, PhD in computers and information- Menoufia university in 2008, Demonstrator, 1997-2002 in electrical engineering department faculty of engineering - Menoufia university, Assistant Lecturer, 2002-2008 Operation Research Department - Faculty of Computers and Information Menoufia University, and Lecturer, 2008-20013 Operation Research Department - Faculty of Computers and Information Menoufia University.

Waiel Fathi Abd El-Wahed Professor of Operation Research and decision support systems Faculty of Computers and Information Menoufia University got his BS.C. in electrical power engineering in May 1985, Master of sciences in Engineering mathematics, 1989, and PH.D. in engineering Mathematics, 1993. The author published his papers in Fuzzy sets and Systems, OMEGA and other local and international conferences. Also, the author has contributed in books with chapters. The interest areas of research are intelligent optimization, multi-objective optimization, information science, and decision support.