A DISJUNCTIVE GRAPH TECHNIQUE FOR SHOP SCHEDULING PROBLEMS

by 6791

KUNG-YING CHIU

B.E. (I.E.), Tunghai University

Taichung, Taiwan, Republic of China, 1968

A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1971

Approved by:

[illegible]

Major Professor

ACKNOWLEDGEMENT

I wish to express my deep sense of appreciation to my major professor, Dr. Said Ashour, for his kind guidance and the personal interest taken in preparation of this master's report.

I also appreciate Mr. T. E. Moore, Senior Operations Research Analyst at North American Rockwell Corporation, for his interest in coding this algorithm and valuable suggestions.

I am also thankful to Dr. F. A. Tillman, Head, Department of Industrial Engineering; Professor J.J. Smaltz, Department of Industrial Engineering and Professor A. H. Duncan, Department of Mechanical Engineering for their kind patronage.

I greatly thank Mrs. Marie Jirak for her assistance in typing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Scheduling problem is perhaps one of the most frequently occuring real-world problems. This problem is of interest because, there is no optimization technique sufficiently powerful to handle one of realistic size. It is not that these problems are mathematically very complicated to state, on the contrary, they can usually be formulated into extremely simple mathematical model for which methods of solution are readily available. The difficulty, however, is the combinatorial nature of these problems.

The development of PERT and CPM was a breakthrough in this area which made it possible for a "pure" scheduling problem (the problem in which only the precedence constraints are added) to be numerically solved for practically any size that might occur. The reason that the above problems can be solved efficiently is that they can be stated as critical path or network flow problems. However, for most real-world situations these models are over-simplified, because they assume unlimited resources. That is, the resources required from project i are available whenever the preceding projects are completed, so that project i could be started immediately. But resources are usually limited in varying time period. To make these models more realistic, one has to introduce various types of resource constraints such as machine availability and capital availability. But as soon as this is done, the problem ceases to be a simple longest path or a network flow problem, and the size of the project for which an optimal solution can be found is reduced. The interest arises recently then to find another formulation or structure of the scheduling problem under resource constraints which should permit the use of network flow techniques.

The machine sequencing problem can be seen as a special case of a scheduling problem under resource constraints. It is a problem of determining the sequence, in which a number of jobs to be processed on various machines in a specified technological ordering such that a certain objective is optimized. Among the objectives usually considered are: (1) minimization of maximum completion time; (2) minimization of the maximum lateness; (3) minimization of in-process inventory; (4) minimization of total idle time; (5) maximization of the machine utilization; (6) minimization of total cost. The machine scheduling problem has been approached by integer-linear programming, dynamic programming, enumerative and simulation techniques.

In this research, the machine scheduling problem has been formulated as a two terminal network flow problem for which the disjunctive constraints are binding. These constraints arise, for instance, when both operations require the same machine during the same period. The objective is to minimize the schedule time, ie., maximum completion time.

1.1 Graph Theoretical Formulation of the Problem

Suppose we have J jobs to be processed on M machines. Each job is to be processed on a particular machine according to the given technological ordering. Such an operation which pertains to job j and machine m is designated as (jm). Each operation (jm) is represented by a node

(k) such that

$$(k) = j + (m-1)J, \qquad j = 1, 2, \ldots, J; \qquad m = 1, 2, \ldots, M .$$

Associated with each operation or node (k), there is a processing time $t(k)$, that is, the time required to perform the operation (k).

In order to illustrate the above formulation, let us consider an example having two jobs to be processed on three machines. The data of this problem is displayed in Table 1.1 as shown below.

Table 1.1 (2x3) Job-Shop Problem

Job j	1			2		
Machine m	1	3	2	2	1	3
Operation (jm)	(11)	(13)	(12)	(22)	(21)	(23)
Operation index number (k)	(1)	(5)	(3)	(4)	(2)	(6)
Processing time $t(k)$	5	4	1	2	1	3

The first row indicates that we have two jobs, job 1 and job 2. The second and third rows show the machine ordering for each job. We therefore know that job 1 must be processed on machine 1 first, machine 3 second and machine 2 last. However, job 2 must be performed on machine 2 first, machine 1 second and machine 3 last. The fourth row indicates the index number of each operation. The node index of operations (11), (13) and (12) can be obtained respectively by the above formula. Similarly, the index number for operations (22), (21) and (23) are (4), (2) and (6), respectively. Associated with each operation (k), there is a processing time $t(k)$ as shown in the last row of table 1.1.

Since it will be necessary to consider the precedence constraints and the machine availability constraints for our problem. The following two sets of operations are defined. First, the set of operations which pertain the same job j such that

$M_j = \{(k) \in N\}$, for each job j

where N is the set of nodes. Thus, the set M_j in our example is shown below

$M_1 = \{(1), (5), (3)\}$

$M_2 = \{(4), (2), (6)\}$

M_1 indicates the precedence relations of operations for job 1 are (1) directly precedes (5), and (5) directly precedes (3). M_2 can be interpreted in the same fashion. Next, the set of operations to be performed on the same machine m is such that

$J_m = \{(k) \in N\}$, for each machine m

the set J_m in our example is shown below

$J_1 = \{(1), (2)\}$

$J_2 = \{(3), (4)\}$

$J_3 = \{(5), (6)\}$

Now, our problem may be formulated as follows:

minimize $s(n+1)$ (1)

subject to $s(\ell) - s(k) \geq t(k)$, $(k), (\ell) \in M_j$ (2)

$j = 1, 2, \ldots, J$,

$s(k) - s(\ell) \geq t(\ell)$, (3)

or

$s(\ell) - s(k) \geq t(k)$, $(k), (\ell) \in J_m$

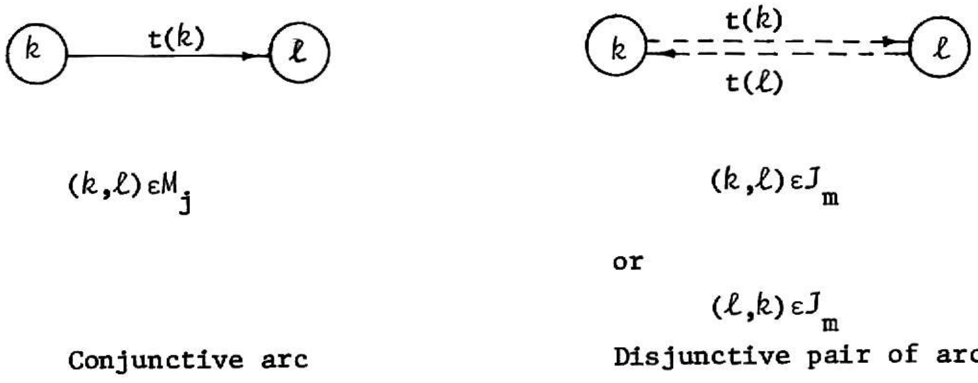
$m = 1, 2, \ldots, M$,

and

$t(k) \geq 0$, $(k) \in N$

Where $s(k)$ is the starting time of operation (k), for $k = 0, 1, 2, \ldots n, n+1$. Nodes (0) and $(n+1)$ are the source and sink of the network. In the above model, constraint (2) expresses the precedence relation fixed by the given technological ordering for those operations which pertain to each job. While constraint (3) indicates the resource restrictions or machine availability constraints which translate the requirement that no time overlap is allowed between any two operations that have to be performed on the same machine.

Following the convention of representing operations (or activities) by nodes, the above mathematical formulation has natural graphical description. Let the set of nodes in the graph, N represents the operations (k), for $k = 0, 1, \ldots, n, n+1$, where node 0 (the source) is the initial operation, while node $n+1$ (the sink) is the final operation which indicates that all operations are completed. The conjunctive constraint (2) is represented by a directed arc from (k) to (ℓ) with length $t(k)$



$t(k)$
$t(k)$
$t(\ell)$
$(k,\ell) \varepsilon M_j$
$(k,\ell) \varepsilon J_m$
or
$(\ell,k) \varepsilon J_m$
Conjunctive arc
Disjunctive pair of arcs

Figure 1.1 Graphical Representation of Constraints

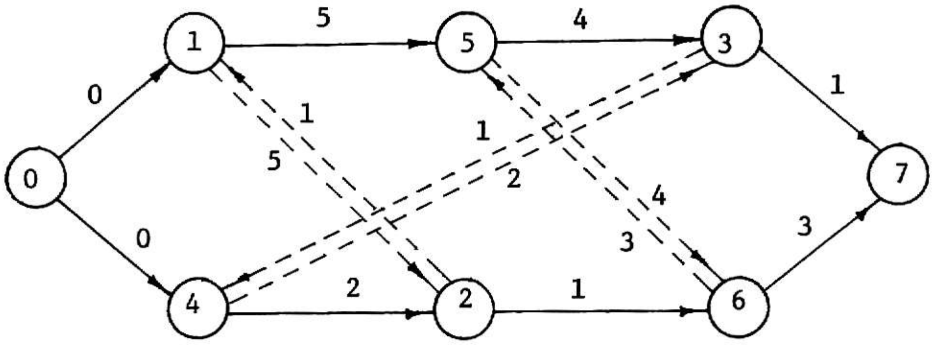


Figure 1.2 Graphical Representation of the (2x3) Job-Shop Problem

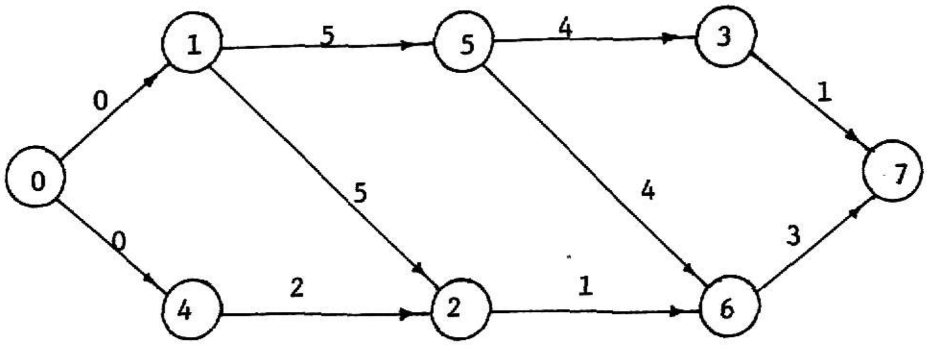


Figure 1.3 Graphical Representation of the Selections of the Arcs {(1,2), (5,4)} for (2x3) Job-Shop Problem

as shown in Figure 1.1. The disjunctive constraint (3) can be represented by a pairwise disjunctive arcs (k,ℓ) and (ℓ,k) with length $t(k)$ and $t(\ell)$, respectively, as shown in Figure 1.1. Considering both constraints (2) and (3), our example can be expressed as a disjunctive graph G, as shown in Figure 1.2.

A disjunctive graph G is usually designated as

$$G = (\mathcal{N},\mathcal{C},\mathcal{D})$$

where $\mathcal{N}$ is the set of nodes, $\mathcal{C}$ the set of conjunctive arcs representing constraint (2) and $\mathcal{D}$ the set of pairwise disjunctive arcs representing constraint (3). A subset of $\mathcal{D}$ containing at most one arc of each disjunctive pair $\{(k,\ell), (\ell,k)\} \in \mathcal{D}$ is called a set of disjunctive arcs, and designated as A. Each set of disjunctive arcs generate a graph of the form

$$G' = (\mathcal{N},\mathcal{C} \cup A)$$

Figure 1.3 shows the graphical representation of the set of selected disjunctive arcs $\{(1,2), (5,6)\}$ for our example.

The machine scheduling problem is then equivalent to one of finding an optimal set of disjunctive arcs, so that the schedule time of the graph can be minimized.

1.2 Literature Review

A number of approaches for solving shop scheduling problems have been proposed. The combinatorial nature of the problem suggests the use of enumerative techniques as methods of solution. The enumerative method is a search technique designed to obtain a subset of optimal solutions from a larger set of feasible schedules. Land and Doig [11] have first developed the enumerative method. It has been named as the branch-and-bound technique by Little _et_. _al_. [12] while solving the traveling salesman problem.

Brooks and White [6] have proposed a branch-and-bound technique which is based on the Gantt chart algorithm [9]. They have modified the algorithm by imbedding a bounding procedure as a criterion for resolving the conflict among operations. The refined algorithm shows that each active schedule developed is better than the preceding one, and each step eliminates solutions for consideration. They have also compared the experimental results obtained by using lower bound as the criterion for resolving the conflicts and those obtained by using the shortest operation time rule and longest remaining time rule. The size of the problems solved varies between (7x10) and (10x18). For the objective of minimization of total schedule time, the results obtained by using lower bound are better than those when the other two criteria are used. However, they have reported on their computational experience that their procedure is too long to adopt on medium size computers, even for problems of moderate size. According to the comparative evaluation of various bounding procedures presented by Ashour and Hiremath [1,2]. By their method, it requires 942 second of mean excution time with 13202 nodes explored to solve (6x5) job-shop problems. However, for (6x4) job-shop problems, the optimal solution has not been reached after the IBM 360/50 computer spent 3600 seconds of mean excution time.

Ashour and Hiremath [1,2] have devised an efficient branch-and-bound technique by imbedding a powerful bounding procedure to reduce the number of nodes explored and hence the computational time involved. Their technique consists of the branching, bounding and backtracking processes which based on the concept of resolving the conflicts among operations. A mathematical analysis and comparative evaluation of various bounding procedures are also presented. A considerable number of experiments has been conducted on IBM 360/50 computer. The size of the problems varies between 3 to 12 jobs and 3 to 5 machines. The results reveal that the job-machine-based bound is more powerful in terms of the number of nodes explored and the computational time required to obtain the optimal solution. The quality of solutions obtained without backtracking is very high. By their method, job-shop problems upto a maximum size of (12x3) or 36 operations can be solved within a reasonable prespecified computer time. The solution of these problems require the exploration of an average number of 4307 nodes and 787 seconds of mean excution time. In this technique, the number of nodes explored can be expected to be reduced by introducing a better criterion to break the tie among lower bounds.

Schrage [14] has proposed a branch-and-bound technique which based on the concept of permutation of the operations. His technique has been developed for non-redundantly enumerating all active schedules by imbedding a dominat check and a redundant check procedures. Both a resource-based bound and a precedence-based bound have been devised to allow backtracking. The algorithm was programmed in FORTRAN for an IBM 7094 computer. A wide variety of resource constrained network problems have been solved. One class of problems solved were the job-shop problems described in [13]. The largest size of job-shop problem that can be solved with proving optimality is a (6x6) or 36 operations problem. It requires 186 iterations to reach the optimal solution. An iteration is defined to be the scheduling of one activity (or operation). The results also show that it requires only 113.4 seconds (or 1.89 min.) of total excution time to solve one (6x6) and two (10x10) job-shop problems. However, the calculations for the two (10x10) job-shop problems were terminated at 4,000 and 2,000 iterations, respectively without proving optimality. Generally speaking, his algorithm represents a general approach to a wide variety of scheduling problems under resource constraints.

Balas [3,4] has developed an implicit enumerative method for solving the machine scheduling problem by finding a mini-maximal path in a disjunctive graph. The basic concept of his method is to generate a sequence of graph and to solve a critical path problem for each graph. Each graph associated with a critical path is represented as a node in the search tree. To branch from one graph is equivalent to reverse an arc on the critical path in the graph. His method starts with a graph in which one of each disjunctive pairs are included. Four major steps are involved in his technique: (1) a testing procedure in which a lower bound is computed and compared with the current solution; (2) an evaluation procedure in which an upper bound is computed and compared with the current solution. In this procedure, all candidate arcs on the critical path will be evaluated, the one which has the minimum evaluated value is selected for further branching (3) a forward procedure in which the selected arc is reversed. Accordingly, a new graph associated with the reversed arc is generated; (4) a backtracking procedure which traces back to the preceding graph until the graph has no predecessor. No computational experience has been reported for determining the size of problems this method can solve. The advantage of his algorithm is that the storage requirement are limited to the data concerned with the current node on the search tree. Since a feasible solution is generated at each stage, one can terminate at any stage when a good solution is obtained. The disadvantage of his algorithm may be considered as follows: First, it requires more computational effort. One has to compute the upper and lower bounds for each node in search tree. Then, one has to evaluate the candidate arcs on the critical path to guide the search. Second, it requires the examination of the graph at each stage to detect any loop which may exist. Third, from the definition of the lower bound, more nodes to be explored for proving optimality. Balas [5] has also proposed a modified mixed-integer programming technique which combines the ideas of Bender's partitioning method with implicit enumeration approach. The proposed algorithm is closely related to [3,4]. He generates constraints on a vector with 0 and 1 components which characterizes the state of the search for a subset of disjunctive arcs. The whole procedure of generating those constraints and finding a feasible solution has been interpreted as a process of generating a graph with degree constraints on its nodes, and then finding a subgraph satisfying the degree constraints (the degree of a node is defined as the difference between the number of arcs from and to the node). No computational experience has been reported.

Charlton and Death [7] have proposed an algorithm which based on the concept of resolving the conflicts among nodes on a disjunctive graph. Their method starts with a graph which has no disjunctive arc included. The disjunctive constraints are checked for all operations at each stage. Whenever the disjunctive constraints are not satisfied for some operations to be processed on the same machine, a conflict exists. At such time, a pair of arcs is selected and its direction is decided in favoring one of the operations (nodes) according to their proposed heuristic rule. Each graph is represented as a node in the search tree, to branch from a node is equivalent to including a disjunctive arc into a graph. The branching procedure continues until a feasible solution has been found. The length of the critical path of a graph is a lower bound on the value of the objective function. Whenever the lower bound of a node is found to be less than the value of the current solution, the previously selected arc at that level is reversed. The branching is then proceeded from that node. If both arcs of the disjunctive pair have been already examined, the search is traced back one level up the tree. These steps continue until the initial node has been reached and all others explored. No computational experience has been reported. However, both a (3x2) flow-shop and job-shop problems and a (5x4) job-shop problem (that only has 13 operations) have been comparatively evaluated by manual calculation. The result shows that their method requires much less computational effort as well as iterations than that of Balas [3,4]. In addition, their method is easier to calculate by hand than any other similar techniques. The disadvantages of their method may be considered as follows: (1) the disjunctive constraints have to be checked for all operations at each stage to detect any conflicts that may exist. However, by Ashour and Hiremath's technique [2], only the conflict for unscheduled operations requires checking; and (3) the proposed heuristic rule is used only for a binary decision, no criterion has been suggested to guide the search when there are more than one candidate arcs to be chosen. Charlton and Death [8] have also developed a general algorithm considering the constraints on machine orderings, machine availability, due date and zoining. Their method can be applied to a wide variety of scheduling problems such as flow-shop, job-shop, machine scheduling without preassignment, machine routing and assembly-line balancing problems. All these problems demonstrate the underlying relation with reference to their graphical representation. Their method is essentially a branch-and-bound approach to enumerate implicitly all feasible solutions by using a search procedure, where the nodes in the search tree represent partial allocation of operations to machines. The algorithm can be terminated with a solution whose objective function value is within a prespecified percentage of the lower limit on the optimal schedule time. Since their method represents a general approach to a wide variety of problems, for the solution of specific problems in practice, a tailored algorithm based on the general approach can be expected to be more efficient.

Greenberg [10] has presented a mixed-integer linear programming formulation for the general shop scheduling problem. By applying a branch-and-bound procedure, a series of non-integer linear programming problems are reduced to moderate proportions. The basic approach is to solve a normal linear programming problem with a given objective function and the precedence constraints. All of these are represented as a node in the search tree. To branch from one node is equivalent to adding a disjunctive constraint to the problem. A new explored node with a minimum objective function value is then chosen for further branching. The branching and bounding procedures continue until all sequence on the machines have been specified. The advantage of his technique is that when solving the non-integer linear programming problem, the constraints involving integer variables need not be present, since they are non-restricted in the non-integer problem. A few small size problems have been solved using General Electric 265 Time Sharing System. A (3x2) job-shop problem with 36 possible solutions require the solution of 19 linear programming problems and 143 seconds of computer time. Larger problems were not attacked because of the memory restriction of the computer system.

1.3 Proposed Research

In this report, the general shop scheduling problem has been reformulated as a disjunctive graph. An algorithm which is based on the concept of resolving the conflicts has been developed. The proposed algorithm can be expected to be more efficient than the similar techniques by imbedding a proposed upper bound and a set of heuristic rules. A more efficient method for checking the conflicts among operations has been also proposed. A job-machine-based lower bound is used to guide the search. The algorithm will be illustrated by a sample problem. A comparative evaluation of various techniques for solving published sample problems will also be presented.

CHAPTER II

DEVELOPMENT OF A DISJUNCTIVE GRAPH ALGORITHM

This chapter contains an enumerative algorithm for solving the shop scheduling problem with the disjunctive constraint. In the context of the graphic nature of the problem, the enumerative task is to resolve the conflicts in favor of a particular node by assigning an arc (arcs) from that node to the other(s) in conflict. An upper bound is used in conjunction with lower bounds, so that a possible optimal schedule can be obtained without explicitly enumerating all possible sequences. The algorithm is illustrated by a sample problem and summarized in formal steps.

2.1 Basic Concepts

This algorithm is developed on the basis of three principal concepts: (1) partitioning of the set of possible solutions into successively smaller subsets; (2) the application of a lower bound on the value of the objective function for identification of a subset containing an optimal solution; and (3) the application of an upper bound in conjunction with lower bounds to identify that an optimal solution is reached. The techniques of partitioning the subset of possible solutions are dictated by the structure of the problem to be solved. In this research, the technique of partitioning the subset of possible solution is based on the concept of successively resolving the conflicts among operations to be performed on the same machine at the same time.

The main idea of conflict is based on the disjunctive constraints, that is, no two operations on the same machine may be scheduled at any common interval of time. A schedule that violates this constraint is non-feasible, and the operations involved are said to be in conflict. Whenever there is a set of operations in conflict, resolving the conflict in favor of one of them implies that the starting of the remaining operations must be delayed. Graphical interpretation of resolving a conflict infavor of a particular node (operation) is nothing more than assigning a disjunctive arc (arcs) from the particular node to the other node (nodes) in conflict. If a conflict exists between two or more operations on a particular machine in the same time interval, it should be resolved in favor of one that gives the best chance to reach the optimal solution or at least the near-optimal solution. One possible way is to establish a lower bound on the objective function for each possible resolution . The minimum lower bound of these resolutions provides a basis for resolving this conflict.

The algorithm starts with a graph which has no assigned disjunctive arcs. This graph is denoted as G(0,0). The conflict is then checked for all unscheduled nodes (operations) in the graph. If a conflict exists, *we first select a machine, then we decide to resolve the conflict* in favor of one of the nodes (operations) in conflict on that machine. Each possible resolution is essentially to assign an arc (arcs) from a particular node (k) to the other node (nodes) (ℓ) in conflict. Consequently, new graphs associated with the possible resolutions are generated. These graphs are denoted as $G(k',\ell')$. As will be noted, if the number of the nodes (operations) in the conflict set is $|F_m|$, the number of arcs to be assigned in favor of one of the nodes in conflict is $(|F_m| - 1)$. Accordingly, the number of graphs generated is $|F_m|$. This procedure may be easier to discuss if a search tree is used. In such a tree, each node associated with a graph represents a possible resolution of a conflict. The tree is initialized by a node (graph) G(0,0) representing the set of all active schedules. Each time, if a conflict is found to exist, we go down one level on the tree. The possible resolutions for the conflict at that level can be conveniently denoted by nodes at that level. The node (graph) having the minimum lower bound is selected. That is, a conflict is resolved at that level. This process of generating a new set of nodes (graphs) at a level from a selected node (graph) at the preceding level is referred to as the branching of the tree.

Whenever a node (graph) at certain level has been selected for branching, we generate possible sequences for operations on machines. The value of this possible solution is an upper bound on the value of the objective function. If this value is found to be better than the previous one, the solution is updated. This solution is compared with the minimum lower bound at that level. If the lower bound is found to be equal to the current solution it implies that an optimal solution may be reached and thus, the branching and bounding processes are ceased. The search tree is traced back along the same branch. The graphical interpretation of the backtracking procedure is essentially to remove the assigned disjunctive arc(s) at each level. The backtracking procedure continues until an unexplored node having a lower bound less than the current solution is found. In a similar manner, branching and bounding procedure are repeated until a better solution is obtained. By a series of such backtrackings, the initial level is eventually reached and all potential branches in the search tree are exhausted and thus, the optimal solution is found. The number of conflict levels represents the number of conflicts to be resolved to obtain an optimal schedule. As will be noted, the number of conflicts to be resolved to obtain an optimal solution varies from one problem to another, even for problems having the same size.

As mentioned earlier, the purpose of using an upper bound with lower bound at each level is to obtain an optimal solution without explicitly enumerating all possible sequences. In order to discuss these two bounds, the following common notation is considered:

L	conflict level in the search tree
(k)	node index, $k = 0, 1, \ldots, n+1$
$t(k)$	processing time of node (k)
$s(k)$	starting time of node (k)
$c(k)$	completion time of node (k)
$m(k)$	machine m which is pertained to node (k)
F_m	set of nodes in conflict on machine m
C	set of conjunctive arcs
A'	set of candidate arcs such that $A' = \{(k',\ell')\}$
A^*	set of chosen arcs such that $A^* = \{(k^*,\ell^*)\}$
S_m	sequence of all nodes to be processed on machine m, in which all nodes are sequenced according to the dispatching rules.

A_m	set of sequenced arcs associated with S_m
$G(k',\ell')$	graph associated with the arc(s) (k',ℓ')
$\underline{B}(k',\ell')$	lower bound for arc(s) (k',ℓ')
$\overline{B}(k^*,\ell^*)$	upper bound for the chosen arc(s) (k^*,ℓ^*)
$T(S)$	schedule time
$A(S)$	optimal set of arcs
τ	transition time, before which no conflict exists, and after which conflict may exist.

The lower bound on the schedule time for all candidate arcs is given by

$$\underline{B}(k',\ell') = \max \left\{ \begin{array}{l} \max \left\{ c(k') + \sum\limits_{\substack{(i)\varepsilon m(k') \\ (i)\neq(k')}} t(i),\ \max\limits_{\substack{m(i) \\ m(i)\neq m(k')}} \left\{ \min\limits_{(i)\varepsilon m(i)} s(i) + \sum\limits_{(i)\varepsilon m(i)} t(i) \right\} \right\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad s(n+1) \end{array} \right\}$$

for all unscheduled nodes (i) such that $c(i) \geq \tau$

This bound is a job-machine-based bound. The value of this bound is the maximum of two expressions. The first expression is a machine-based bound. The second expression is a job-based bound. The value of this job-machine-based bound is computed from the data included in the graph $G = (N, C \cup A^* \cup (k',\ell'))$. Such a graph represents a possible resolution in favor of arc(s) (k',ℓ'). This graph is branched directly from the $G = (N, C \cup A^*)$ in the search tree. The starting time of each node in the graph can be computed by critical path method such that

$$s(\ell) = \max_{k<<\ell} [s(k) + t(k)], \qquad k = 0, 1, \ldots, n, \quad \ell = 1, 2, \ldots, n+1.$$

The completion time of each node can be obtained such that

$$c(k) = s(k) + t(k), \qquad k = 1, 2, \ldots, n.$$

The machine based bound is also the maximum of two expressions. The first expression gives the bound on the particular machine, $m(k')$. It consists of two terms. The first term is the completion time of node (k'). This is also the earliest time at which an unscheduled node could be started on machine $m(k')$, because the other nodes in the conflict set, $\{F_m\}$ could be started on machine $m(k')$ immediately after the completion of the node (k'). The second term is the sum of the processing times of the unscheduled nodes (other than node (k')) which require processing on the same machine $m(k')$. The second expression gives the maximum of the bounds on the machine other than $m(k')$. It consists of two terms. The first term is the earliest starting time at which an unscheduled node (i) could be started on machine $m(i)$, where $m(i) \neq m(k')$. The second term is the sum of the processing times of unscheduled node (i) which requires processing on machine $m(i)$, where $m(i) \neq m(k')$.

The upper bound on the schedule time for the chosen arc(s) (k^*,ℓ^*) is determined such that

$$\bar{B}(k^*,\ell^*) = s'(n+1)$$

where $s'(n+1)$ is the starting time of the last node of the graph

$G = (N, C \cup A_m)$

The objective of using an upper bound at each level is to detect an optimal or near-optimal solution before all conflicts among operations have been resolved and thus, the number of nodes explored in the search tree can be reduced. The upper bound can also be used as a criterion to break the tie, if it exists between lower bounds. However, in order to simplify the algorithm, this function is not included. And the tie between lower bounds is simply broken by the one of the following dispatching rules:

1. Select the node which has the earliest completion time, i.e., set $(k) << (\ell)$, if $c(k) < c(\ell)$.
2. Select the node which has the earliest starting time, i.e., set $(k) << (\ell)$, if $s(k) < s(\ell)$.
3. Select the node which has the smallest index number, i.e., set $(k) << (\ell)$, if $(k) < (\ell)$.

As mentioned earlier, the idea of this algorithm is somewhat similar to those reported in [2,7]. However, some significant differences from the other similar methods should be stated. First, in this method, the branching and bounding process may be ceased before all conflicts are resolved. Therefore, the number of nodes explored to reach the optimal solution can be expected to be reduced. Second, since an actual solution is obtained at each level, this algorithm allow one to terminate the search at anytime with a good, but not necessarily optimal, solution. Third, so far as the graphical structure concerned, this algorithm is designed such that the number of arcs assigned depends on the number of operations in conflict at that level, while the existing methods [4,7] are structured in such a way that only one arc to be chosen at each level.

2.2 Sample Problem

In order to demonstrate the proposed algorithm, a sample problem consisting of four jobs and three machines presented in [2] is used. The problem is given as shown below.

Job j	1			2			3			4		
machine m	2	3	1	1	3	2	3	1	2	1	2	3
operation (jm)	(12)	(13)	(11)	(21)	(23)	(22)	(33)	(31)	(32)	(41)	(42)	(43)
node (k)	(5)	(9)	(1)	(2)	(10)	(6)	(11)	(3)	(7)	(4)	(8)	(12)
processing time t(k)	4	2	3	8	4	5	6	3	9	7	6	2

In order to follow the method of solution easily, Table 2.1 and Figure 2.17 should be followed throughout all steps.

Step 1. Initialize the scheduling tree with node (graph) G(0,0) at level L = 0, set the transition time $\tau = 0$ and schedule time $T(S) = 0$. Construct the initial graph, G(0,0) as shown in Figure 2.1. The numbers in two boxes for each node (k) are, from left to right $s(k)$ and $c(k)$ which can be computed by critical path method such that

$$s(\ell) = \max_{k<<\ell} [s(k) + \tau(k)], \qquad k = 0, 1, \ldots, 12, \quad \ell = 1, 2, \ldots, 13,$$

and

$$c(k) = s(k) + t(k), \qquad k = 1, 2, \ldots, 12,$$

or

$$s(5) = [s(0) + t(0)] = 0,$$

$$s(2) = [s(0) + t(0)] = 0,$$

$$s(11) = [s(0) + t(0) = 0,$$

$$s(4) = [s(0) + t(0)] = 0,$$

$$s(9) = [s(5) + t(5)] = 0 + 4 = 4,$$

$$s(10) = [s(2) + t(2)] = 0 + 8 = 8,$$

$$s(3) = [s(11) + t(11)] = 0 + 6 = 6,$$

$$s(8) = [s(8) + t(8)] = 0 + 7 = 7,$$

$$s(1) = [s(9) + t(9)] = 4 + 2 = 6,$$

$$s(6) = [s(6) + t(6)] = 8 + 4 = 12,$$

$$s(7) = [s(7) + t(7)] = 6 + 3 = 9,$$

$$s(12) = [s(8) + t(8)] = 7 + 6 = 13,$$

$$s(13) = \max\begin{Bmatrix} s(1) + t(1) \\ s(6) + t(6) \\ s(7) + t(7) \\ s(12) + t(12) \end{Bmatrix} = \max\begin{Bmatrix} 6 + 3 \\ 12 + 5 \\ 9 + 9 \\ 13 + 2 \end{Bmatrix} = \max\begin{Bmatrix} 9 \\ 17 \\ 18 \\ 15 \end{Bmatrix} = 18,$$

and

$$c(5) = s(5) + t(5) = 0 + 4 = 4,$$

$$c(2) = s(2) + t(2) = 0 + 8 = 8,$$

$$c(11) = s(11) + t(11) = 0 + 6 = 6,$$

$$c(4) = s(4) + t(4) = 0 + 7 = 7,$$

$$c(9) = s(9) + t(9) = 4 + 2 = 6,$$

$c(10) = s(10) + t(10) = 8 + 4 = 12,$

$c(3) = s(3) + t(3) = 6 + 3 = 9,$

$c(8) = s(8) + t(8) = 7 + 6 = 13,$

$c(1) = s(1) + t(1) = 6 + 3 = 9,$

$c(6) = s(6) + t(6) = 12 + 5 = 17,$

$c(7) = s(7) + t(7) = 9 + 9 = 18,$

$c(12) = s(12) + t(12) = 13 + 2 = 15.$

Step 2. Check for conflicts between nodes pertained to the same machine by the relation

$$- t(\ell) < s(\ell) - s(k) < t(k), \quad \text{for } c(\ell) \geq 0 \quad \text{and} \quad c(k) \geq 0.$$

Since all nodes (k) and (ℓ) have $c(k)$ and $c(\ell) \geq 0$, so all of them have to be checked for conflict. We first check conflict between nodes pertained to machine 1 as follows:

$- t(2) < s(2) - s(1) < t(1)$ or $- 8 < 0 - 6 < 3,$

$- t(3) < s(3) - s(2) < t(2)$ or $- 3 < 6 - 0 < 8,$

$- t(4) < s(4) - s(3) < t(3)$ or $- 7 < 0 - 6 < 3.$

Therefore, a conflict exists among nodes (1), (2), (3) and (4) on machine 1. Then, check the conflict between nodes pertained to machine 2 as follows:

$- t(6) < s(6) - s(3) \nless t(5)$ or $- 5 < 12 - 0 \nless 4,$

$- t(7) < s(7) - s(5) \nless t(5)$ or $- 9 < 9 - 0 \nless 4,$

$- t(8) < s(8) - s(5) \nless t(5)$ or $- 6 < 7 - 0 \nless 4,$

$- t(7) < s(7) - 3(6) < t(6)$ or $- 9 < 9 - 12 < 5,$

$- t(8) < s(8) - s(7) < t(7)$ or $- 6 < 7 - 9 < 9,$

Therefore, a conflict exists among (6), (7) and (8) on machine 2. We then check the conflict between nodes pertained to machine 3 as follows:

$$- t(10) < s(10) - s(9) \nless t(9) \quad \text{or} \quad - 8 < 8 - 4 \nless 2,$$

$$- t(11) < s(11) - s(9) < t(9) \quad \text{or} \quad - 6 < 0 - 4 < 2,$$

$$- t(12) < s(12) - s(9) \nless t(9) \quad \text{or} \quad - 2 < 13 - 4 \nless 2,$$

$$- t(11) \nless s(11) - s(10) < t(10) \quad \text{or} \quad - 6 \nless 0 - 8 < 8,$$

$$- t(12) \nless s(12) - s(10) < t(10) \quad \text{or} \quad - 2 \nless 2 - 4 < 4,$$

$$- t(12) < s(12) - s(11) \nless t(11) \quad \text{or} \quad - 2 < 13 - 0 \nless 6.$$

Therefore, a conflict exists between node (9) and (11) on machine 3. Now we select a machine m(k) such that

$$c(k) = \min c(\ell), \qquad \text{for} \quad = 1, 2, 3, 4, 6, 7, 8, 9, 11.$$

since node (9) has the minimum completion time c(9) or 6, we select machine 3 on which the conflict will be resolved in favor of node (9) or node (11). Set L = L+1, or 1 and τ = c(9), or 6.

Step 3. Compute the lower bound for candidate arcs (9,11) and (11,9). We first compute the lower bound $\underline{B}(9,11)$ for arc (9,11) from the data pertained in the graph G(9,11) as shown in Figure 2.2 such that

$$\underline{B}(9,11) = \max \left\{ \max \left\{ \begin{array}{l} c(9) + [t(10) + t(11) + t(12)] \\ \max \left\{ \begin{array}{l} \min \begin{Bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \end{Bmatrix} + [t(1) + t(2) + t(3) + t(4)] \\ \min \begin{Bmatrix} s(6) \\ s(7) \\ s(8) \end{Bmatrix} + [t(6) + t(7) + t(8)] \end{array} \right\} \end{array} \right\}, s(13) \right\}$$

$$= \max\left\{\max\left\{\max\begin{Bmatrix} 6 + (4 + 6 + 2) \\ \\ \left\{\begin{matrix} \min\begin{Bmatrix} 6 \\ 0 \\ 12 \\ 0 \end{Bmatrix} + (3 + 8 + 3 + 7) \\ \\ \min\begin{Bmatrix} 12 \\ 15 \\ 7 \end{Bmatrix} + (5 + 9 + 6) \end{matrix}\right\} \end{Bmatrix}\right\}, 24\right\}$$

$$= \max\left\{\max\left\{\max\begin{Bmatrix} 18 \\ \begin{Bmatrix} 0 + 21 \\ \\ 7 + 20 \end{Bmatrix} \end{Bmatrix}, 24\right\}\right\}$$

$$= \max\left(\max\,[18,\ 27],\ 24\right)$$

$$= \max\,[27,\ 24]$$

$$= 27.$$

Similarly, the lower bound $\underline{B}(11, 9)$ is computed from the data pertained in the graph G(11,9) as shown in Figure 2.3 such that

$$\underline{B}(11,9) = \max\left\{\max\left\{\begin{array}{l} c(11) + [t(9) + t(10) + t(12)] \\ \max\left\{\begin{array}{l} \min\begin{Bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \end{Bmatrix} + [t(1) + t(2) + t(3) + t(4)] \\ \\ \min\begin{Bmatrix} s(6) \\ s(7) \\ s(8) \end{Bmatrix} + [t(6) + t(7) + t(8)] \end{array}\right\} \end{array}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{array}{l} 6 + (2 + 4 + 2) \\ \max\left\{\begin{array}{l} \min\begin{Bmatrix} 8 \\ 0 \\ 6 \\ 0 \end{Bmatrix} + (3 + 8 + 3 + 7) \\ \\ \min\begin{Bmatrix} 12 \\ 9 \\ 7 \end{Bmatrix} + (5 + 9 + 6) \end{array}\right\} \end{array}\right\}, 18\right\}$$

$$= \max\left\{\max\left\{\begin{array}{l} 14 \\ \max\left\{\begin{array}{l} 0 + 21 \\ \\ 7 + 20 \end{array}\right\} \end{array}\right\}, 18\right\}$$

$$= \max\{\max[14, 27], 18\}$$

$$= \max[27, 18]$$

$$= 27.$$

Step 4. Since $\underline{B}(9,11) = \underline{B}(11,9) = 27$, a tie exists between the two bounds. We therefore break the tie by one of the dispatching rules. We first check the completion time of node (11) and node (9) as shown in G(0,0). As $c(11) = c(9) = 6$, a tie still exists, we therefore check the starting time of them. Since $s(11) < s(9)$, or $0 < 6$. We choose the arc (11,9).

Step 5. Compute the upper bound $\bar{B}(11,9)$. By following the dispatching rules, we generate the sequences for all nodes on machines from the data pertained in G(11,9) as follows: We first consider the completion time of all nodes on machine 1. Since

$c(4) < c(2) < c(3) <$ (1) or $7 < 8 < 9 < 11$,

we have the sequence $(4) \ll (2) \ll (3) \ll (1)$ or $S_1 = \{(4)\ (2)\ (3)\ (1)\}$. Similarly, we have

$$S_2 = \{(5)\ (8)\ (6)\ (7)\}$$

and

$$S_3 = \{(11)\ (9)\ (10)\ (12)\}$$

These sequences are associated with a set of sequenced arcs, A_m such that

$$A_m = \{(4,2)\ (2,3),\ (3,1);\ (5,8),\ (8,6),\ (6,7);\ (11,9),\ (9,10),\ (10,12)\}$$

By assigning the set of arcs, A_m into the graph G(0,0), we obtain a graph $\bar{G}(11,9)$ as shown in Figure 2.4. The upper bound $\bar{B}(11,9)$ is computed from the graph $\bar{G}(11,9)$ such that

$$\begin{aligned}\bar{B}(11,9) &= s'(13)\\ &= 33.\end{aligned}$$

Since $\bar{B}(11,9) < T(S)$ or $33 < \infty$, we set $T(S) = 33$ and $A(S) = A_m$.

Step 6. Compare the minimum lower bound with the current solution. The lower bound $\underline{B}(11,9) < T(S)$ or $29 < 33$, we include the arc (11,9) in the set of chosen arcs A^*. We then branch from the graph G(11,9) and go to step 2.

Step 2. Check for conflicts between nodes pertained to the same machine by the relation

$$- t(\ell) < s(\ell) - s(k) < t(k), \quad \text{for } c(k) \geq 6 \quad \text{and} \quad c(\ell) \geq 6.$$

We first check conflicts between nodes pertained to machine 1 as follows:

$- t(2) \nless s(2) - s(1) < t(1)$ or $-8 \nless 0 - 8 < 3$,

$- t(3) < s(3) - s(1) < t(1)$ or $-3 < 6 - 8 < 3$,

$- t(3) < s(3) - s(2) < t(2)$ or $-3 < 6 - 0 < 8$,

$- t(4) < s(4) - s(3) < t(3)$ or $-7 < 0 - 6 < 3$.

Therefore, a conflict exists among nodes (1), (2), (3) and (4) on machine 1. Then, check the conflict between nodes pertained to machine 2 as follows: Since the node (5) has completion time $c(5) < \tau$ or $4 < 6$, we then check the conflict between the other unscheduled nodes.

$- t(7) < s(7) - s(6) < t(6)$ or $-9 < 9 - 12 < 5$,

$- t(8) < s(8) - s(7) < t(7)$ or $-6 < 7 - 9 < 9$.

Therefore, a conflict exists among nodes (6), (7) and (8) on machine 2. We then check the conflict between unscheduled nodes (9), (10) and (12) on machine 2 as follows:

$- t(10) < s(10) - s(9) \nless t(9)$ or $- 4 < 8 - 6 \nless 2$

$- t(12) < s(12) - s(9) \nless t(9)$ or $- 2 < 13 - 6 \nless 2$

$- t(12) < s(12) - s(10) \nless t(10)$ or $- 2 < 13 - 8 \nless 4$

Therefore, no conflict exists among nodes on machine 3. Now we select a machine m(k) such that

$$c(k) = \min c(\ell) \quad \text{for } \ell = 1, 2, 3, 4, 6, 7, 8.$$

Since node (4) has the minimum completion time c(4) or 7, we decide to resolve the conflict among nodes (1), (2), (3) and (4) on machine 1. Set L = L+1 or 2 and τ = c(4) or 7.

Step 3. Compute the lower boudn for candidate arcs ((1,2), (1,3), (1,4)), ((2,1), (3,2), (3,4)) and ((4,1), (4,2), (4,3)). In a similar manner, the lower bounds are computed from the graphs which can be obtained by assigning the candidate arcs into the graph G(11,9), respectively. We first compute the lower bound $\underline{B}$ ((1,2), (1,3), (1,4)) from the data pertained in G((1,2), (1,3), (1,4)) as shown in Figure 2.5 such that

$$\underline{B}((1,2)(1,3),(1,4)) = \max\left\{\max\left\{\begin{array}{l} c(1)+ [t(2) + t(3) + t(4)] \\ \max\left\{\begin{array}{l} \min\left\{\begin{array}{l} s(6) \\ s(7) \\ s(8) \end{array}\right\} + [t(6) + t(7) + t(8)] \\ \\ \min\left\{\begin{array}{l} s(9) \\ s(10 \\ s(12) \end{array}\right\} \quad [t(9) + t(10) + t(12)] \end{array}\right\} \end{array}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{array}{l} 11 + (8 + 3 + 7) \\ \max\left\{\begin{array}{l} \min\left\{\begin{array}{r} 23 \\ 14 \\ 18 \end{array}\right\} + (5 + 9 + 6) \\ \\ \min\left\{\begin{array}{r} 6 \\ 19 \\ 24 \end{array}\right\} + (2 + 4 + 2) \end{array}\right\} \end{array}\right\}, 28\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}29 \\ \max\left\{\begin{matrix}14 + 20 \\ \\ 6 + 8\end{matrix}\right\}\end{matrix}\right\}, 28\right\}$$

$$= \max\{\max[29, 34], 28\}$$

$$= \max[34, 28]$$

$$= 34.$$

Similarly, B((2,1), (2,3) (2,4)) is computed from G((2,1), (2,3) (2,4)) as shown in Figure 2.6 such that

$$\underline{B}((2,1),(2,3),(2,4)) = \max\left\{\max\left\{\begin{matrix}c(2) + [t(1) + t(3) + t(4)] \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}s(6)\\s(7)\\s(8)\end{Bmatrix} + [t(6) + t(7) + t(8)] \\ \\ \min\begin{Bmatrix}s(9)\\s(10)\\s(12)\end{Bmatrix} + [t(9) + t(10) + t(12)]\end{matrix}\right\}\end{matrix}\right\}, 23\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}8 + (3 + 3 + 7) \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}12\\11\\15\end{Bmatrix} + (5 + 9 + 6) \\ \\ \min\begin{Bmatrix}6\\8\\21\end{Bmatrix} + (2 + 4 + 2)\end{matrix}\right\}\end{matrix}\right\}, 23\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}21 \\ \max\left\{\begin{matrix}11 + 20 \\ \\ 6 + 8\end{matrix}\right\}\end{matrix}\right\}, 23\right\}$$

$$= \max\left(\max\,[21, 31], 23\right)$$

$$= \max\,[31, 23]$$

$$= 31.$$

The lower bound for arcs ((3,1), (3,2), (3,4)) is computed from the data pertained in G((3,1) (3,2), (3,4)) as shown in Figure 2.7 such that

$$\underline{B}((3,1),(3,2),(3,4)) = \max\left\{\max\left\{\begin{matrix}c(3) + [t(1) + t(2) + t(4)] \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}s(6)\\ s(7)\\ s(8)\end{Bmatrix} + [t(6) + t(7) + t(8)] \\ \\ \min\begin{Bmatrix}s(9)\\ s(10)\\ s(12)\end{Bmatrix} + [t(9) + t(10) + t(12)]\end{matrix}\right\}\end{matrix}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}9 + (3 + 8 + 7) \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}21\\ 9\\ 16\end{Bmatrix} + (5 + 9 + 6) \\ \\ \min\begin{Bmatrix}6\\ 17\\ 22\end{Bmatrix} + (2 + 4 + 2)\end{matrix}\right\}\end{matrix}\right\}, 26\right\}$$

$$= \max\left\{\max\left[\max\begin{Bmatrix}27 \\ \max\begin{Bmatrix}9 + 20 \\ \\ 6 + \ \ 8\end{Bmatrix}\end{Bmatrix}\right], 26\right\}$$

$$= \max\{\max\,[27,\ 29],\ 26\}$$

$$= \max\,[29,\ 26]$$

$$= 29.$$

Similarly, the lower bound for arcs ((4,1), (4,2), (4,3)) is computed from the graph *G* ((4,1), (4,2), (4,3)) as shown in Figure 2.8 such that

$$\underline{B}((4,1),(4,2),(4,3)) = \max\left\{\max\begin{Bmatrix}c(4) + [t(1) + t(2) + t(3)] \\ \max\begin{Bmatrix}\min\begin{Bmatrix}s(6) \\ s(7) \\ s(8)\end{Bmatrix} + [t(6) + t(7) + t(8) \\ \\ \min\begin{Bmatrix}s(9) \\ s(10) \\ s(12)\end{Bmatrix} + [t(9) + t(10) + t(12)\end{Bmatrix}\end{Bmatrix}, s(13)\right\}$$

$$= \max\left\{\max\begin{Bmatrix}7 + (3 + 8 + 3) \\ \max\begin{Bmatrix}\min\begin{Bmatrix}19 \\ 10 \\ 7\end{Bmatrix} + (5 + 9 + 6) \\ \\ \min\begin{Bmatrix}6 \\ 15 \\ 13\end{Bmatrix} + (2 + 4 + 2)\end{Bmatrix}\end{Bmatrix}, 24\right\}$$

$$= \max\left\{\max\left[\begin{matrix}21 \\ \max\left(\begin{matrix}7 + 20 \\ \\ 6 + \ \ 8\end{matrix}\right)\end{matrix}\right], \ 24\right\}$$

$$= \max\{\max\ [21,\ 27],\ 24\}$$

$$= \max\ [27,\ 24]$$

$$= 27.$$

Step 4. Choose the arcs ((4,1), (4,2), (4,3)) with the minimum lower bound $\underline{B}$((4,1), (4,2),(4,3)) or 27.

Step 5. Compute the upper bound for arc ((4,1), (4,2), (4,3)). By following the dispatching rules, we generate the sequences for all nodes on machines from the data pertained in G ((4,1), (4,2), (4,3)) as follows:

S_1 = {(4) (3) (1) (2)}

S_2 = {(5) (8) (7) (6)}

S_3 = {(11) (9) (12) (10)}

These sequences are associated with a set of sequenced arcs, A_m such that

A_m = {(4,3),(3,1),(1,2);(5,8),(8,7),(7,6); (11,9), (9,12), (12,10)}

By assigning the set of arcs, A_m into the graph G(0,0), we have a graph $\bar{G}$((4,1),(4,2),(4,3)) as shown in Figure 2.9. The upper bound $\bar{B}$((4,1), (4,2),(4,3)) is computed from the graph $\bar{G}$((4,1),(4,2),(4,3)) such that

$$\bar{B}((4,1),(4,2),(4,3)) = s'(13)$$

$$= 30.$$

Since $\bar{B}((4,1),(4,2),(4,3)) < T(S)$ or $30 < 33$, we set $T(S) = 30$ and $A(S) = A_m$.

Step 6. The lower bound $\underline{B}((4,1),(4,2),(4,3)) < T(S)$ or $27 < 30$. We therefore include the arcs ((4,1), (4,2), (4,3)) into the set A^*. We then branch from the graph $G((4,1),(4,2),(4,3))$ and go to step 2.

Step 2. Check for conflicts between nodes pertained to the same machine by the relation

$$-t(\ell) < s(\ell) - s(k) < t(k), \quad \text{for } c(k) \geq 7 \quad \text{and} \quad c(\ell) \geq 7.$$

We first check the conflicts between unscheduled nodes (1), (2) and (3) on machine 1 as follows:

$$-t(2) < s(2) - s(1) < t(1) \quad \text{or} \quad -8 < 7 - 8 < 3,$$

$$-t(3) < s(3) - s(2) < t(2) \quad \text{or} \quad -3 < 7 - 7 < 8.$$

Therefore, a conflict exists among nodes (1), (2) and (3) on machine 1. We then check the conflict between unscheduled nodes (6), (7) and (8) on machine 2 as follows:

$$-t(7) \not< s(7) - s(6) < t(6) \quad \text{or} \quad -9 \not< 10 - 19 < 5,$$

$$-t(8) \not< s(8) - s(6) < t(6) \quad \text{or} \quad -6 \not< 7 - 19 < 5,$$

$$-t(8) < s(8) - s(7) < t(8) \quad \text{or} \quad -6 < 7 - 10 < 6.$$

Therefore, a conflict exists between nodes (7) and (8) on machine 2. We then check the conflict between unscheduled nodes (9), (10) and (12) pertained to machine 3 as follows:

$$- t(10) < s(10) - s(9) \nless t(9) \quad \text{or} \quad - 4 < 15 - 6 \nless 2,$$

$$- t(12) < s(12) - s(9) \nless t(9) \quad \text{or} \quad - 2 < 13 - 6 \nless 2,$$

$$- t(12) \nless s(12) - s(10) < t(10) \quad \text{or} \quad - 2 \nless 13 - 15 < 4.$$

Therefore, no conflict exists between nodes on machine 3. Now we select a machine $m(k)$ such that

$$c(k) = \min c(\ell) \qquad \text{for } \ell = 1, 2, 3, 7, 8.$$

Since node (3) has the minimum completion time c(3) or 10, we select machine 1 on which the conflict will be resolved in favor of one of the nodes in the conflict set. Set L = L+1, or 3 and τ= c(3), or 10.

Step 3. Compute the lower bound for arcs ((1,2),(1,3)),((2,1), (2,3)) and ((3,1), (3,2)). We first compute the lower bound for arcs ((1,2), (1,3)). From the data pertained in the graph G((1,2)(1,3)) as shown in Figure 2.10, we have

$$\underline{B}((1,2),(1,3)) = \max\left\{\max\left\{\max\left\{\begin{array}{l} c(1) + [t(2) + t(3)] \\ \min\begin{Bmatrix} s(6) \\ s(7) \\ s(8)\end{Bmatrix} + [t(6) + t(7) + t(8)] \\ \min\begin{Bmatrix} s(10) \\ s(12)\end{Bmatrix} + [t(10) + t(12)]\end{array}\right\}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\max\left\{\begin{array}{l} 11 + (8 + 3) \\ \min\begin{Bmatrix} 23 \\ 12 \\ 7\end{Bmatrix} + (5 + 9 + 6) \\ \min\begin{Bmatrix} 19 \\ 13\end{Bmatrix} + (4 + 2)\end{array}\right\}\right\}, 28\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}22 \\ \max\left\{\begin{matrix}7 + 20 \\ \\ 13 + 6\end{matrix}\right\}\end{matrix}\right\}, 28\right\}$$

$$= \max\left(\max\,[22,\ 27],\ 28\right)$$

$$= \max\,[27,\ 28]$$

$$= 28.$$

Similarly, from the data pertianed in $G((2,1),(2,3))$, as shown in Figure 2.11, we have

$$\underline{B}((2,1),(2,3)) = \max\left\{\max\left\{\begin{matrix}c(2) + [t(1) + t(3)] \\ \max\left\{\begin{matrix}\min\left\{\begin{matrix}s(6)\\ s(7)\\ s(8)\end{matrix}\right\} + [t(6) + t(7) + t(8) \\ \\ \min\left\{\begin{matrix}s(10)\\ s(12)\end{matrix}\right\} + [t(10) + t(12)]\end{matrix}\right\}\end{matrix}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{matrix}15 + (3 + 3) \\ \max\left\{\begin{matrix}\min\left\{\begin{matrix}19\\ 18\\ 7\end{matrix}\right\} + (5+9+6) \\ \\ \min\left\{\begin{matrix}15\\ 13\end{matrix}\right\} + (4+2)\end{matrix}\right\}\end{matrix}\right\}, 27\right\}$$

$$= \max\left\{\max\left[\begin{matrix}21 \\ \max\left\{\begin{matrix}7 + 20 \\ \\ 13 + 6\end{matrix}\right\}\end{matrix}\right], 27\right\}$$

$$= \max\{\max\,[21,\ 27],\ 27\}$$

$$= \max\,[27,\ 27]$$

$$= 27.$$

We then compute the lower bound for arcs ((3,1),(3,2)) from the graph G((3,1)(3,2)) as shown in Figure 2.12 such that

$$\underline{B}((3,1),(3,2)) = \max\left\{\max\left[\begin{matrix}c(3) + [t(1) + t(2)] \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}s(6)\\ s(7)\\ s(8)\end{Bmatrix} + [t(6) + t(7) + t(8)] \\ \\ \min\begin{Bmatrix}s(10)\\ s(12)\end{Bmatrix} + [t(10) + t(12)]\end{matrix}\right\}\end{matrix}\right], s(13)\right\}$$

$$= \max\left\{\max\left[\begin{matrix}10 + (3 + 8) \\ \max\left\{\begin{matrix}\min\begin{Bmatrix}22\\ 10\\ 7\end{Bmatrix} + (5 + 9 + 6 \\ \\ \min\begin{Bmatrix}18\\ 13\end{Bmatrix} + (4 + 2)\end{matrix}\right\}\end{matrix}\right], 27\right\}$$

$$= \max\left\{\max\left\{\max\begin{bmatrix}21 & \\ & \max\begin{Bmatrix}7 + 20\\ \\ 13 + 6\end{Bmatrix}\end{bmatrix}, 27\right\}\right\}$$

$$= \max\{\max [21, 27], 27\}$$

$$= \max [27, 27]$$

$$= 27.$$

Step 4. Since $\underline{B}$ ((2,1), (2,3)) = $\underline{B}$((3,1),(3,2)) or 27, a tie exists. We therefore break the tie by using the dispatching rules. From G((4.1), (4,3), (4,3)), as shown in Figure 2.8, we find that the completion time $c(3) < c(2)$ *or* $10 < 15$, *we therefore choose the arcs* ((3,1), (3,2)).

Step 5. Generate the sequences for all nodes on machines from the data pertained in the graph G((3,1),(3,2)), as shown in Figure 2.12. The sequence S_m is obtained by following the dispatching rules such that

S_1 = {(4) (3) (1) (2)}

S_2 = {(5) (8) (7) (6)}

S_3 = {(11) (9) (12) (10)}

these sequences are associated with a set of sequenced arcs A_m such that

A_m = {(4,3),(3,1),(1,2); (5,8),(8,7),(7,6); (11,9),(9,12),(12,10)}

Similarly, the graph $\bar{G}((3,1), (3,2))$, as shown in Figure 2.13, is obtained by assigning A_m into $G(0,0)$. The upper bound is then computed such that

$$\bar{B}((3,1),(3,2)) = s'(13)$$

$$= 30.$$

Since $\bar{B}((3,1),(3,2)) = T(S)$ or $30 = 30$, we go to step 6.

Step 6. Since the lower bound $\underline{B}((3,1),(3,2)) < T(S)$ or $27 < 30$, we include the arcs $((3,1), (3,2))$ into the set of chosen arcs A^*. We therefore branch from the graph $G((3,1),(3,2))$ and go to step 2.

Step 2. Check for conflicts between nodes pertained to the same machine by the relation

$$- t(\ell) < s(\ell) - s(k) < t(k) \qquad \text{for} \qquad c(k) \geq 10 \quad \text{and} \quad c(\ell) \geq 10$$

We first check for the conflict between unscheduled nodes (1) and (2) on machine 1 as follows:

$$- t(2) < s(2) - s(1) < t(1) \qquad \text{or} \qquad -8 < 10 - 10 < 3.$$

Therefore, a conflict exists between nodes (1) and (2) on machine 1. We then check the conflict between unscheduled nodes (6), (7) and (8) on machine 2 as follows:

$$- t(7) \nless s(7) - s(6) < t(6) \qquad \text{or} \qquad - 9 \nless 10 - 22 < 5,$$

$$- t(8) \nless s(8) - s(6) < t(6) \qquad \text{or} \qquad - 6 \nless 7 - 22 < 5,$$

$$- t(8) < s(8) - s(7) < t(7) \qquad \text{or} \qquad - 6 < 7 - 10 < 9.$$

Therefore, a conflict exists between nodes (7) and (8) on machine 2. We then check the conflict between unscheduled nodes (10) and (12) on machine 3 as follows:

$$-\ t(12) \not< s(12) - s(10) < t(10) \quad \text{or} \quad -\ 2 \not< 13 - 18 < 4.$$

Therefore, no conflict exists between nodes on machine 3. Now we select a machine m(k) such that

$$c(k) = \min c(\ell) \qquad \text{for} \qquad \ell = 1, 2, 7, 8.$$

Since node (1) has the minimum completion time c(1) or 13, we select machine 1 on which the conflict will be resoved in favor of node (1) or node (2). Set L = L+1, or 4 and τ =c(11), or 13.

Step 3. We first compute the lower bound for arcs (1,2). From the data pertained in the graph G(1,2) as shown in Figure 2.14, we have

$$\underline{B}(1,2) = \max\left\{\max\left\{\begin{array}{l} c(1) + t(2) \\ \max\left\{\begin{array}{l} \min\begin{Bmatrix} s(7) \\ s(8) \end{Bmatrix} + [t(7) + t(8) \\ \min\begin{Bmatrix} s(10) \\ s(12) \end{Bmatrix} + [t(10) + t(12)] \end{array}\right\} \end{array}\right\}, s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{array}{l} 13 + 8 \\ \max\left\{\begin{array}{l} \min\begin{Bmatrix} 10 \\ 7 \end{Bmatrix} + (9 + 6) \\ \min\begin{Bmatrix} 21 \\ 13 \end{Bmatrix} + (4 + 2) \end{array}\right\} \end{array}\right\}, 30\right\}$$

$$= \max\left\{\max\left\{\begin{matrix} 21 \\ \max\left\{\begin{matrix} 7 + 15 \\ \\ 21 + 6 \end{matrix}\right\} \end{matrix}\right\}, 30\right\}$$

$$= \max\{\max\ [21,\ 27],\ 30\}$$

$$= \max\ [27,\ 30]$$

$$= 30.$$

We then compute the lower bound for arc (2, 1) from the graph G(2,1) as shown in Figure 2.15 such that

$$\underline{B}(2,1) = \max\left\{\max\left\{\begin{matrix} c(2) + t(1) \\ \max\left\{\begin{matrix} \min\begin{Bmatrix} s(7) \\ s(8) \end{Bmatrix} + [t(7) + t(8)] \\ \\ \min\begin{Bmatrix} s(10) \\ s(12) \end{Bmatrix} + [t(10) + t(12)] \end{matrix}\right\} \end{matrix}\right\},\ s(13)\right\}$$

$$= \max\left\{\max\left\{\begin{matrix} 18 + 3 \\ \max\left\{\begin{matrix} \min\begin{Bmatrix} 10 \\ 7 \end{Bmatrix} + (9 + 6) \\ \\ \min\begin{Bmatrix} 18 \\ 13 \end{Bmatrix} + (4 + 2) \end{matrix}\right\} \end{matrix}\right\},\ 27\right\}$$

$$= \max\left(\max\left[\begin{matrix}21 \\ \max\left[\begin{matrix}7 + 15 \\ 13 + 6\end{matrix}\right]\end{matrix}\right], 27\right)$$

$$= \max\left(\max\,[21, 22], 27\right)$$

$$= \max\,[22, 27]$$

$$= 27.$$

Step 4. We choose the arc (2,1) which has the minimum lower bound of 27.

Step 5. Generate the sequences for all nodes on machines from the data pertained in the graph G(2,1), as shown in Figure 2.9. The sequences S_m is obtained by following the dispatching rules such that

$S_1 = \{(4)\ (3)\ (2)\ (1)\}$

$S_2 = \{(5)\ (8)\ (7)\ (6)\}$

$S_3 = \{(11)\ (9)\ (12)\ (10)\}$

these sequences are associated with a set of sequenced arcs A_m such that

$A_m = \{(4,3), (3,2), (2,1); (5,8), (8,7), (7,6); (11,9), (9,12), (12,10)\}$

Similarly, the graph $\bar{G}(2,1)$, as shown in Figure 2.16, is obtained by assigning A_m into G(0,0). The upper bound is then computed such that

$\bar{B}(2,1) = s'(13)$

$= 27.$

As $\bar{B}(2,1) < T(S)$ or $27 < 30$, we set $T(S) = 27$ and $A(S) = A_m$.

Step 6. *Since the lower bound* $\underline{B}(2,1) = T(S)$ *or 27*, we backtrack.

Step 8. Backtrack along the search tree by setting L = L-1 or 3.

Step 9. Since the two unexplored nodes have lower bounds greater or equal to the schedule time T(S) or 27, we go to step 7.

Step 7. Backtrack along the search tree by removing the arc ((3,1) *(3,2)) from the set A*.*

Step 8. Set L = L-1 or 2.

At this point, the backtracking continues until level 1 is reached. Since all unexplored nodes have lower bounds greater or equal to the schedule time T(S) or 27. The computation is stopped at level 1 with *an optimal solution of 27 and the sequenced arcs A(S)* such that

$A(S) = \{(4,3),(3,2),(2,1);(5,8),(8,7),(7,6);\ (11,9),(9,12),(12,10)\}$

<u>2.3 Computational Algorithm</u>

The algorithm discussed above is stated below in formal steps:

Step 1. Initialize the scheduling tree

1.1 Set conflict level L = 0, transition time $\tau = 0$, and schedule time $T(S) = \infty$.

1.2 Construct the initial graph, including conjunctive arcs only, $G = (N,C)$.

1.3 Compute the starting time, $s(k)$ and the completion time, $c(k)$ for all nodes in the graph.

Step 2. Check for conflicts

$$-\ t(\ell) < s(\ell) - s(k) < t(k)$$

where

$$(k),\ (\ell) \ \varepsilon\ J_m$$

for $c(k) \geq \tau$, $c(\ell) \geq \tau$

2.1 If the above relation holds, a conflict exists betweein node (k) and node (ℓ), set $L = L+1$ and select a machine $m(k)$ such that

$$c(k) = \min_{(\ell)\varepsilon F_m} [c(\ell)]$$

then set $\tau = c(k)$ and go to step 3.

2.2 If the above relation does not hold, no conflict exists, and go to step 7.

Step 3. Compute the lower bound $\underline{B}(k',\ell')$ for (k'), (ℓ') ε F_m such that

$$\underline{B}(k',\ell') = \max \left\{ \max\left(c(k') + \sum_{\substack{(i)\varepsilon m(k') \\ (i)\neq(k')}} t(i),\ \max_{\substack{m(i) \\ m(i)\neq m(k')}} \left[\min_{(i)\varepsilon m(i)} s(i) + \sum_{(i)\varepsilon m(i)} t(i) \right] \right),\ s(n+1) \right\}$$

for all (i) such that $c(i) \geq \tau$

Step 4. Select arc(s) $(k*,\ell*)$ having the minimum lower bound. If a tie exists, check the completion time of nodes $(k*)$ and $(\ell*)$.

4.1 If $c(k^*) < c(\ell^*)$, select arc (k^*,ℓ^*).

4.2 If $s(k^*) < s(\ell^*)$, select arc (k^*,ℓ^*).

4.3 If $(k^*) < (\ell^*)$, select arc (k^*,ℓ^*).

Step 5. Compute the upper bound $\bar{B}(k^*,\ell^*)$ by generating a set of sequenced arcs A_m, then check it with the schedule time $T(S)$. If $\bar{B}(k^*,\ell^*) < T(S)$, set $T(S) = \bar{B}(k^*,\ell^*)$ and $A(S) = A_m$.

Step 6. Compare the minimum lower bound with the current schedule time

6.1 If $\underline{B}(k^*,\ell^*) \geq T(S)$, go to step 7.

6.2 If $\underline{B}(k^*,\ell^*) < T(S)$, include the arc(s) (k^*,ℓ^*) in the set of chosen arcs A^* and branch from the graph $G(k^*,\ell^*)$, go to step 2.

Step 7. Remove the arc (k,ℓ) from the set of chosen arcs A^*.

Step 8. Backtrack by setting $L = L-1$

8.1 If $L>1$, go to step 9.

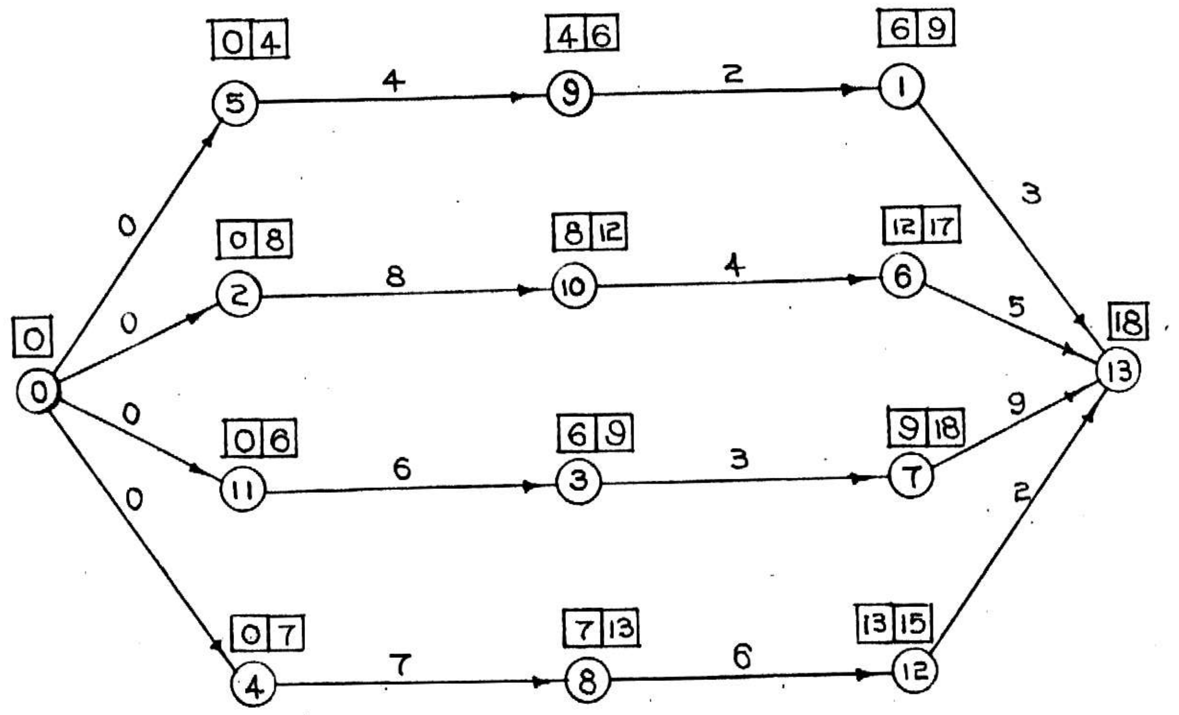
8.2 If $L=1$, $T(S)$ is an optimal schedule time with a set of sequenced arcs $A(S)$.

Step 9. Compare the lower bound for unexplored node(s) at conflict level in the scheduling tree

9.1 If $\underline{B}(k,\ell) \geq T(S)$, terminate the branch and go to step 7.

9.2 If $\underline{B}(k,\ell) < T(S)$, go to step 4.

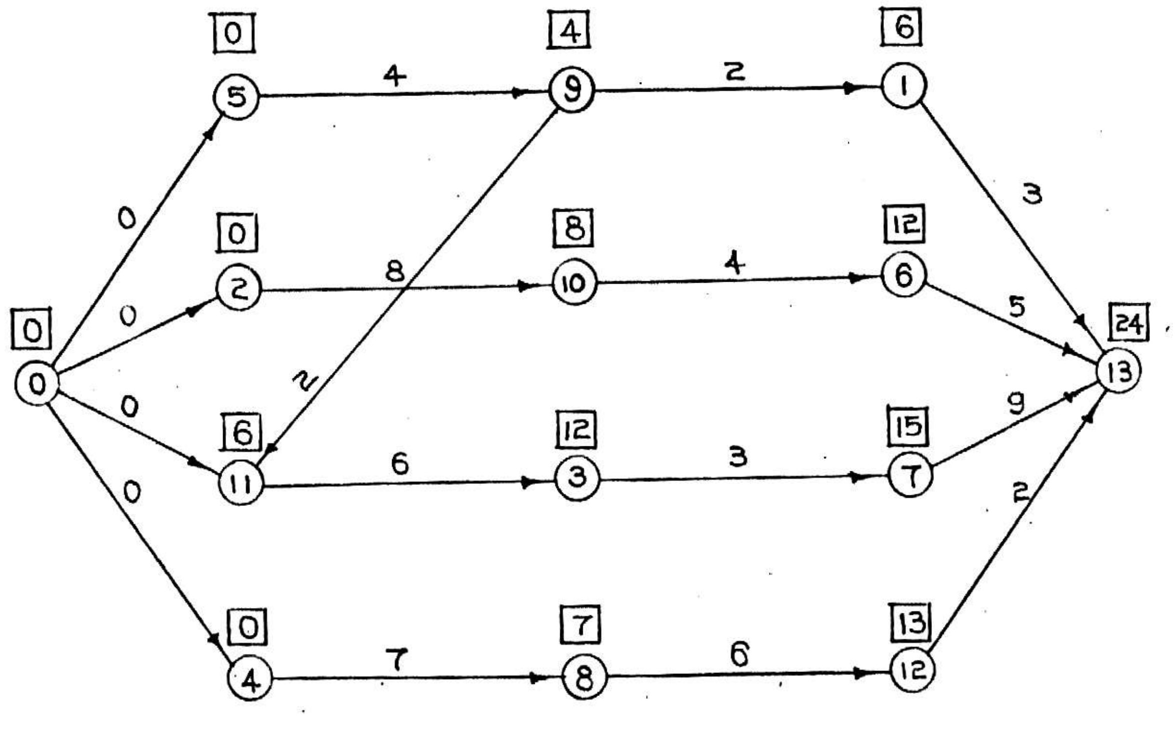
In order to follow the steps easily, a flow chart of the algorithm is shown in Figure 2.18.



G(0,0)

Note: The number in two boxes for each node (k) are from left to right starting time $s(k)$ and completion time $c(k)$.

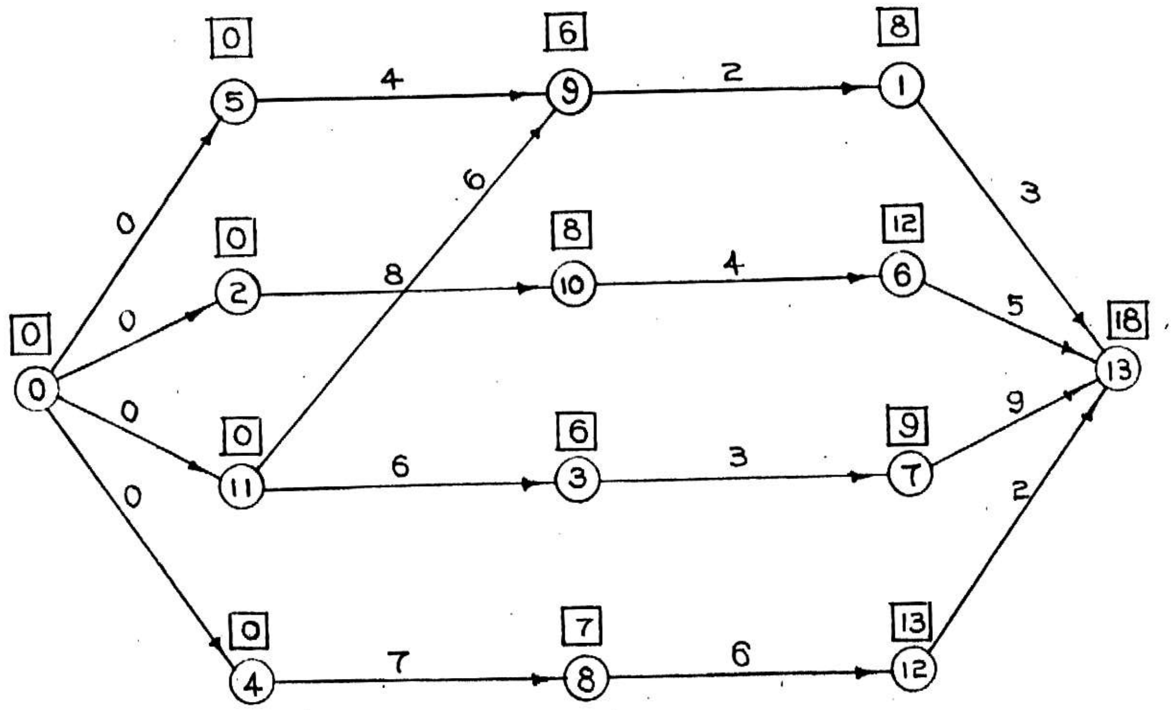
Figure 2.1 The Initial Graph of Sample Problem



G(9,11)

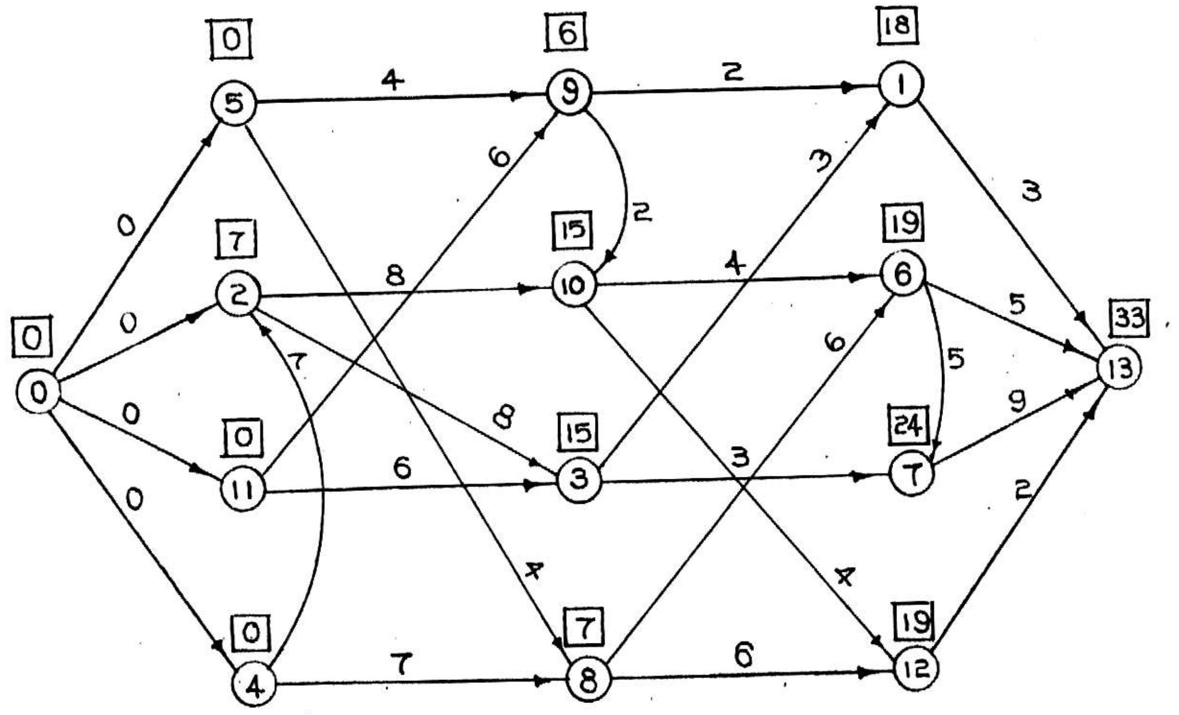
Note: The number in box for each node (k) is starting time s(k).

Figure 2.2 A Graph Depicting the Resolution in Favor of Node (9) at Level 1



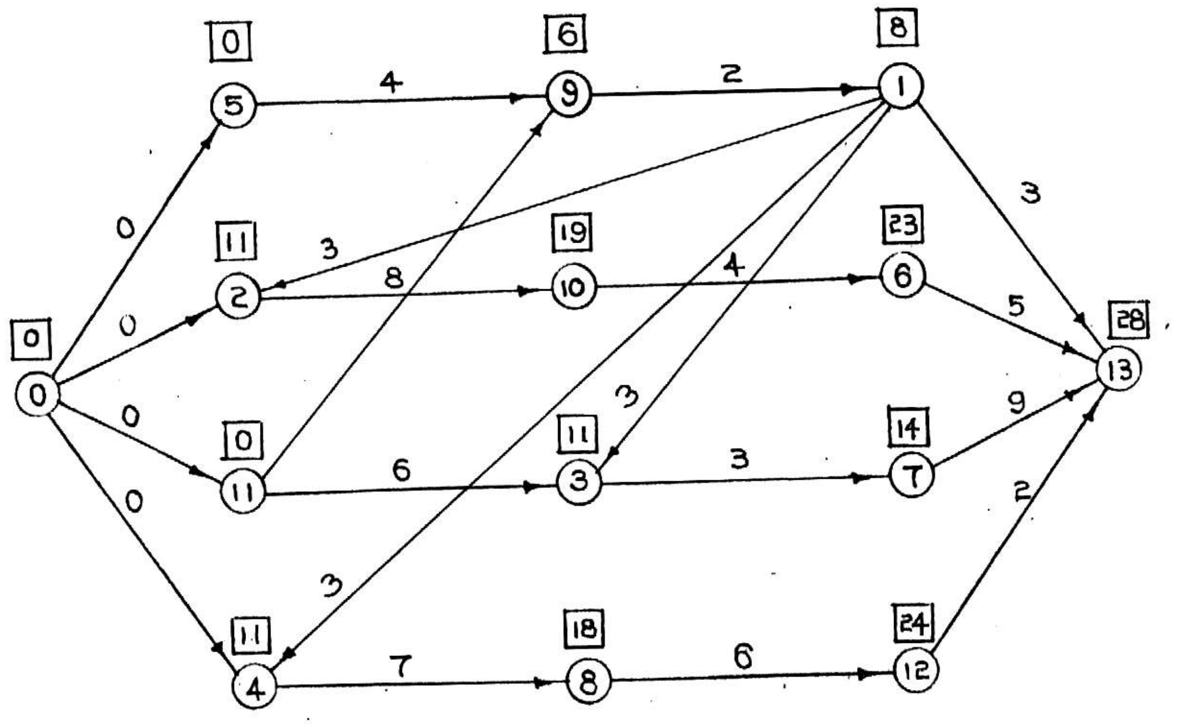
G(11,9)

Figure 2.3 A Graph Depicting the Resolution in Favor of Node (11) at Level 1



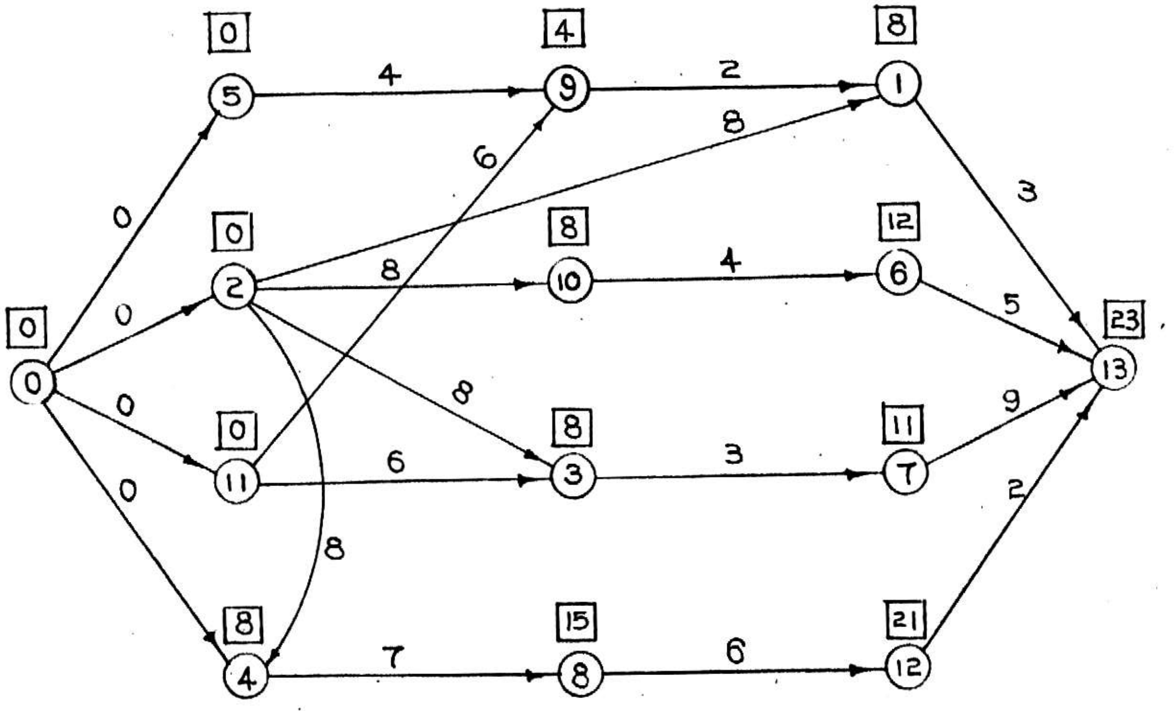
$\bar{G}$ (11,9)

Figure 2.4 A Graph Depicting a Feasible Solution at Level 1



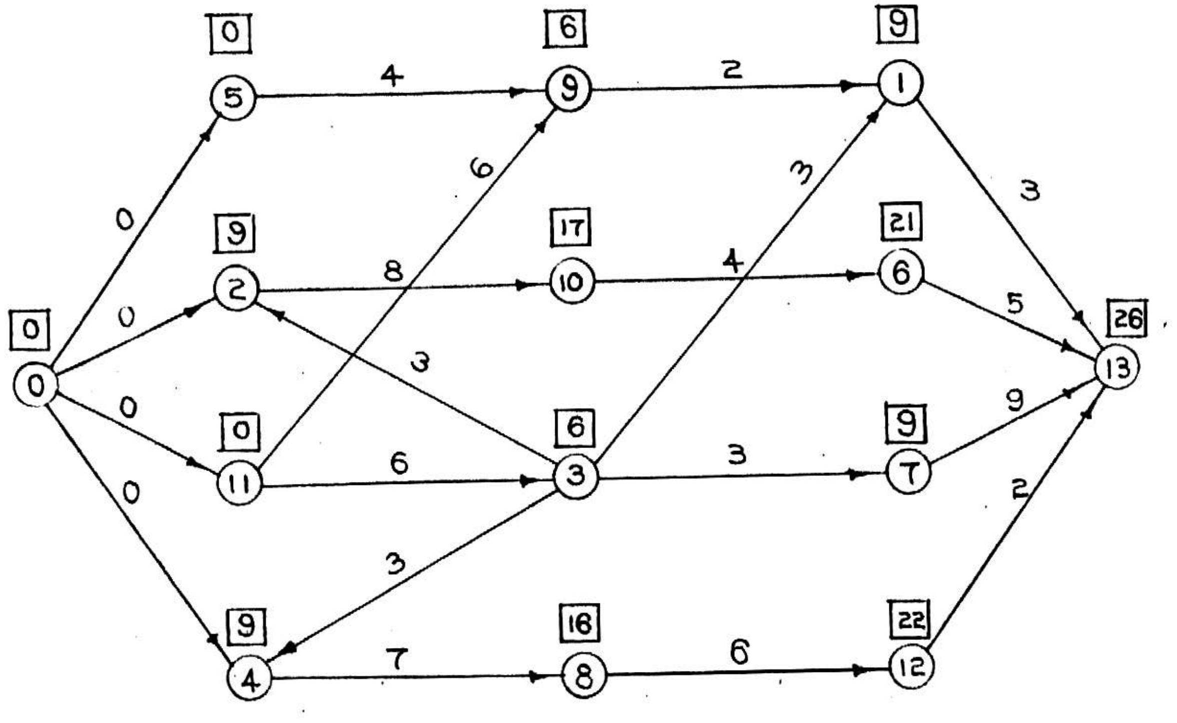
G((1,2),(1,3),(1,4))

Figure 2.5 A Graph Depicting the Resolution in Favor of Node (1) at Level 2



G((2,1),(2,3),(2,4))

Figure 2.6 A Graph Depicting the Resolution in Favor of Node (2) at Level 2



G((3,1),(3,2),(3,4))

Figure 2.7 A Graph Depicting the Resolution in Favor of Node (3) at Level 2

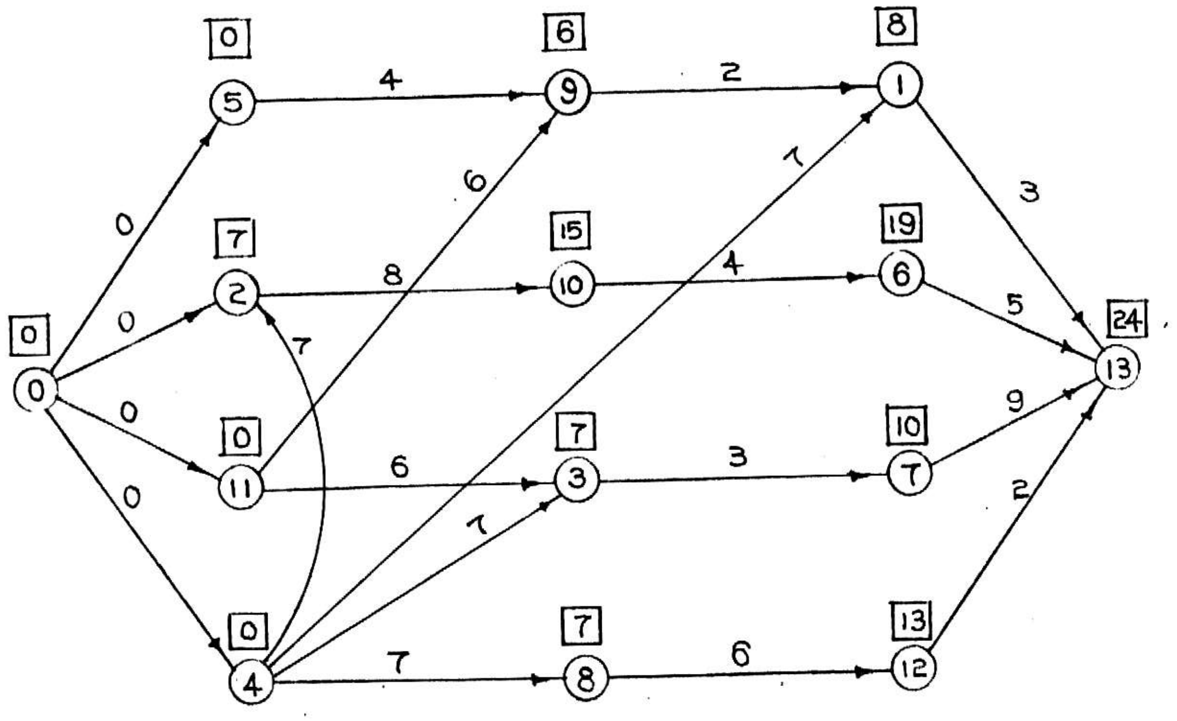
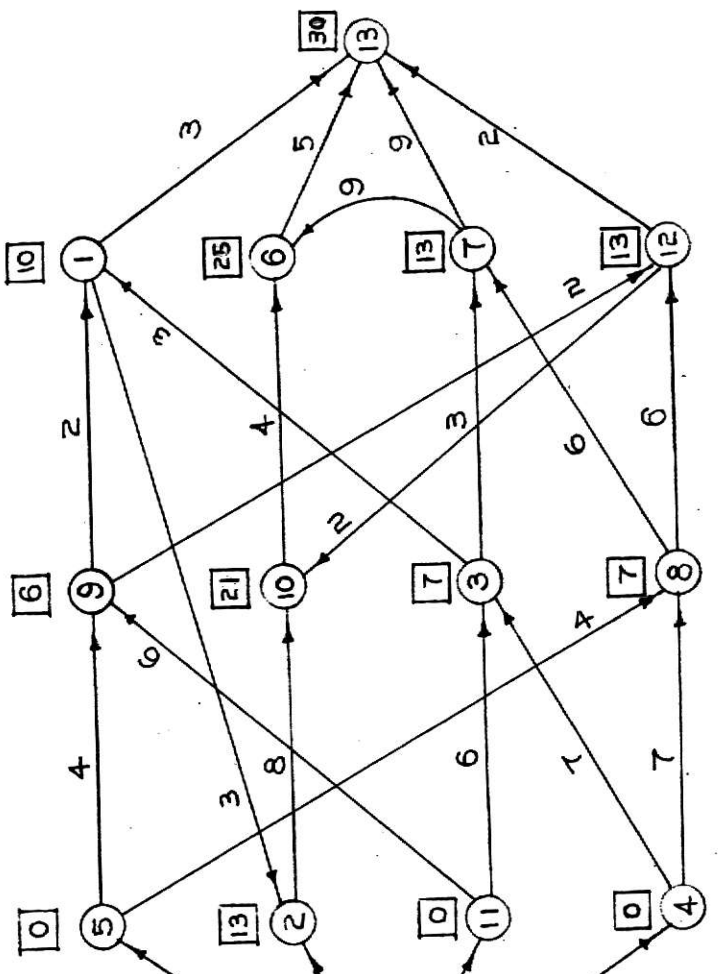
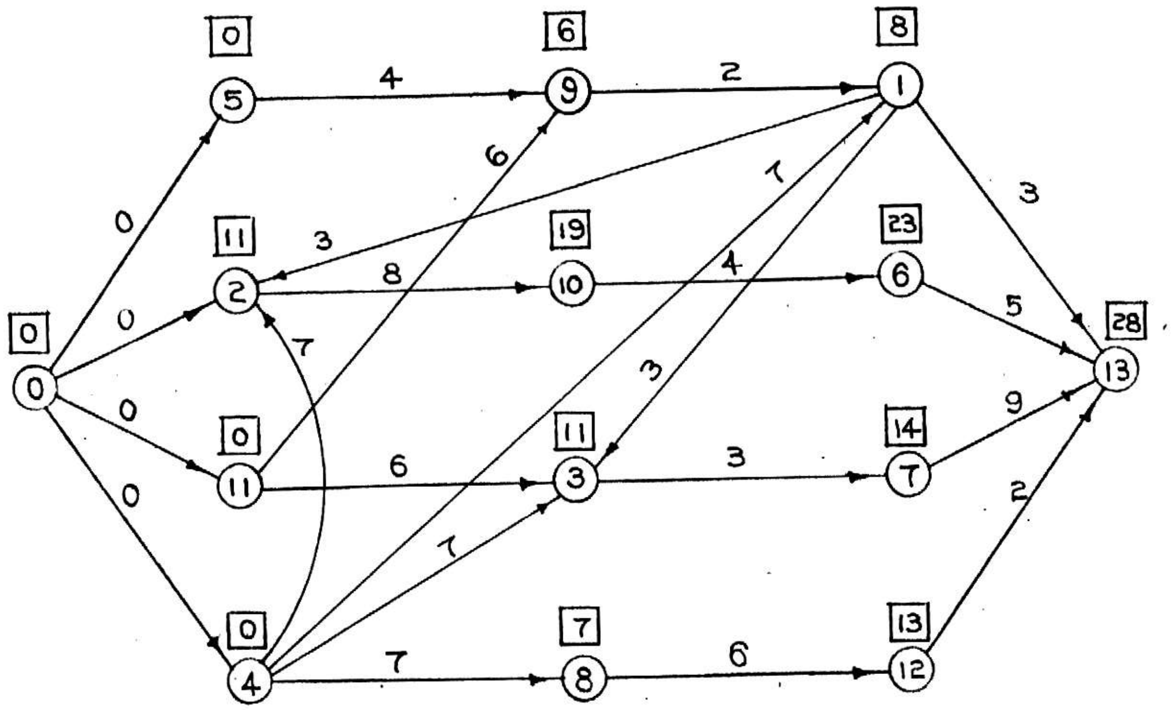

G((4,1),(4,2),(4,3))

Figure 2.8 A Graph Depicting the Resolution in Favor of Node (4) at Level 2



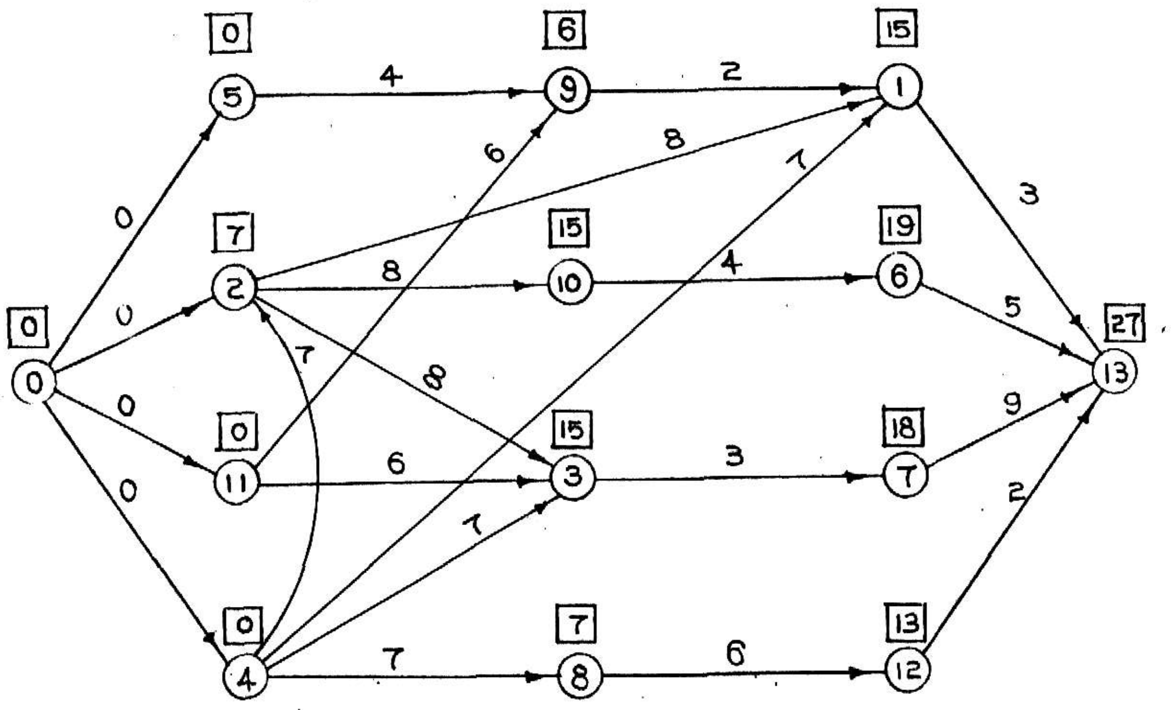
$\bar{G}((4,1),(4,2),(4,3))$

Figure 2.9 A Graph Depicting a Feasible Solution at Level 2



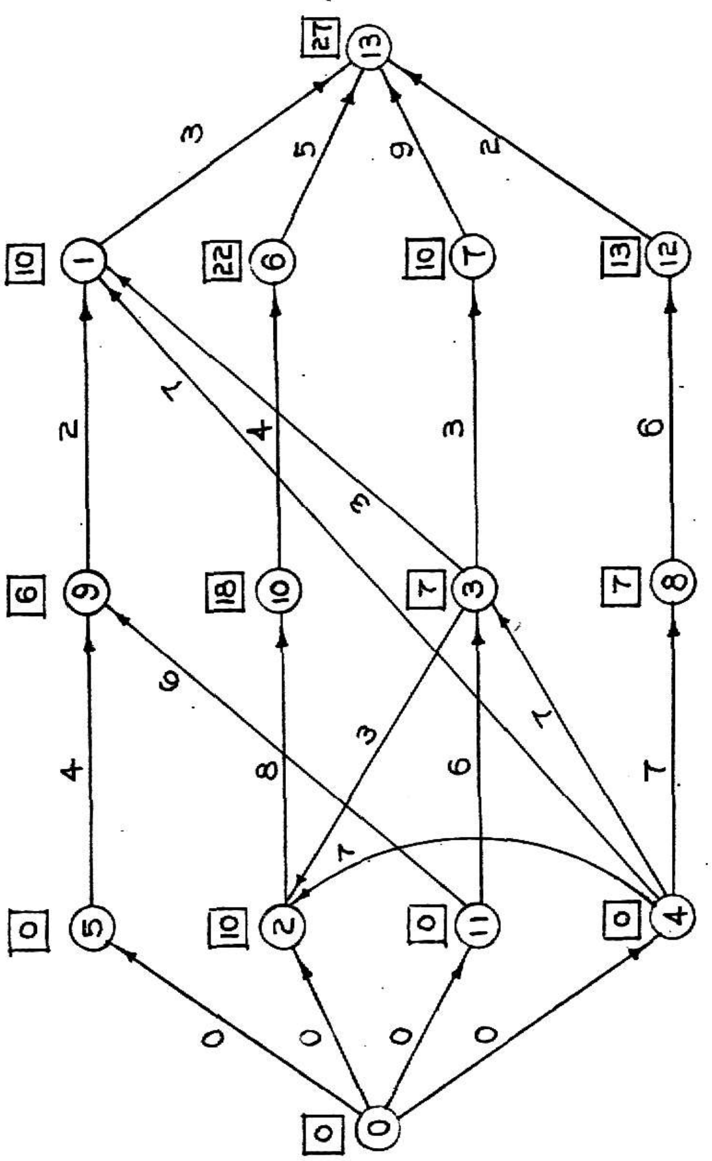
G((1,2),(1,3))

Figure 2.10 A Graph Depicting the Resolution in Favor of Node (1) at Level 3



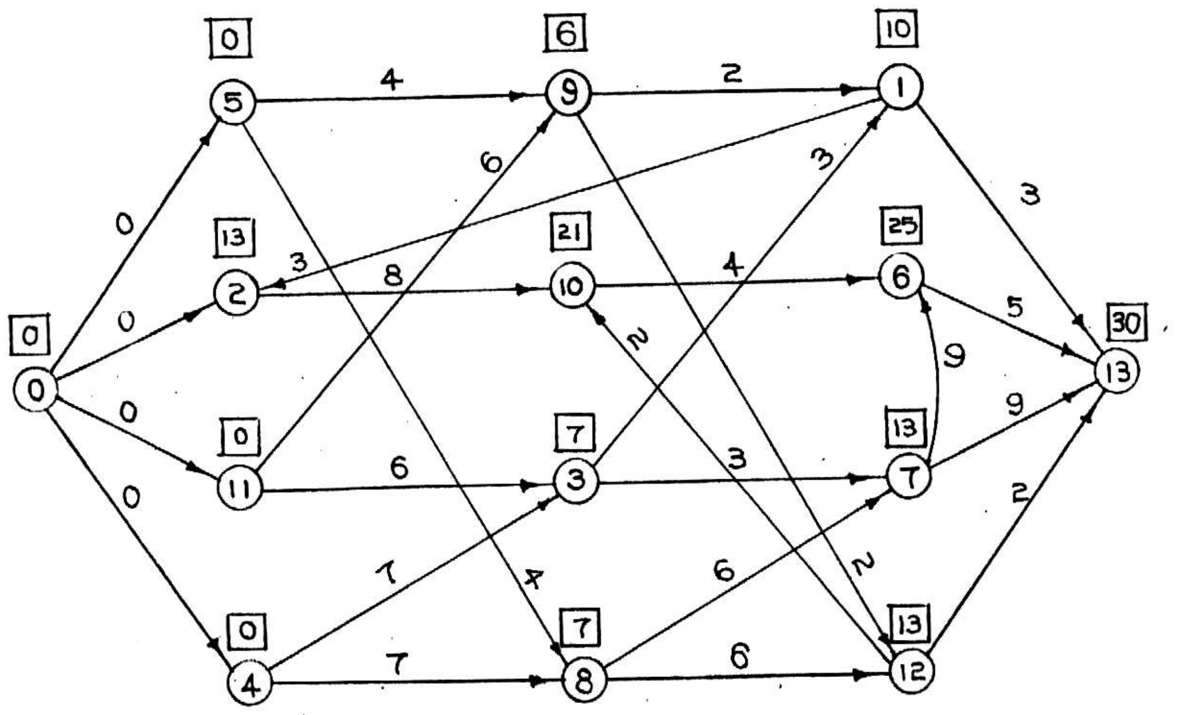
G((2,1),(2,3))

Figure 2.11 A Graph Depicting the Resolution in Favor of Node (2) at Level 3



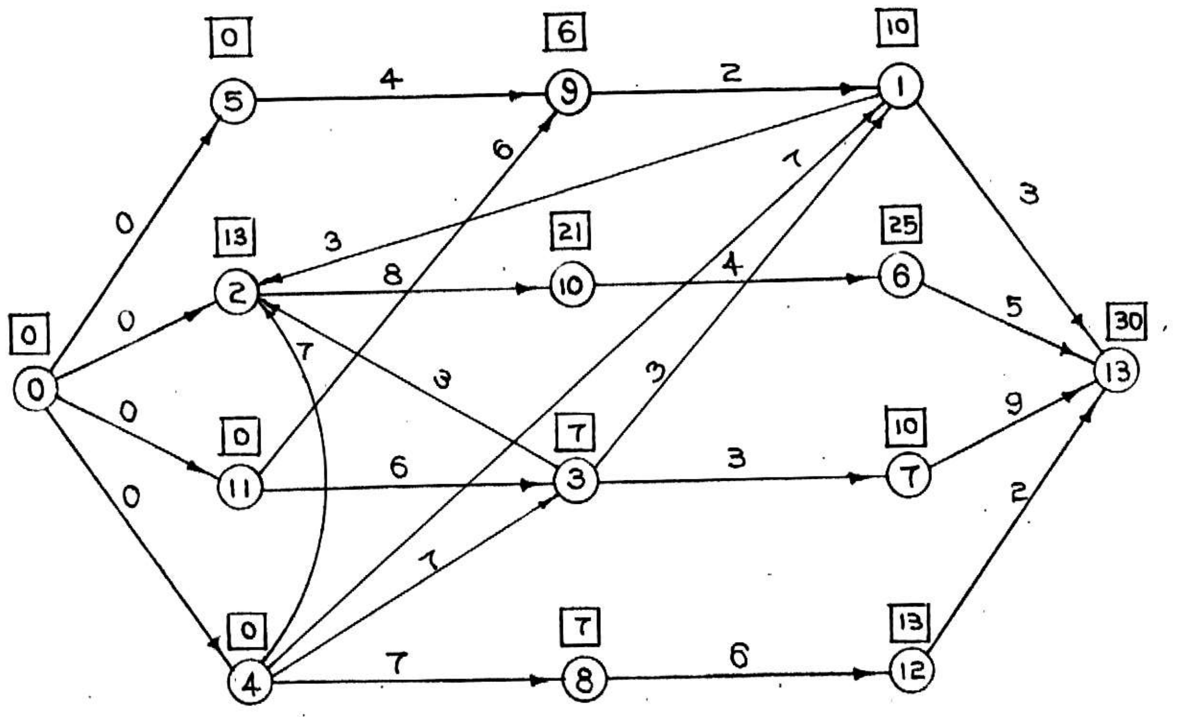
G((3,1),(3,2))

Figure 2.12 A Graph Depicting the Resolution in Favor of Node (3) at Level 3



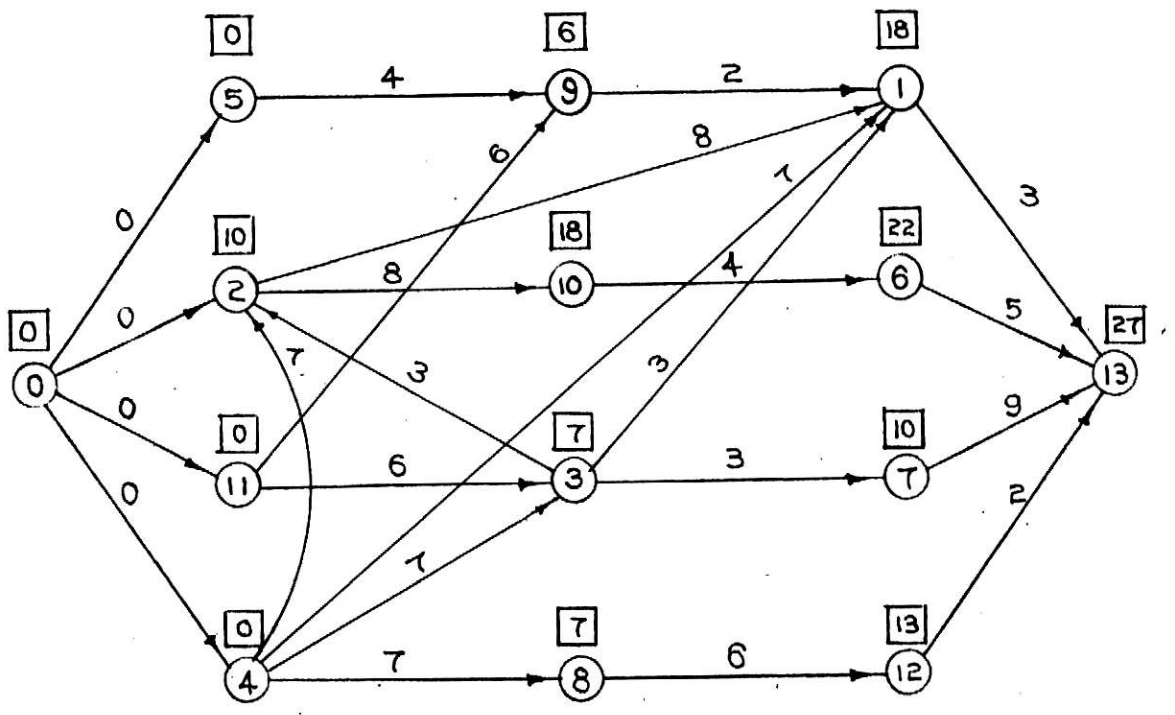
$\bar{G}((3,1),(3,2))$

Figure 2.13 A Graph Depicting a Feasible Solution at Level 3



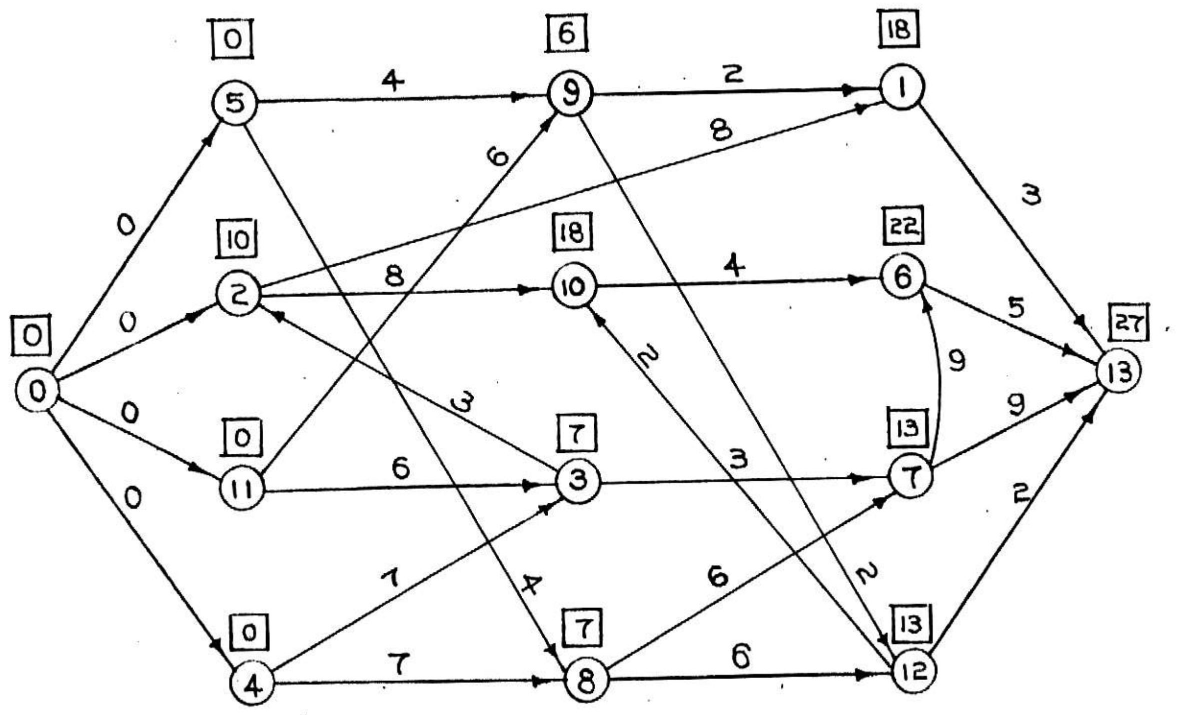
G(1,2)

Figure 2.14 A Graph Depicting the Resolution in Favor of Node (1) at Level 4



G(2,1)

Figure 2.15 A Graph Depicting the Resolution in Favor of Node (2) at Level 4



$\bar{G}(2,1)$

Figure 2.16 A Graph Depicting an Optimal Solution

Table 2.1 Solution of the Sample Problem

Back-tracking i	Conflict Level L	Step	Trans-ition Time τ	Conflict Set F_m	Node $G(k',\ell')$	Lower Bound $\underline{B}(k',\ell')$	Completion Time $c(k')$	Starting Time $s(k')$	Index Number (k')	Minimum Lower Bound $\underline{B}(k^*,\ell^*)$	Set of Chosen Arcs A^*	Upper Bound $\bar{B}(k^*,\ell^*)$	Schedule Time $T(S)$
	0	1	0	ϕ							ϕ		∞
	1	2	6	{(9),(11)}									
		3			G(9,11)	27	6	4	-				
		4			G(11,9)	27	6	0	-	27			
		5										33	33
		6									{(11,9)}		
	2	2	7	{(1),(2),(3),(4)}									
		3			G((1,2),(1,3),(1,4))	34	-	-	-				
					G((2,1),(2,3),(2,4))	31	-	-	-				
					G((3,1),(3,2),(3,4))	29	-	-	-				
		4			G((4,1),(4,2),(4,3))	27	-	-	-	27			
		5										36	36
		6									{(11,9),(4,1),(4,2),(4,3)}		
	3	2	10	{(1),(2),(3)}									
		3			G((1,2),(1,3))	28	-	-	-				
					G((2,1),(2,3))	27	15	-	-				
		4			G((3,1),(3,2))	27	10	-	-	27			
		5										30	30
		6									{(11,9),(4,1),(4,2),(4,3),(3,1),(3,2)}		
	4	2	13	{(1),(2)}									
		3			G(1,2)	30	-	-	-				
		4			G(2,1)	27	-	-	-	27			
		5										27	27[†]
	3	8											
		7									{(11,9),(4,1),(4,2)(4,3)}		
	2	8											
		7									{(11,9)}		
0	1	8											
		7									ϕ		

[†]An optimal solution

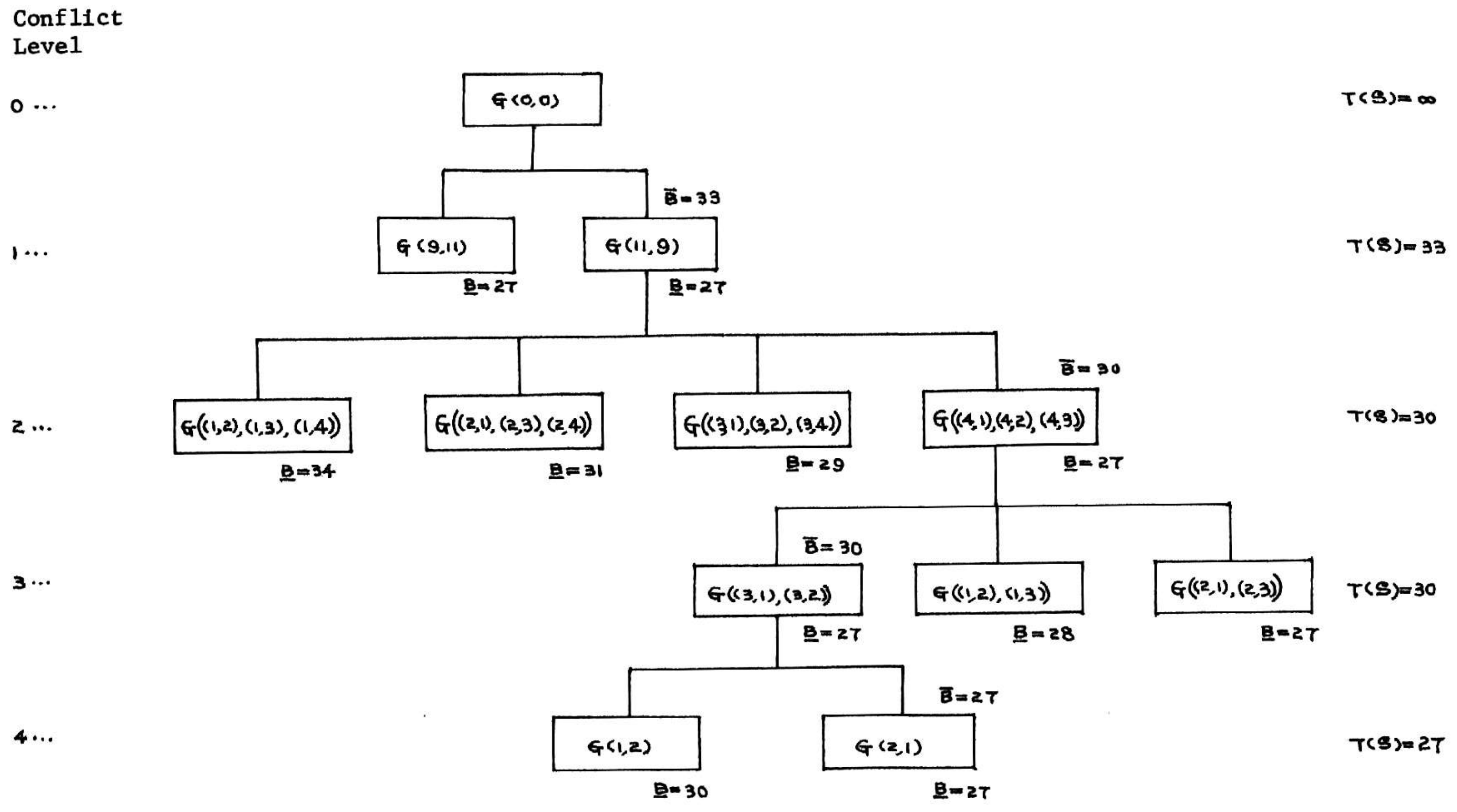


Figure 2.17 The Scheduling Tree of the Sample Problem

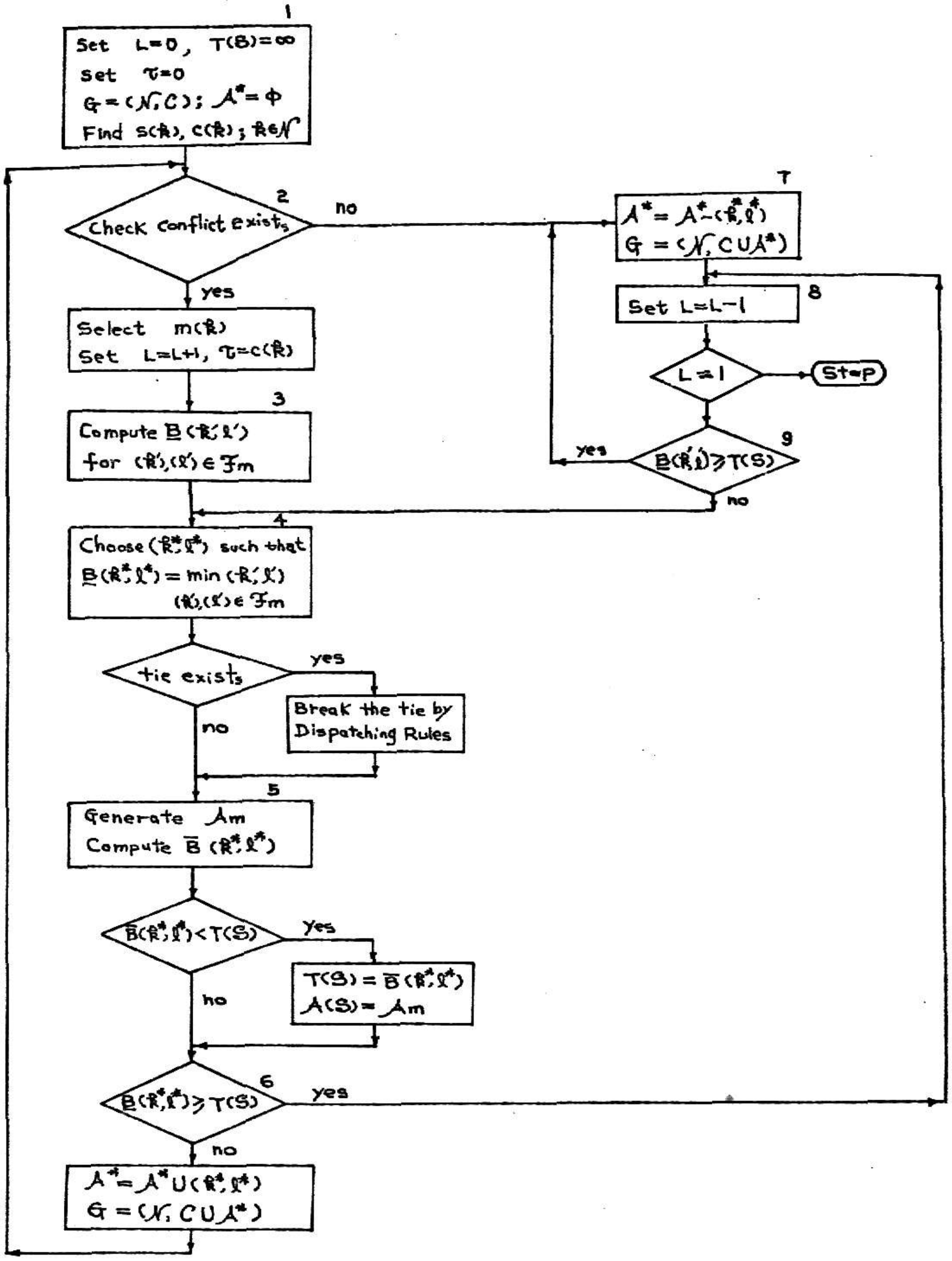


Figure 2.18 A Flow Chart for Disjunctive Graph Algorithm

CHAPTER III

COMPARATIVE EVALUATION

In this chapter, ten published shop scheduling sample problems are solved by five existing methods. These methods have been proposed seperately by Brooks and White [6], Balas [4], Ashour and Hiremath [2], Schrage [14] and Charlton and Death [7]. The results obtained by these methods are comparatively evaluated with that by the proposed method. *The sizes of the problems vary between 2 to 5 jobs and 2 to 4 machines.* The criterion for this evaluation is based on the number of nodes explored to prove optimality.

The results obtained for solving these problems by various techniques are summarized in Table 3.1. The first problem is presented in Balas [3]. His method requires exploration of 12 nodes. In his method, one has to compute the lower bound and upper bound for each graph. Then, one has to evaluate the candidate arcs on the critical path to guide the search. Charlton and Death's method requires much less computational effort than that by Balas. The reason is that their method employs a heuristic rule to guide the search. Lower bound is used only when one backtracks. Their method requires the exploration of 7 nodes to obtain optimal solution to the same problem. Both Brooks and White and Ashour and Hiremath methods require the setting of 3 conflict levels and the exploration of 7 nodes to prove the optimality. However, by the proposed method, it only requires two conflict level and 5 nodes. This improvement is achieved by applying an upper bound in conjunction with a lower bound at each level. Therefore, the optimal solution could be recognized before all the conflicts had been resolved. Schrage's method requires 6 iterations to prove optimality. An iteration is defined to be the scheduling of one activity (operation). This happens once at each conflict level at which two or more nodes are explored.

The second problem is also presented in Balas [3]. By his method, the computation is not terminated after 149 nodes had been explored. Charlton and Death's method require the exploration of 13 nodes. Schrage's method requires at least 40 iterations. Brooks and White's and Ashour and Hiremath methods require the exploration of 77 and 14 nodes, respectively. However, the proposed method only requires 13 nodes. The third problem is presented in Charlton and Death [7]. Schrage's method requires 7 iteration to solve the problem. While, all the other methods require 4 nodes, the proposed method requires 2 nodes only.

The fourth problem is presented in Brooks and White [6]. Both their method and Ashour and Hiremath's method require the exploration of 17 nodes. However, in Balas method, the computation is not terminated after 40 nodes have been explored. This fact may be attributed to the definition of the lower bound [4]. In this procedure, one has to explore more nodes to prove optimality. Schrage's method requires 23 iterations; however, the solution obtained is not optimal. Charlton and Death's method requires the exploration of 20 nodes. While, the proposed method requires only 13 nodes to be explored. For problems 5 there is no significant difference among the results obtained by the different techniques.

Problem 6 is our sample problem appearing in Chapter 2. By Balas' method, the graph of this problem is too complicated to solve by manual calculation. As mentioned earlier, this method requires a solution of a critical path problem involving a complete set of disjunctive arcs in each graph. Whenever, a job-shop problem has a moderate size, the graph which displays this problem would be very complicated. In Charlton and Death's method, it was not terminated after exploring 41 nodes. This fact may be explained by two reasons. First, their method is based on the concept of resolving the conflicts among nodes in a graph. At each stage, a pair of arcs is chosen and its direction is decided in favor of a particular node in the conflict set. However, when there are more than two nodes in conflict on the same machine, chosing an arc in favor of a particular node may not necessarily mean that the conflict has been resolved for that node. Therefore, it may be necessary to resolve the conflict in favor of that node again in the next stage. In such cases, some of the schedules may be generated several times. Second, the lower bound used in his method is defined as the length of the critical path. As a result, it is found that the value of this bound may not always be high enough to prove optimality when one backtracks. Consequently, more nodes need to be explored. Brooks and White's method requires the exploration of 52 nodes for solving this problem. While Ashour and Hiremath's method requires only 21 nodes. This improvement is achieved by implementing a more powerful bounding procedure. The proposed method assists in improving the bounding procedure by imbedding heuristic rules to break the tie between lower bounds. As observed, the number of nodes explored is reduced almost to half of that by Ashour and Hiremath's method.

Problem 7 to 10 are illustrative examples presented in Greenberg [10]. Problem 7, 8 and 9 which are solved by various techniques indicates no significant difference in the number of nodes explored, but are presented here for purpose of comparison. The tenth problem, Greenberg's method required the solution of 19 linear programming problems. Charlton and Death's method require the exploration of 11 nodes. Balas' method require 5 nodes. Both Brooks and White's method and Ashour and Hiremath's method require 6 nodes respectively. However, the method proposed here only requires the exploration of 3 nodes to prove optimality.

Table 3.1 Comparative Evaluation of Various Techniques for Solving Published Sample Problems

No.	Job Set	References	Optimal Solution	Number of Nodes Explored					
				Brooks and White Method (1965)	Balas Method (1968)	Ashour and Hiremath Method (1969)	Schrage[††] Method (1970)	Charlton and Death Method (1970)	Proposed Method
1	(3x2) Flow-Shop	Balas (1967)	31	7	12	7	6	7	5
2	(5x4) Job-Shop[†]	Balas (1967)	13	77	>149*	14	>40*	13	9
3	(2x3) Job-Shop	Charlton and Death (1970)	16	4	4	4	7	4	2
4	(3x4) Flow-Shop	Brooks and White (1965)	32	17	>40*	17	23***	20	13
5	(2x3) Job-Shop	Ashour (1967)	13	4	3	4	6	4	2
6	(4x3) Job-Shop	Sample Problem	27	52	**	21	20	>41*	11
7	(2x2) Flow-Shop	Greenberg (1968)	10	4	2	4	4	4	2
8	(2x2) Job-Shop	Greenberg (1968)	9	2	2	2	4	2	2
9	(2x3) Job-Shop	Greenberg (1968)	14	2	3	2	6	2	2
10	(3x2) Job-Shop	Greenberg (1968)	8	6	5	6	13	11	3

[†] This problem consists of 13 operations.

[††] This is number of iterations. An iteration is defined to be the scheduling of one activity (operation). This happens once at each conflict level at which two or more nodes are explored.

* The procedure is not terminated after exploring certain number of nodes.

** The graph of this problem is complicated to solve by hand using Balas Method.

*** The solution found is 34 which is not optimal.

CHAPTER IV

SUMMARY AND CONCLUSIONS

The basic objective of this report is to develop an implicit enumerative technique for solving shop scheduling problems. A shop scheduling problem can be formulated as a two-terminal network flow problem for which the disjunctive constraints are binding. These constraints arise, for instance, when both operations require the same machine during the same period. The enumerative task in this algorithm is to successively resolve the conflict among operations which violate the disjunctive constraints. Resolving the conflict in favor of a particular operation is called a possible resolution. Graphically, each possible resolution is equivalent to assigning an arc (or arcs) from that particular node (operation) to the other(s) in the conflict set. Consequently, graphs associated with the possible resolutions are generated. In order to obtain an optimal solution without explicitly enumerating all possible resolutions, three main processes are included in this algorithm. These are: (1) branching process which generate a new set of graphs (or possible resolutions) from a previously selected graph; (2) bounding process which helps one select a graph for further branching, and thus makes it possible to achieve reduction in generating graphs at each level; (3) backtracking process which tests the solution for optimality.

In Chapter I, the various existing enumerative techniques are comprehensively reviewed with reference to their basic concepts, computational experience, advantages and disadvantages. In Chapter II, the basic concept of the proposed algorithm is presented. The computational algorithm for the proposed method is summarized in formal steps. The sample problem is solved to illustrate the computational algorithm. In Chapter III, ten published shop scheduling sample problems have been solved by five existing techniques. The results obtained by these methods have been comparatively evaluated with that by the proposed method. The sizes of these problems vary between 2 to 5 jobs and 2 to 4 machines. The criterion for this evaluation is based on the number of nodes explored to prove optimality. The results show that the proposed method requires relatively smaller number of nodes explored than that by the other techniques. This fact may be attributed to the better quality of the proposed bounding procedure. In this bounding procedure, heuristic rules are applied to assist the lower bound in guiding the search when a tie exists between lower bounds. In addition, an upper bound is used in conjunction with the lower bound at each level and therefore, an optimal solution can be recognized before all the conflicts have been resolved. The other features of this algorithm worth mentioning are as follows:

1. The conflicts need only to be checked for the unscheduled nodes by a simplified constraint equation at each level. Thus, the more nodes that have been scheduled, the less the computational effort required to check for conflicts.

2. The storage requirements are limited to the data pertained to the current nodes in the search tree. All the lower-bounds of possible resolutions can be computed by using the same storage space for a graph.

3. By the virtue of a graph, it permits one to use an efficient critical path method.

4. An actual solution is obtained at each level, this algorithm permits one to terminate the search at anytime with a good, but not necessarily optimal solution.

5. Combining the above four features, problems of larger size could be solved within a reasonable time.

In conclusion, it is suggested that more computational experience is necessary before more reliable results can be obtained. Further investigations which would be recommended are as follows:

1. The number of nodes explored to prove optimality using upper bounds versus not using upper bounds.

2. The quality of the solution obtained without backtracking using the heuristic rules when the lower bound fails in guiding the search.

3. The computational time required to obtain an optimal solution for all above cases.

To generalize this algorithm by imbedding more constraints such as due date and concurrency requirements (certain operations must be performed concurrently) is also recommended.

REFERENCES

[1] Ashour, S. "A Branch-and-Bound Algorithm for Flow-Shop Scheduling Problems," AIIE Transactions, Vol. II, No. 2, June, 1970.

[2] Ashour, S. and S.R. Hiremath, "Branch-and-Bound Approach for Job-Shop Scheduling Problems," presented before the 37th National Meeting of the Operations Research Society of America, Washington, D.C., April 20-22, 1970.

[3] Balas, E., "Discrete Programming by the Filter Method," Operations Research, Vol. 15, 1967, pp. 915-957.

[4] Balas, E., "Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm," Operations Research, Vol. 17, No. 6, 1968, pp. 941-956.

[5] Balas, E., "Machine Sequencing: Disjunctive Graphs and Degree-Constrained Subgraphs, "IBM, New York Scientific Center Report No. 320-2971, April 1, 1969.

[6] Brooks, G.H. and C. White, "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem, "The Journal of Industrial Engineering, Vol. 16, No. 1, 1965, pp. 34-40.

[7] Charlton, J. M. and C. C. Death, "A Generalized Machine - Scheduling Algorithm, "Operations Research Quarterly, Vol. 21, No. 1, 1970, pp. 127-134.

[8] Charlton, J. M. and C. C. Death, "A Method of Solution for General Machine-Scheduling Problems, "Operations Research, Vol. 18, No. 4, 1970, pp. 689-706.

[9] Giffler, B. and G. Thompson, "Algorithms for Solving Production Scheduling Problems," Operations Research, Vol. 8, No. 5, 1960, pp. 487-503.

[10] Greengberg, H. H., "A Branch-and-Bound Solution to the General Scheduling Problem," Operations Research, Vol. 16, No. 2, 1968, pp 353-36.

[11] Land, A. H. and A. Doig, "An Automatic Method of Solving Discrete Programming Problems, "Econometrica, Vol. 28, No. 3, 1960, pp. 497-520.

[12] Little, J. D. C., K. G. Morty, D. W. Sweeney and C. Karel, "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, No. 6, 1963, pp. 972-989.

[13] Moth, J. F. and G. L. Thompson, Industrial Scheduling, Prentice-Hall, Englewood Cliffs, N.J. 1963.

[14] Schrage, L., "Solving Resource - Constrainted Network by Implicit Enumeration - Nonpreemptive Case, "Operations Research, Vol. 18, No. 2, 1970, pp. 263-278.

A DISJUNCTIVE GRAPH TECHNIQUE FOR SHOP SCHEDULING PROBLEMS

by

KUNG-YING CHIU

B.E. (I.E.), Tunghai University

Taichung, Taiwan, Republic of China, 1968

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1971

The shop scheduling problem is formulated as a two-terminal network flow problem for which the disjunctive constraints are binding. These constraints arise when both operations require the same machine during the same period. An enumerative technique is proposed to solve the problem by successively resolving the conflicts among those operations which violate the disjunctive constraints. Graphical interpretation of resolving the conflict in favor of a node (operation) is equivalent to assigning an arc (arcs) from that node to the other(s) in conflict. Consequently, each possible resolution is represented as a graph. A branch-and-bound procedure is included in this disjunctive graph algorithm, so that an optimal solution can be obtained without explicitly enumerate all possible resolutions.

This algorithm utilizes some heuristic rules to guide the search when a tie exists between lower bounds. An upper bound is also used in conjunction with lower bounds to detect an optimal solution before all the conflicts have been resolved. An actual solution is obtained at each level, thus one can terminate the search at any time with a good, but not necessarily optimal, solution. Storage requirement are mostly limited to the data pertained to a graph.

Ten published shop scheduling sample problems have been solved by five existing techniques. The results obtained by these methods have been comparatively evaluated with that by the proposed method. The criterion for this evaluation is based on the number of nodes explored to prove optimality. The results show that the proposed method requires relatively smaller number of nodes explored than that by the other techniuqes.