# PROBABILISTIC MODELLING OF BRANCH AND BOUND ALGORITHMS

Benjamin W. Wah and Chee Fen Yu

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

**Abstract**

Branch and bound algorithms are organized and intelligently structured searches of solutions for enumerative-type problems such as NP-complete problems. In this paper, we propose a probabilistic model of branch and bound algorithms with best first search. We have (1) estimated the total virtual memory space requirement; and (2) predicted the number of subproblems evaluated before the process terminates. The model is useful for designing virtual memory support of branch and bound algorithms. It is also important to justify that approximate branch and bound algorithms can be very effective in reducing the total number of iterations.

## 1. Introduction

A branch and bound algorithm is an efficient algorithm to solve for problems that are put into the form of a constrained optimization.

$$\begin{array}{lll} \text{Minimize} & C_0(x) & \\ \text{subject to} & g_1(x) \geq 0 & \\ & g_2(x) \geq 0 & \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ & g_m(x) \geq 0 & \\ \text{and} & x \in X & \end{array}$$

in which X represents the domain of optimization defined by the m constraints, normally an euclidean n-space, and x denotes a vector $(x_1, x_2, \ldots, x_n)$. (Problems that are NP-complete can be put into this form. There exists problems that are not NP-complete, but are put into this form as well.) A solution vector that lies in x is called a feasible solution, and a feasible solution for which $C_0(x)$ is minimal is called an optimal solution.

The branch and bound algorithm is an organized and intelligently structured search of the space of all feasible solutions. It has been extensively studied in areas such as artificial intelligence and operations research [9, 19, 21, 22]. It has been applied to solve problems in scheduling [17, 20], knapsack [13, 14], travelling salesman [5], facility allocation [2], integer programming [4, 6], and many others. Dominance relations similar to those used in dynamic programming have been used to prune search tree nodes. Theoretical properties of branch and bound algorithms have been developed in several studies [10, 12, 16, 22].

In branch and bound algorithms [19, 21], the space of all feasible solutions is repeatedly partitioned into smaller and smaller subsets, and both the lower and upper bounds are calculated for solutions within each subset. After each partitioning, subsets with lower bounds (in the case of minimization) that exceed either the value of a known feasible solution or the least upper bound of all subsets are excluded from further consideration. The partitioning process continues until a feasible solution is found such that the value is no greater than the lower bound of any subsets.

The state of the partitioning process at any time can be represented as a partial tree (Figure 1). Each node in the tree represents a partition and is called a subproblem. The partitioning process selects a partition and breaks up this partition into smaller partitions. This extends the node in the partial tree representing this partition by one level and uses the sons to denote the smaller partitions. In Figure 1, node j is expanded in the partitioning process into k other partitions, which are represented as sons of node j in the partial tree.

There are two essential features of a branch and bound algorithm: the branching rule and the bounding rule. With respect to the partial tree in Figure 1, each

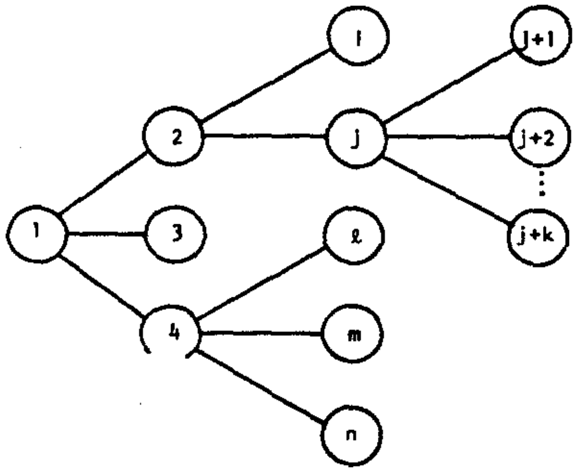


Figure 1. A branch and bound tree.

Research supported by National Science Foundation Grant ECS81-05968 and a grant from the Center for Automation, Design and Manufacturing of Purdue University

node in the tree has two numbers associated with it - the upper bound and the lower bound of the sub-problem. The leaf nodes in the partial tree are candidates for partitioning. A leaf node of the partial tree whose lower bound is less than both the value of a known feasible solution and the greatest upper bound of all leaf nodes is active; otherwise, it is terminated and need not be considered in any further computation.

The branching algorithm examines the set of active leaf nodes and, based on some predefined criterion, selects one for expansion. If the set of active nodes is maintained in a first-in first-out (FIFO) list, the algorithm is called a breadth-first search. If the set is maintained in a last-in, first-out list, the algorithm is called a depth-first search. Lastly, if the node selected for expansion is one with the minimum lower bound, the search algorithm is called a best-first search.

In a breadth-first search, the nodes of the tree will always be examined in levels. That is, a node at a lower level will always be examined before a node at a higher level. This search will always find a goal node nearest to the root; however, the sequence of nodes examined is predetermined, so the search is "blind." The depth-first search has a similar behavior except that a subtree is generated completely before the other subtrees are examined. In both algorithms, the next node to be examined is known, so the state of the parent node leading to the next node from the root node is easily found and is unique. Furthermore, the memory space required for storing the state is very small. These two algorithms are, therefore, space-saving.

In contrast, the best-first search is space-consuming because all active subproblems must be stored as intermediate data in the computer. The total number of nodes expanded, however, is minimum in the sense that any branching operation performed under this policy must also be performed under other policies, provided that all the bounds are unique [19]. Since time is a more critical factor in evaluating large optimization problems, the behavior of the best-first search merits further study.

One study [11] shows that depth-first, breadth-first and best-first searches are special cases of heuristic search. In heuristic search, an evaluation function f(n) for a sub-problem n is computed as the sum of cost of an optimal path from a given start node to n and cost of an optimal path from n to a goal. An ordered search algorithm picks up a sub-problem with the minimum value of f for expansion each time. Any general heuristic functions can be included in the computation and the choice of a heuristic function depends on the application.

Once the subproblem has been selected for partitioning, some undetermined parameters in the subproblem must be selected so that alternatives for these parameters can be defined and multiple sub-problems created. For example, in the traveling salesman problem, the undetermined alternatives are the set of untraversed edges. In expanding a sub-problem, an untraversed edge (i,j) is selected, and two alternatives can be created: (1) the edge is traversed and the salesman goes directly from city i to city j and (2) vice versa. The parameter chosen to be expanded is usually done ad hoc.

After new sub-problems are created, the bounding algorithm is applied to evaluate the upper and lower bounds of a sub-problem. Generally, only the lower bound is evaluated, because the merits of using the upper bound are small. The bounding algorithm that is designed is highly dependent on the problem. For example, in an integer programming problem, a linear program with relaxed integer constraints can be used as a lower bound [18]; in the traveling salesman problem, an assignment algorithm [1] or a spanning tree algorithm can be used as the bounding algorithm.

As an example to illustrate the use of branch and bound algorithms, the evaluation of an integer programming problem is shown here. The integer programming problem may be expressed as

$$\begin{aligned} &\text{Minimize} && CX \\ &\text{subject to} && AX \geq B \\ &&& X^T = (x_1, x_2, \ldots, x_n) \\ &&& x_i, \text{ non-negative integer, } i=1,2,\ldots n. \end{aligned}$$

These problems differ from ordinary linear programming problems in that the variables are restricted to non-negative integer values.

One approach to the problem is the following. Apply the dual simplex method to a sub-problem. If the optimal solution is integral, a feasible solution has been generated, otherwise, create two new sub-problems as follows. Choose a variable that has a non-integer value (say $x_i = 4.4$) and restrict that variable to the next lower integral value for one problem ($x_i \leq 4$) and to the next higher integral value ($x_i \geq 5$) for the other. The variable chosen is the one with the greatest up or down penalty. The up penalty for a variable $x_i$ having a value of $a_i$ is the estimate of the amount by which the solution to the current subproblem would increase if the integral constraint $x_i \geq \lceil a_i \rceil$ was introduced. The down penalty is similar, except that it is associated with the constraint $x_i \leq \lfloor a_i \rfloor$. The lower bound of a new sub-problem is the sum of the optimal simplex solution and the associated penalty. This process is repeated on the new sub-problems.

Figure 2(b) shows the branch and bound tree for the problem in Figure 2(a). The dual simplex method gives an optimal solution of 14.2 for the original problem. Since the variables are not integral, a feasible solution has not been generated. Up and down penalties are calculated for the variables and $x_1$ has the greatest penalty ( U = 1.8 ). Two new sub-problems are then created, one with $x_1=0$ and the other with $x_1 \geq 1$. The lower bounds are calculated as in Figure 2(b). The dual simplex method is then applied to the sub-problem with the smaller lower bound and a feasible solution is generated with all variables having integral values. This constitutes an optimal solution since the lower bound of the remaining sub-problem is greater.

In this paper we present an approximate stochastic model of the branch and bound algorithm. This model is important because it forms the basis of design of a virtual memory operating system supporting branch and bound algorithms [25]. Furthermore, the model is useful in studying the behavior of approximate branch and bound algorithms [24].

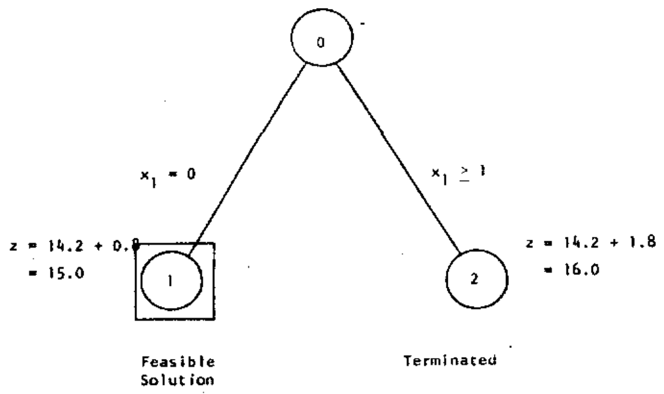
$$\begin{aligned} \min x_0 &= 7x_1 + 3x_2 + 4x_3 \\ x_1 + 2x_2 + 3x_3 &\geq 8 \\ 3x_1 + x_2 + x_3 &\geq 5 \\ x_1, x_2, x_3 &\geq 0, \ \textit{Integer} \end{aligned}$$

Figure 2a. An example of an integer programming problem.

Optimal dual simplex solution

$$\boxed{\begin{array}{l} x_0 = 14.2 \\ x_1 = 0.4 \\ x_2 = 3.8 \\ x_3 = 0 \end{array}}$$

| variable | down penalty | up penalty |
|---|---|---|
| $x_1$ | 0.8 | 1.8 |
| $x_2$ | 0.3 | 0.13 |



Optimal dual simplex solution

$$\boxed{\begin{array}{l} x_0 = 15.0 \\ x_1 = 0 \\ x_2 = 5 \\ x_3 = 0 \end{array}}$$

z = lower bound

Figure 2b. Branch and bound tree for Figure 2a.

## 2. The Model of the Branch and Bound Process with Best First Search

The branch and bound process can be modelled as two walls moving towards each other. The front wall indicates the value of the lower bound for the sub-problems currently expanded. The back wall represents the minimum of all the feasible solutions.

Initially, the front wall is undefined and the back wall is at infinity. The lower bound for the problem is evaluated and this is taken to be the position of the front wall. The problem is then branched into two or more sub-problems and a lower bound is calculated for each sub-problem. Since the lower bounds of descendent sub-problems are always greater than the lower bound of ancestor sub-problems, the front wall always moves to the right (see Figure 3). Once the current sub-problem has been expanded, the front wall moves to the position of the minimum of the set of active sub-problems. This sub-problem is then expanded and the process repeats.

When a sub-problem generated becomes a feasible solution, the value of the solution is compared with the position of the back wall. If the position of the back wall is greater than the value of the new feasible solution, the back wall is set to this value; otherwise the feasible solution is ignored. Successive expansion of the sub-problems cause the front and back walls to approach each other and the process is terminated when the two walls coincide.

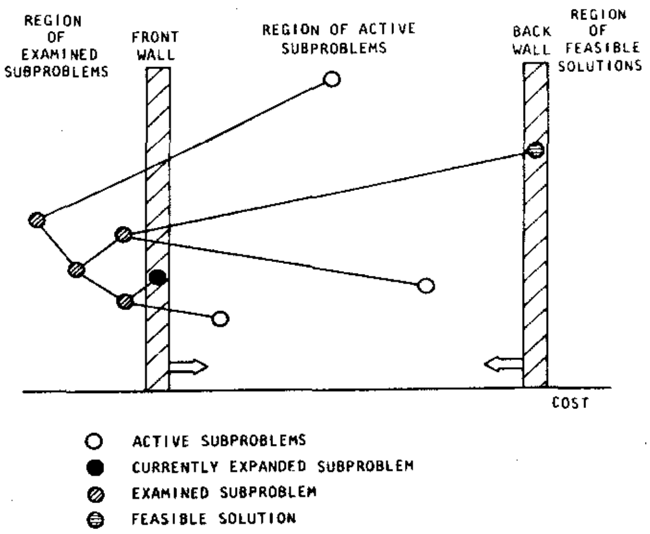

REGION OF EXAMINED SUBPROBLEMS
FRONT WALL
REGION OF ACTIVE SUBPROBLEMS
BACK WALL
REGION OF FEASIBLE SOLUTIONS
COST
○ ACTIVE SUBPROBLEMS
● CURRENTLY EXPANDED SUBPROBLEM
⊘ EXAMINED SUBPROBLEM
⊖ FEASIBLE SOLUTION

Figure 3. The model of branch and bound process with best-first search.

In the following sections, the positions of the front and back walls are calculated. Some simplifying assumptions are made in order for the calculations to be tractable.

### 2.1 The Position of the Front Wall

The solution of the following problem is desired: given the position of the front wall, what is the expected number of sub-problems examined; or inversely, given the number of sub-problems examined, what is the expected position of the front wall. The set of examined sub-problems consists of sub-problems that have been processed and no longer belong to the set of active sub-problems. The following assumptions are made in the derivation:

(A1) The differences between the lower bounds of the expanded sub-problems and the parent sub-problem are independent, identically distributed random variables satisfying the gamma density function [8]

$$f_G(y;\alpha,\lambda) = \begin{cases} \dfrac{\lambda^\alpha}{\Gamma(\alpha)}\, y^{\alpha-1} e^{-\lambda y} & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{1}$$

The density function is monotonic if $\alpha \leq 1$, and unbounded near the origin when $\alpha < 1$. For $\alpha > 1$, the graph is bell-shaped and as $\alpha \rightarrow \infty$, the density function becomes normal [3]. A Gamma density function is chosen because it represents a very general class of density functions. As shown in the next section, this assumption is valid for integer programming.

(A2) Each parent sub-problem is expanded into two smaller sub-problems. This assumption is valid for a class of NP-complete problems.

Let I be the lower bound of the first sub-problem. Let N(x) be the number of sub-problems examined when the front wall is at position x and E(N(x)) be the expected value of N(x). When the parent sub-problem is expanded, y' and y'' are the differences between the lower bounds of the expanded sub-problems and the parent sub-problem. E(N(x)) can be written in the form of a renewal equation [23].

$$E(N(x)) = 1 + \int_0^\infty E_1(N(x-y')dF_G(y') + \int_0^\infty E_1(N(x-y'')dF_G(y'')$$

or

$$E(N(x)) = 1 + 2\int_0^\infty E_1(N(x-y)dF_G(y) \tag{2}$$

where

$$E_1(N(x-y)) = \begin{cases} E(N(x-y)) & \text{if } y < x \\ 0 & \text{if } y \geq x \end{cases} \tag{3}$$

The evaluation of the above renewal equation would result in an incomplete gamma function that cannot be solved analytically. Since $f_G(y) \to 0$, as $y \to \infty$, the assumption that x is reasonably large implies that for any $y > x$, $f_G(y) \simeq 0$. This leads to an approximate renewal equation which can be written as:

$$E'(N(x)) = 1 + 2\int_0^\infty E'(N(x-y))dF_G(y) \tag{4}$$

To solve Eq. (4), a solution is guessed and is substituted into Eq. (4) in order to verify it. Assume that $E'(N(x)) = k\,e^{mx} - 1$. Substituting into Eq. (4), we obtain an identity

$$k\,e^{mx} - 1 = 1 + 2\int_0^\infty (k\,e^{m(x-y)} - 1)dF_G(y)$$

or

$$1 = 2\int_0^\infty e^{-my}dF_G(y) \tag{5}$$

Using the density function of Eq. (1) and substituting it into Eq. (5), m can be solved,

$$m = \lambda(2^{1/\alpha} - 1) \tag{6}$$

To solve for the constant k, we use the boundary condition $E'(I) = 1$. Substituting for $x = I$ in the assumed solution, we obtain $k = 2e^{-mI}$. Therefore,

$$E'(N(x)) = 2\,e^{\lambda(2^{1/\alpha}-1)(x-I)} - 1 \tag{7}$$

As similar to problems in general renewal theory, the derivation of the distribution function of N(x) is difficult. The expected value of N(x) will, therefore, be used in the calculation of the position of the back wall.

At this time, it is important to know the total number of sub-problems generated. All the sub-problems to the left of the front wall must have been examined (non-terminal nodes) and all the sub-problems to the right of the front wall are active and not examined (terminal nodes). Assuming a well balanced binary tree, the approximate expected total number of nodes in the branch and bound tree is $E(N_T(x))$ and using Eq. (7),

$$E(N_T(x)) \simeq 2\,E'(N(x)) + 1 \tag{8}$$

## 2.2 The Position of the Back Wall

To determine the position of the back wall, the mechanism involved in generating a feasible solution must be understood. Let n be the number of input parameters. n can be the number of variables in an integer programming problem; n can be the number of cities that a travelling salesman wishes to visit; n can also be the number of nodes in a graph of the vertex covering problem. Before a feasible solution can be obtained, a chain of sub-problem expansions must be generated. The number of sub-problems in a chain can be less than n (vertex covering problem), equal to n (integer programming problem) or greater than n (travelling salesman problem). To evaluate the position of the back wall, the following additional assumptions are made:

(A3) Every chain that results in a feasible solution is made up of n sub-problem evaluations. Each chain starts at the origin and has a length equal to the sum of n independent gamma distributed random variables. Variable length chains will be considered in the future.

(A4) The chains leading to feasible solutions are independent. This assumption is not true in general but is necessary for mathematical tractability.

The number of chains due to $E(N_T(x))$ nodes in the branch and bound tree is C(n,x), and the maximum is $C_{max}(n,x)$ which is given by the following equation.

$$E(N_T(x)) = \sum_{j=0}^{n-1} \left\lceil \frac{C_{max}(n,x)}{2^j} \right\rceil \tag{9}$$

$C_{max}(n,x)$ can be solved by first calculating its approximate value without the ceiling in Eq. (9). and searching for the solution in the vicinity of the approximate value.

The actual number of chains formed is, of course, less than $C_{max}(n,x)$. The position of the back wall estimated using $C_{max}(n,x)$ will, therefore, be a lower bound of the actual position.

By assumption (A3), the length of each chain is also gamma distributed with a density function $f_c(y)$ since the family of gamma densities is closed under convolution.

$$f_c(y) = f_G(y;\, n * \alpha,\, \lambda) \tag{10}$$

Since all the chains are assumed independent (assumption (A4)), the position of the back wall is given by the minimum value of all the chains. The distribution function of the minimum of C(n,x) independent, identically distributed random variables is $F_B(y)$ where,

$$F_B(y) = 1 - [1 - F_c(y)]^{C(n,x)} \tag{11}$$

The expected position of the back wall is E(b)

$$E(b) = \int_0^\infty y\, dF_B(y) \tag{12}$$

## 2.3 The Distribution of Sub-Problems Behind the Front Wall

In this section, the distribution function of the difference between the lower bounds of active sub-problems and the front wall is calculated. The generation of active sub-problems is depicted in Figure 4. The distribution functions of $z_1$ and $z_2$ are sought. Since $y_1$ and $y_2$ are gamma distributed, an assumption which simplify the calculation is the following.

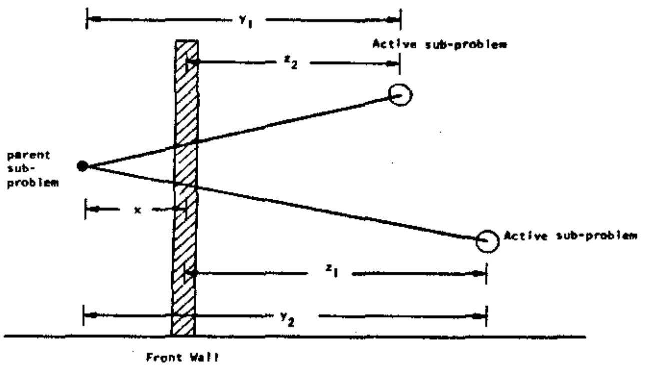


Figure 4. The generation of a pair of active sub-problems.

(A5) The parent sub-problem and the corresponding active sub-problems it generates can be at any position as long as they lie on opposite sides of the front wall.

From the above assumption, it implies that x and $z_2$ have the same distribution function. Since $y_1$ is gamma distributed, x and $z_2$ are also gamma distributed.

$$f_x(y) = 2 * f_{z_2}(y) = f_G(y; \alpha/2, \lambda) \tag{13}$$

Similarly, $z_1$ is gamma distributed with an additional constraint that $z_1 > z_2$. Therefore,

$$f_{z_1}(y) = f_G(y; \alpha/2, \lambda)[1 - F_G(y; \alpha/2, \lambda)] \tag{14}$$

In this section, we have derived the analytical behavior of the branch and bound algorithms and have shown that the active sub-problems behind the front wall are gamma distributed. In the next section, the analytical model is compared against some simulation results.

## 3. Comparison of the Analytical Model with Simulations

A program to solve an integer program was written in the c language and run on a VAX 11/780 computer at Purdue. It takes about 10 minutes of CPU time and 15 Mbytes of memory to solve a 20 variable 20 constraint integer program. The cumulative statistics on the increase in the lower bounds of son subproblems with respect to the parent subproblem was collected over the duration of the solution process. An exponential distribution was fitted on the collected statistics and the results are shown in Table 1. The .20 critical value for the Kolmogorov-Smirnov test is 0.076. Thus the hypothesis that the density function of the increase in lower bounds of son sub-problems is exponential may be accepted. (An exponential distribution implies $\alpha = (1)$ in Eq. 1.)

Table 1 Testing of the hypothesis that the increase in lower bounds of son sub-problems is exponentially distributed for 20 variable, 20 constraint integer programming problems. (A histogram of 200 buckets is used for the experimental distribution.)

| Problem # | Sample mean | Kolmogorov-Smirnov Variable, $D_n$ |
|---|---|---|
| 1 | 0.184 | 0.057 |
| 2 | 0.322 | 0.063 |
| 3 | 0.170 | 0.062 |
| 4 | 0.230 | 0.076 |
| 5 | 0.312 | 0.049 |

Using the analytical expressions derived in Section 2, we have plotted in Figures 5 to 7 the performance of the branch and bound algorithm using best-first search. In Figures 5 and 6, the position of the front and back walls are plotted for two runs of the integer program using the measured mean of the exponential distribution. It is seen that the expected number of sub-problems examined increases exponentially with the position of the front wall. On the other hand, the position of the back wall approaches the front wall as the number of sub-problems examined is increased. However, the approach is rather slow and the slope of the graph for the back wall is steep. This implies that as the problem size becomes larger, the number of sub-problems that have to be examined before the process terminates increases exponentially.

In spite of the various assumptions that we have

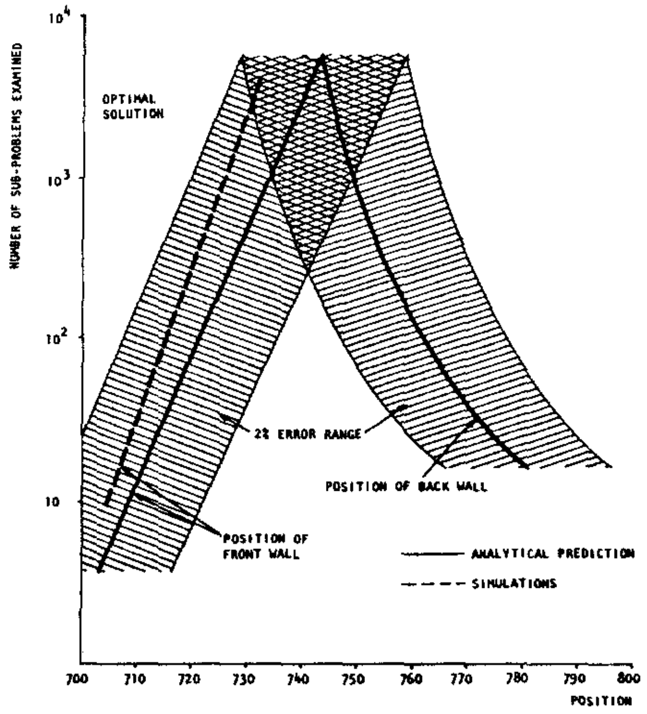


Figure 5. The positions of the front and back walls for a 20 variable 20 constraint integer program - RUN 1.

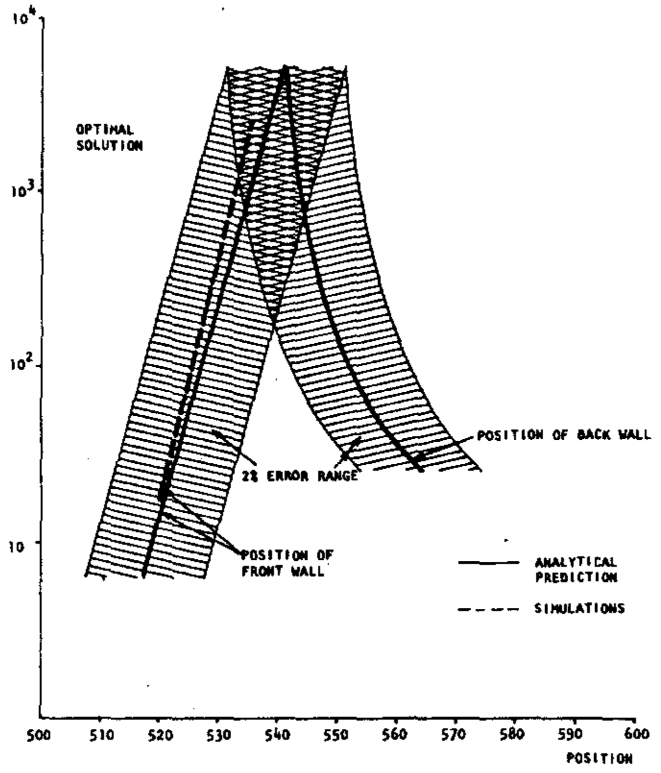


Figure 6. The positions of the front and back walls for a 20 variable 20 constraint integer program - RUN 2.

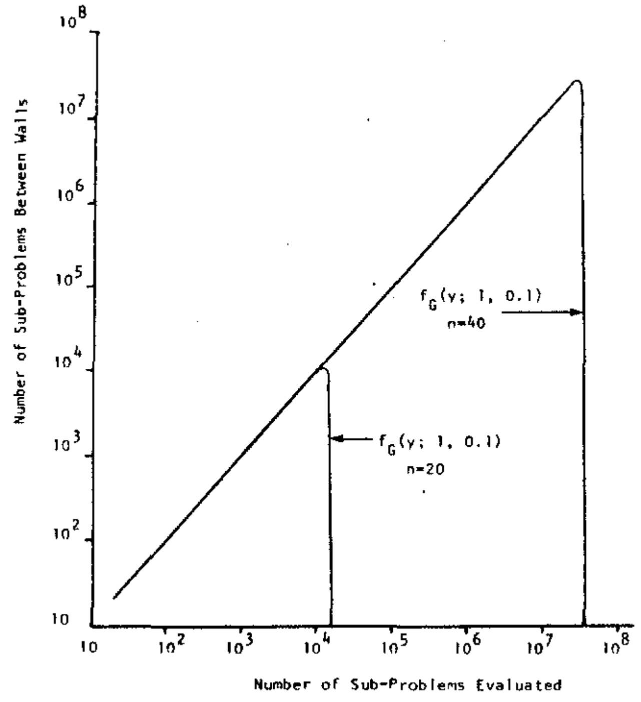


Figure 7. The number of active sub-problems between the two walls as a function of the number of sub-problems evaluated.

made, the estimated positions of the front wall match to within two percent of the simulated position. The results concerning the simulation of the back wall position is not plotted because an initial feasible solution is not generated in our runs and the first feasible solution obtained usually becomes the optimal solution. Nonetheless, assuming 2% error in the estimated positions of the back wall which are plotted in Figures 5 and 6, the number of iterations at termination is predicted correctly. Due to the steepness of the curves and the exponential scale used, the predicted number of iterations may lie in a range of several orders of magnitude.

In Figure 7, the analytical number of active sub-problems as a function of the number of sub-problems examined is plotted. It indicates that the number of active sub-problems first grows to a maximum and decreases to zero at the termination of the process. Furthermore, as n is doubled, the number of active sub-problems grows by a factor of $10^4$. These indicate the need of an efficient memory management scheme for storing the active sub-problems.

## 4. Concluding Remarks and Implications

In this paper, we have studied the probabilistic modelling of the branch and bound algorithms. The model consists of two walls approaching each other. The front wall represents the value of the lower bound for the sub-problem currently being expanded. The back wall represents the minimum of all feasible solutions. These two walls approach each other and eventually coincides at the termination of the process.

In the derivation of the positions of the front and back walls, it was assumed that the differences between the lower bounds of the expanded sub-problems and the parent sub-problems are independent, identically distributed random variables satisfying the gamma density function. Subsequently, it was shown that the distribution is exponential for integer programming (gamma distribution with $\alpha = 1$ ). Some simplifying assumptions were also introduced to make the model mathematically tractable. Lastly, the distribution and number of active sub-problems were derived.

The model clearly shows that the sub-problems constitute a dynamically varying list ordered by lower bounds. The access characteristics of the branch and bound algorithm calls for the access of items at the head of this list and insertion into the list. The $B^+$-tree [15] which requires dynamic rebalancing, appears to the best organization for pages in the virtual memory operating system, especially if the index portion of the tree can be kept in main memory.

Nonetheless, the overhead incurred by a page fault is considerable. This may be minimized with a properly designed replacement algorithm. We have developed a simulation model and used the gamma distribution function derived in Eq. (14) to drive the replacement algorithm. Due to the special shape of the gamma distribution with $\alpha = 0.5$, when a constant number of sub-problems at the head of the list are removed, it results in a smaller number of page faults. However, usage of the best-first search implies that sub-problem with the minimum lower bound is evaluated each time which may be replaced by the replacement algorithm. To avoid the additional page faults due to this fetch, the first page of the $B^+$ tree is always kept in primary memory. Analysis and simulations show that the replacement algorithm is very effective [25].

The results derived on the positions of the front wall are very important in showing the effectiveness of approximate branch and bound algorithms. Lawler proposed the use of branch and bound algorithms as a general purpose heuristic to compute solutions that differ from the optimum by no more than a prescribed amount [19]. Suppose it was decided at the outset that a deviation of 10% from the optimum is tolerable. If a feasible solution of 150 is obtained, then all sub-problems with lower bounds of 136.4 or more $(= \frac{150}{1.1})$ will be terminated.

In general, suppose a feasible solution of value F is obtained initially and the optimal solution has value P $(P \geq F)$. Let $\eta$ be the prescribed degree of accuracy. This implies that all sub-problems with values greater than $\min\left[\frac{F}{1+\eta}, P\right]$ can be eliminated. From Eq. (7) the number of subproblems evaluated is,

$$E'_\eta(N(x)) = 2\, e^{\lambda(2^{\frac{1}{\alpha}}-1)(\min(\frac{F}{1+\eta},P)-I)} - 1 \tag{15}$$

The number of subproblems evaluated when the optimal solution is found is

$$E'_0(N(x)) = 2\, e^{\lambda(2^{\frac{1}{\alpha}}-1)(P-I)} - 1 \tag{16}$$

The number of iterations saved using approximation is $E_0(N(x)) - E_\eta(N(x))$. If a good initial feasible solution is generated, then the maximum number of iterations saved is when $F = P$ and is proportional to $2^{\lambda P(2^{1/\alpha}-1)} - 2^{\frac{\lambda P(2^{1/\alpha}-1)}{1+\eta}}$ which is exponential in $\eta$.

The simple derivation here illustrates that the approximate branch and bound algorithm is an effective technique in reducing the total number of iterations. A future paper will pursue on the anomalies of approximate branch and bound algorithms [24].

## REFERENCES

[1] W. L. Eastman, "A Solution to the Traveling Salesman Problem," presented at the *American Summer Metting of the Econometric Society,* Cambridge, Mass., Aug. 1958.

[2] M. A. Efroymson and T. C. Ray, "A Branch and Bound Algorithm for Plant Location," *Operations Research,* Vol. 14, pp. 361-368, 1966.

[3] W. Feller, *An Introduction to Probability Theory and its Applications,* Vol. II, 2nd edition, John Wiley & Sons, Inc., 1971.

[4] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming,* John Wiley and Sons, Inc., New York, 1972.

[5] R. Garfinkel, "On Partitioning the Feasible Set in a Branch and Bound Algorithm for the Asymmetric Travelling Salesman Problem," *Operations Research,* Vol. 21, No. 1, pp. 340-342, 1973.

[6] A. M. Geoffrion and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," *Management Science,* Vol. 18, No. 9, pp. 465-491, May 1972.

[7] L. Guibas and R. Sedgewick, "A Dichromatic Framework for Balanced Trees," *Proc. 19'th Symp. Foundations of Computer Science,* pp. 8-21, 1978.

[8] P. G. Hoel, S. C. Port and C. J. Stone, *Introduction to Probability Theory,* Houghton Mifflin Co., 1971.

[9] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press, Maryland, 1978.

[10] T. Ibaraki, "Computational Efficiency of Approximate Branch and Bound Algorithms," *Math. of Oper. Research,* Vol. 1, No. 3, pp. 287-298, 1976.

[11] T. Ibaraki, "Theoretical Comparisons of Search Strategies in Branch and Bound Algorithms," *Int. Jr. of Comp. and Info. Sci.,* Vol. 5, No. 4, pp. 315-344, 1976.

[12] T. Ibaraki, "Depth-m Search in Branch-and-Bound Algorithms," *Int. Jr. of Comp. and Inf. Sci.,* Vol. 7, No. 4, pp. 315-343, 1978.

[13] G. Ingargiola and J. Korsh, "A Reduction Algorithm for Zero-one Single Knapsack Problems," *Management Science,* Vol. 20, No. 4, pp. 460-663, 1973.

[14] G. Ingargiola and J. Korsh, "A General Algorithm for One Dimensional Knapsack Problems," *Operations Research,* Vol. 25, No. 5, pp. 752-759, 1977.

[15] D. E. Knuth, *The Art of Computer Programming, Sorting, and Searching,* Vol. 3, Addison-Wesley, 1973.

[16] W. Kohler and K. Steiglitz, "Characterization and Theoretical Comparison of Branch and Bound Algorithms for Permutation Problems," *JACM,* Vol. 21, No. 1, pp. 140-156, 1974.

[17] B. Lageweg, J. Lenstra and A. Rinnooy Kan, "Job-shop Scheduling by Implicit Enumeration," *Management Science,* Vol. 24, No. 4, pp. 441-400, 1977.

[18] A. H. Land and A. Doig, "An Automatic Method for Solving Discrete Programming Problems," *Econometrica,* Vol. 28, pp. 497-520, 1960.

[19] Lawler, E. L. and Wood, D. W., "Branch and Bound Methods: A Survey," *Operations Research,* Vol. 14, pp. 699-719, 1966.

[20] J. Lenstra, "Sequencing by Enumerative Methods," *Math. Centre. Tract 69,* Mathematisch Centrum, Amsterdam, 1976.

[21] L. Mitten, "Branch and Bound Methods: General Formulation and Properties," *Operations Research,* Vol. 18, pp. 24-34, 1970.

[22] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence,* McGraw Hill, New York, 1971.

[23] S. M. Ross, *Applied Probability Models with Optimization Applications,* Holden-Day, San Francisco, 1970.

[24] B. Wah and G.J. Li, "The Anomalies of Parallel Approximate Branch and Bound Algorithms," under preparation.

[25] C.F.Yu *Virtual Memory Support for Branch and Bound Algorithms,* Masters Thesis, Purdue University, 1982.