ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF
NATIONAL PLANNING

التخطيط القومي

Memo. No. 1213

BRANCH-AND-BOUND REVISITED: A SURVEY OF BASIC CONCEPTS AND THEIR APPLICATIONS IN SCHEDULING

BY

SALAH E ELMAHGRABY
ALWALID N. ELSHAFEI

DEC. 1977

CAIRO
SALAH SALEM St.-NASR CITY

MODERN TRENDS IN LOGISTICS RESEARCH

Proceedings of a Conference Held at
The George Washington University

Edward S. Bres III	Arnoldo C. Hax
A. Charnes	Fred Kornet, Jr.
W. W. Cooper	Arthur I. Mendolia
S. E. Elmaghraby	Frank Proschan
A. N. Elshafei	David A. Schrady
Richard L. Francis	Jeremy F. Shapiro
Walter D. Gaddis	Ronald W. Shephard
Donald P. Gaver	William W. Snavely
Murray A. Geisler	Gerald L. Thompson
Donald Gross	S. Zacks

Edited by

W. H. Marlow

The MIT Press
Cambridge, Massachusetts, and London, England

CONTRIBUTORS

Edward S. Bres III is a Research Associate, Center for Cybernetic Studies, University of Texas at Austin.

A. Charnes is the University of Texas System Professor, Jesse H. Jones Professor of Biomathematics and Management Science, and Director, Center for Cybernetic Studies, University of Texas at Austin.

W. W. Cooper is University Professor of Public Policy and Management Science, School of Urban and Public Affairs, Carnegie-Mellon University.

S. E. Elmaghraby is University Professor of Operations Research and Industrial Engineering, and Director, Operations Research Program, North Carolina State University.

A. N. Elshafei is Expert, The Institute of National Planning, Nasr City, Cairo, Egypt.

Richard L. Francis is Professor, Department of Industrial and Systems Engineering, University of Florida.

Walter D. Gaddis served as Deputy Chief of Naval Operations (Logistics) from April 11, 1973 to August 1, 1975.

Donald P. Gaver is Professor and Associate Chairman for Research, Department of Operations Research and Administrative Sciences, United States Naval Postgraduate School.

Murray A. Geisler is former Director of Logistics Studies, Rand Corporation, Santa Monica, California.

Donald Gross is Professor, Department of Operations Research, George Washington University.

Arnoldo C. Hax is Associate Professor of Management Science, Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Fred Kornet, Jr. served as Deputy Chief of Staff for Logistics, Department of the Army, from January 2, 1973 to August 31, 1975.

W. H. Marlow is Professor and Chairman, Department of Operations Research, Principal Investigator of Program in Logistics, and Director, Institute for Management Science and Engineering, George Washington University.

Arthur I. Mendolia served as Assistant Secretary of Defense (Installations and Logistics) from June 21, 1973 to March 31, 1975.

Frank Proschan is Professor, Department of Statistics, Florida State University.

David A. Schrady is Chairman, Department of Operations Research and Administrative Sciences, United States Naval Postgraduate School.

Jeremy F. Shapiro is Professor of Operations Research and Management, Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Ronald W. Shephard is Professor of Engineering Science and Chairman, Department of Industrial Engineering and Operations Research, University of California, Berkeley.

William W. Snavely served as Deputy Chief of Staff for Systems and Logistics, Department of the Air Force, from February 1, 1973 through August 31, 1975.

Gerald L. Thompson is Professor of Applied Mathematics and Industrial Administration, Carnegie-Mellon University.

S. Zacks is Professor and Chairman, Department of Mathematics and Statistics, Case Western Reserve University, and Consultant, Program in Logistics, George Washington University.

Chapter 8

BRANCH-AND-BOUND REVISITED: A SURVEY OF BASIC CONCEPTS AND THEIR APPLICATIONS IN SCHEDULING*

S. E. Elmaghraby and A. N. Elshafei**
North Carolina State University

8.1 Preliminaries

The term "branch-and-bound" (B&B) has increasingly become a household term among students and researchers in the field of scheduling and sequencing. In this chapter we shall take a fresh look at this approach and assess its content, utility, and potential. In delineating the subject matter of our discussion, perhaps it is equally valid to emphasize that which is not among our aims. This chapter is not a comprehensive survey of B&B concepts and applications. Several survey articles that have appeared in recent years serve that function adequately, if not superbly; see, for example, References [8.1, .4, .21, .36, .40]. Nor does this paper aspire to be a comparative evaluation of the very many B&B approaches that have been proposed in the open literature to solve one scheduling problem or another. For examples of such studies, the reader is referred to the papers of Ashour [8.2], Ashour and Quraishi [8.3], Davis [8.8], and Kan [8.34], among others.

What we do wish to present is an inventory of the basic concepts underlying the "theory" of B&B; we wish in fact to establish that such theory exists and to illustrate these basic concepts by examples from the field of scheduling and sequencing. In this we are

---

*The preparation of this chapter was partially supported by the Office of Naval Research under Contract N00014-70-A-0120-0002, by the National Science Foundation under Grant P1K1470-000, and by the Army Research Office-Durham under Contract DA-ARO-D-31-124-72-G106, with North Carolina State University.

**Now at the Institute of National Planning, Cairo, Egypt

motivated by two objectives. The first is to summarize, in what we hope is a convenient place, the multitude of concepts that have emerged over the past few years. We hope that such a summary will provide a handy reference and basic understanding to student and researcher alike. The second is to help the profession assess the current and future potential of this approach. In this respect, one may compare B&B as a problem solving approach, to simulation which is another, and by now a very popular problem-solving approach. One may then ask fundamental questions similar, but not necessarily identical, to those asked in the study of simulation. For instance, in Monte Carlo simulation one often raises the question of variance-minimizing techniques. In B&B one may ask questions relative to the rate of convergence to the optimum.

As much as possible we shall draw our examples from the field of scheduling and sequencing. However, since problems of scheduling (and sequencing) are almost universally modeled as integer or mixed programming problems (linear or nonlinear), we shall feel free to illustrate some concepts with reference only to the integer (or mixed) program, without the need to motivate the model by the scenario of the scheduling problem. Ordinarily, we shall be dealing with integer linear problems (ILP) and, in particular, with 0,1 ILPs. As is well known, an ILP can be translated into a 0,1 ILP by the simple binary expansion of the variables. In a couple of instances, we could not find examples from scheduling and, to the best of our knowledge, none exist that use a particular concept. Then we took the liberty to illustrate by examples from other fields of application, such as location-allocation. We do not feel particularly apologetic about taking such liberty since these problems are themselves modeled as integer (linear or nonlinear) programs. Such models provide the link to problems in scheduling.

In the sequel we shall be talking about "partial solutions" and "completions." The term "partial solution" is actually a misnomer, since it refers to something that provides no solution whatsoever to the original problem. For instance, a schedule of a subset of the jobs, or a series of cities visited by the salesman in the traveling salesman problem (TSP), are referred to as partial solutions, yet they provide no "solution" to the problems posed, which are: a complete schedule of all jobs in the first case, and a complete tour over all cities in the second case. The reader will hopefully bear with this misuse of language. By the completion of a partial solution we mean the specification of the values of the remaining variables so that their union with the partial solution yields a point in the original solution space. A partial solution is said to be fathomed if one of the two following conditions is satisfied.

(1) We determine that its best feasible completion is better (yields a better objective value) than the best feasible solution known to date (assuming one is in hand).

(2) We determine that the partial solution has no feasible completion better than the incumbent (this includes infeasibility, which is translated into infinite penalty).

The concept of fathoming is illustrated in Example 8.3.

8.2 Fundamentals

The approach of B&B is basically a heuristic tree search in which the space of feasible solutions, which may contain a very large (or denumerable) number of points, is systematically searched for the optimum. According to Mitten and Warburton [8.42], "the search proceeds iteratively by alternately applying two operations: subset formation and subset elimination. In the former, new subsets of alternatives are formed, while in the latter some subsets of alternatives may be eliminated from further consideration. The procedure terminates when a collection recognized to contain only optimal solutions is reached." The search has two guiding principles: first, that every point in the space is enumerated either explicitly or implicitly, and, second, that the minimum number of points be explicitly enumerated. (We view B&B as an approach for implicit enumeration, though we concede that, mainly due to historical coincidences, the label "implicit enumeration" has been applied to approaches that need not employ the "bounding" feature of B&B.)

The implicit enumeration of feasible points is accomplished through dominance (which may or may not employ bounding) and feasibility considerations. Each of these concepts will be discussed in greater detail below, but first we give a laconic description of them to afford the uninitiated reader a general grasp of the subject. The basic idea in B&B is to divide the feasible space, denoted by S, into subsets $S_1, S_2, \ldots, S_k$ which may or may not be mutually disjoint. Assuming that the optimum falls in subset S_k, a bound on its value is determined: an upper bound (u.b.) in the case of maximization, and a lower bound (ℓ.b.) in the case of minimization or, better still, both an upper and a lower bound in either case. Based on such bounds two actions may take place: (i) a particular subspace S_k is selected for more intensive search by further partitioning into its subsets (this is the branching, or "formation" function); (ii) some feasible points (subspaces) are declared "noncandidates" for the optimum, and thus are eliminated from further considerations. This latter idea is one of "dominance" since it is based on the determination that any element of a particular subset S_i is better (or worse), in the sense of the criterion function, than any element in another subset S_j. Then indeed we may declare the points in S_j (or in S_i) as noncandidates for the optimum and eliminate them from further analysis.

While dominance may be established on the basis of the bounds evaluated on subsets S_k, it is also true that dominance can be established independent of any bounding considerations. In some circles (especially in the scheduling literature) these are referred to as "elimination" procedures. The final result is the same, namely, it establishes that certain subspaces cannot contain the optimum because they are dominated by other subsets. A similar idea lies behind the feasibility considerations. They arise because in the majority of cases one is forced to hypothesize a rather "rich" original space S. At some stage of analysis, if it can be established that certain subsets of S are in fact infeasible (in the sense of violating some constraint of the problem), then indeed such subsets can also be eliminated from further study.

Heuristics enter the tree search in all three basic phases of the approach: in the definition of the partitioning procedure, in the calculation of the bounds, and in the philosophy of searching the tree. But we wish to draw the reader's attention to the following important and rather crucial distinction: the formal structure of B&B admits the use of heuristics (as does the simplex algorithm of linear programming). However, these are "reliable heuristics" in the sense that if they run to completion, the optimum will be achieved. Furthermore, if the procedure is terminated before it has achieved the optimum, it yields a bound on the error committed. (This is in sharp contrast to "heuristic problem-solving procedures" which lay no claim to either optimality or to measuring the error committed at premature abortion.)

A more formal definition of the B&B procedure was advanced by Mitten [8.39] in 1970 which was expanded upon in later work in 1973 by Mitten [8.40], and Mitten and Warburton [8.42]. Mitten defines the operations of "branching," "bounding," and "branch-and-bounding" in terms of set functions. The necessary properties of each function were given in terms of operator and operands, which map all the known concepts of B&B into topological domains. He establishes the relations between the B&B recursive function and the set of optimal feasible solutions by postulating various analytic and topological conditions such as continuity, completeness, and compactness. In the case of finite solution space, the convergence of the B&B recursive function is easily seen. However, in denumerable or nondenumerable spaces, Mitten demonstrates that if the B&B recursive function is a contraction mapping in a complete metric space with appropriately defined elements, then fixed-point theorems could be invoked to establish convergence.

To gain more insight into Mitten's construction, we assume that it is desired to solve the problem: maximize $f(x)$ for $x \in X$. (For example, X may be the integer feasible points in an ILP.) Typically, B&B

proceeds by searching the space $T \supseteq X$ for the set $\sigma^* \triangleq \{x \in X: f(x) = f^*\}$ of optimal solutions, where $f^* = \sup_{x \in X} f(x)$ and where $\triangleq$ means "is defined to equal." The search proceeds by examining subsets $\sigma \in X$, and collections of such subsets. Let S denote the family of all possible collections of subsets that could be encountered by a given B&B procedure. As shorthand notation, let $\cup(s)$ denote a subset of X comprised of the elements in $\bigcup_{\sigma \in s} \{\sigma\}$; that is

$\cup(s) \triangleq \bigcup_{\sigma \in s} \{\sigma\}$ where $s \in S$.

As mentioned above, alternative possibilities in B&B are considered in sets rather than one at a time. Furthermore, B&B examines successively smaller and smaller subsets of X (the subset formation operation), always eliminating those subsets that can be shown not to contain an optimal solution (the elimination operation). It is assumed that once sets are "small enough" in some sense, then there is a procedure available for distinguishing the optimal solutions from the nonoptimal solutions, the so-called fathoming procedure. Therefore, let S^- denote the set of fathomable collections $\{s\}$; here s is a fathomable collection iff $\sigma \in s$ satisfies $\sigma \subset \sigma^*$ or $\sigma \cap \sigma^* = \phi$. We assume that a procedure is available for separating one from the other. That is, $s \in S^-$ iff the following hold.

(a) $s = s_1 \cup s_2$ with $\sigma \subseteq \sigma^*$ for every $\sigma \in s_1$ and $\sigma \cap \sigma^* = \phi$ for every $\sigma \in s_2$

(b) There is a means available for forming the collections s_1 and s_2 .

As a minimum requirement, we insist that any collection of singleton sets (sets containing one point of X each) is fathomable, since such sets cannot be subdivided. One may now state the objectives as: find a collection $s^* \in S$ such that $\cup(s^*) \subseteq \sigma^*$ and

$\bigcup(s^*) = \phi$ only if $\sigma^* = \phi$. Mitten defines the B&B procedure in terms of the set operations called branching, upper bounding, lower bounding, and, finally, branch-and-bounding. Let S_F (for formation) and S_E (for elimination) be two subfamilies of S. Branching may be defined as a function $F: S_F \to S_E$ such that for each $s \varepsilon S_F$ the following hold.

(a) $F(s) = \bigcup_{\sigma \varepsilon s} \{d(\sigma)\}$, where $d(\sigma)$ is either σ or a collection of proper subsets of σ whose union is σ

(b) $F(s) = s$ iff $s \varepsilon S^-$

In words, this latter condition (a) states that each σ in s either remains unchanged under $F(s)$ or is broken up into a collection of proper subsets. This is illustrated in Figure 8.1, in which $s = \{\sigma_1, \sigma_2, \sigma_3\}$; $d(\sigma_2) = \sigma_2$; $d(\sigma_3) = \sigma_3$, but σ_1 has been "broken up" into four (disjoint) subsets. Clearly, $\bigcup_j \sigma_{1j} = \sigma_1$.
Upper bounding is a real-valued function $u: \bigcup(S_E) \to \underline{\underline{R}}$ with the following properties.

(a) $u(\sigma) \geq f(x)$ for all $x \varepsilon \sigma \varepsilon \bigcup(S_E)$

(b) $u(\sigma) \geq u(\sigma_0)$ if $\sigma_0 \subseteq \sigma$; $\sigma_0, \sigma \varepsilon \bigcup(S_E)$

(c) $u(\{x\}) = f(x)$, $x \varepsilon \sigma$

These latter conditions (a) and (b) follow from the common concepts of upper bounds and set inclusion. Condition (c) ensures that the upper bound on singleton subsets is the "value" of that point under the mapping f. Lower bounding is a real-valued function $\ell: S_E \to \underline{\underline{R}}$ such that the following hold for any $s \varepsilon S_E$.

(a) $\ell(s) \leq f^*$

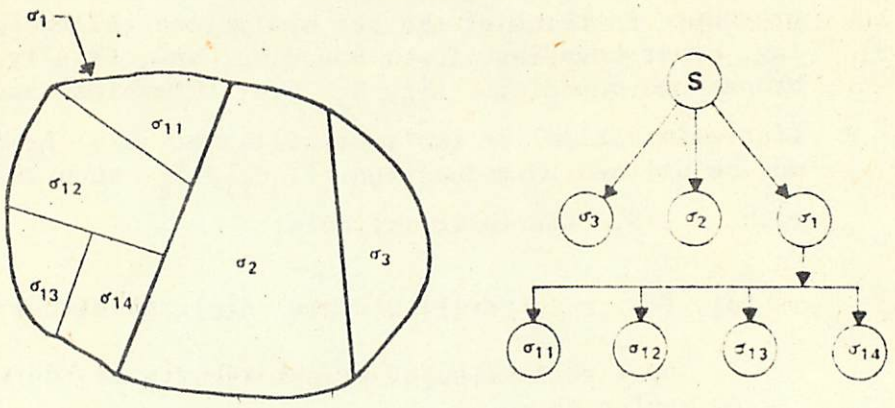


Figure 8.1 - Partitioning and branching.

(b) $\ell(s) \leq \ell(F(s))$

(c) $\ell(s) \geq f(x)$ for any $x \in s$

(d) If $s' \subset s$ is such that, for every $\sigma' \in s'$ either $u(\sigma') = -\infty$ or $u(\sigma') < \ell(s)$, then $\ell(s') = \ell(s)$.

These latter conditions (a) and (b) follow from the common concepts of lower bound and set division into subsets. Condition (c) ensures that the lower bound of a singleton subset is tight. Condition (d) guarantees that infeasible sets $(u(\sigma') = -\infty)$ or dominated sets $(u(\sigma') < \ell(s))$ cannot affect the value of the lower bound $\ell(s)$.

In Mitten's view, the bounding operation is an elimination operation through infeasibility and dominance. He defines it as a function $E: S_E \to S_F$ defined for $s \in S_E$ by

$E(s) = s - \{\sigma\varepsilon s: u(\sigma) = -\infty \text{ or } u(\sigma) < \ell(s)\}$

The strict inequality in the above statement implies strong bounding, since all optimal solutions in σ^* are retained. If an inequality is substituted, the resulting bounding operation is said to be weak since we then ensure that at least one element of σ^* will be retained. Finally, the B&B recursive operation is a function $G: S_F \to S_F$ defined by $G(s) = E(F(s))$. In other words, the successive formation and elimination of subsets is the heart of the procedure, hopefully leading to an optimum without the need to enumerate all singleton subsets.

If X contains finitely many points, it can be shown that $s^n = G(s^{n-1})$, for some finite $n > 1$, is an element of the fathomable set S^- , so that the procedure will terminate in a finite number of iterations. In the case X is not finite, Mitten shows that G will not "cycle" provided that each collection in S_F and S_E contains only finitely many sets. (Cycling means that there exists an m such that $G^m(s) = s$ and $s \varepsilon S_F - S^-$.) Note that even though a procedure may never cycle, it may not terminate in a finite number of steps. With this formal structure established, Mitten proceeds to illustrate his concepts by two examples: ILP and sequential unimodal search. This latter illustration is interesting since it claims to be the following.

(i) An example in which neither the procedure nor the sets involved are finite.

(ii) The only currently known application of B&B methods employing a branching rule that can be demonstrated to be optimal (the Fibonacci search).

This led Mitten to the following two conclusions. First, that the existence of an optimal branching rule for the sequential unimodal search suggests some interesting avenues of investigation in other areas of application. Second, that since attempts to extend the sequential search method to higher dimensions $\underline{\underline{R}}^n (n>1)$ have been notably unsuccessful, perhaps a fresh attack on the problem in $\underline{\underline{R}}^n$ via B&B would provide a new perspective. We wish only to remark that viewing sequential unimodal search as an application of B&B may raise some eyebrows, since none of the concepts usually associated with B&B are present in the standard search pattern, including the optimal pattern. The viewing is justified, however, if one sticks to the formal definition of B&B's search as composed of set formation and set elimination, both of which are indeed present in sequential unimodal search.

8.3 Branching

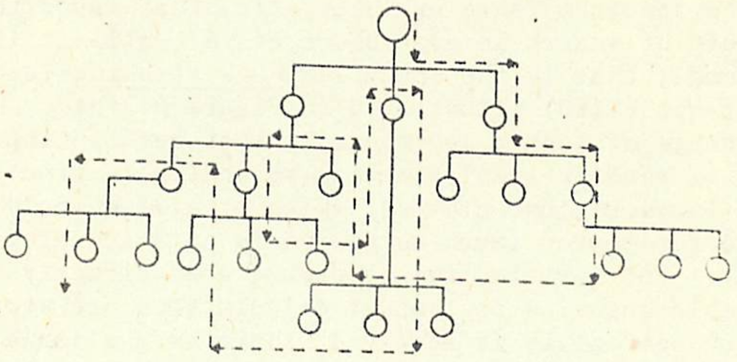
Branching proceeds by dividing the solution space into subspaces, which are themselves divided into subspaces, and so forth, until subsets containing exactly one point each are reached. The graphic representation is a tree, the search tree, whose numbering runs opposite to the set content. Thus, S_0 is the empty set ϕ which represents in fact the whole space S before any division has taken place. A terminal node of the tree, S_M, M large, contains a complete solution X which represents in fact a singleton set. Intermediate nodes of the search tree generally represent partial solutions generically represented by S_k. Hereafter we use the terms "branching" and "dividing the solution space" synonymously. The choice of the node from which to branch is basically a decision related to the philosophy of searching the tree, which is open to the use of heuristics.

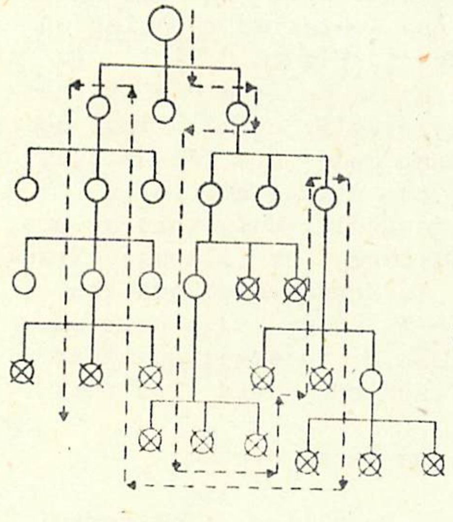
Basically, there are two extreme philosophies with innumerable intermediate variations. On the one end of the spectrum there are heuristics (for example, branch from the node with the smallest lower bound) that favor the nodes higher up the tree. In this case, the construction of the search tree will proceed "horizontally"; this is the so-called jump-tracking (or "flooding") illustrated in Figure 8.2(a). The advantage of this procedure is its economy in the number of singleton sets created prior to the determination of the optimum. Its disadvantage lies in the vast memeory required to store all the unbranched-from nodes. At the other end of the spectrum there are heuristics that favor the pursuit of search in one subset of S until it is fathomed; that is the so-called back-tracking last-in-first-out (LIFO) illustrated in Figure 8.2(b). The advantage of such a procedure is that certain information is readily available when branching is from the node (subset) just created, which otherwise would need to be recomputed (such as the basis of a linear program). Furthermore, the procedure goes directly to a feasible solution so that if calculations are stopped before optimality is achieved, there is available a feasible solution as well as an upper (or lower) bound on the optimum value.

Other procedures may be adopted which fall between these two extremes, such as the so-called choosing up the tree procedure illustrated in Figure 8.2(c). In this case, branching always continues from one of the m nodes just created until eventually either final nodes are obtained, or all descendant nodes are infeasible, or their lower bounds exceed the actual cost of a known solution (they are dominated). When this occurs, the next intermediate node is chosen as follows. Track up the tree until a node ξ is found which has the property that one of the $m - 1$ other nodes created when branching took place from ξ is still an intermediate node. Branching is then continued from this intermediate node.

We are now able to state our first dictum.

I. The Branches Need Not Be a Partition. The definition of the partitioning procedure is synonymous with the definition of the branching procedure. In the majority of cases there are several ways by which S may be partitioned: the decision is basically a heuristic one. Different procedures lead to different numbers of subsets of any subspace S_k and, consequently, different numbers of "stages" or "levels" of the search tree. The concept we wish to advance here as Dictum I is that the division of a subspace S_k


(a) Jump-tracking


(b) Back tracking
⊗ Fathomed nodes

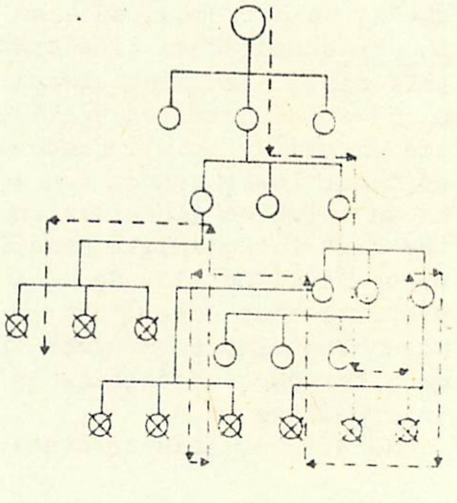

(c) Choosing up-the-tree
⊗ Fathomed nodes

Figure 8.2 - Three patterns of search in branching.

into subsets $S_{k1}, S_{k2}, \ldots, S_{kr}$ need not be a partition in the sense that $S_{ki} \cap S_{kj} = \phi$. This is true because the subsets may have some points in common. We illustrate this concept with two examples.

Example 8.1. Consider the problem of scheduling N jobs on M_t identical machines available in period t (say day t), $t = 1,2,\ldots,H$, where H is the "planning horizon." The jobs are related by precedence constraints. A job j has: processing time y_j during which it occupies one machine uninterrupted; a desired completion time (the so-called "due date") d_j; and a cost $c_j(T_j - d_j)$ which is a function of the difference between the actual completion time of the job T_j and its due date. (The function c_j is quite general except for the mild restriction that it be nondecreasing away from d_j.) It is desired to find the schedule of the N jobs with minimum total cost.

An ILP model was advanced by Elmaghraby [8.13]. Recognizing the computational difficulties entailed in a frontal attack, he proposed a B&B procedure in which the "levels" of the search tree correspond to the jobs, and a subspace S_k defines a partial solution in which the first $k - 1$ jobs have specified start times. Notationally, this is given by $\lambda_{is_i} = 1$ for $i = 1,2,\ldots,k-1$ in which λ_{it} is a 0,1 variable denoting the start or non-start of job i in period t and s_i represents the start time of i. Because of the precedence constraints, let a_k denote the earliest availability of job k (its earliest start time), and b_k its latest completion time (an "absolute" deadline beyond which the job may not be completed). Clearly, $\lambda_{ks_k} = 1$ for some value of s_k in the interval $a_k \le s_k \le b_k - y_k + 1$. Hence the subspace S_k is partitioned into $b_k - a_k - y_k + 1$ subsets, which may be represented by branches in the search tree as shown in Figure 8.3. Note that some of these subsets may be infeasible due to the machine availability constraints, in which case the subsets may be eliminated from further considerations.

An alternative approach may run as follows. Let the levels of the search tree correspond to time periods. In any period t, a number of jobs, say $n \le N$, are eligible for being started (by virtue of the precedence relations). The machine availability constraints may limit the number of jobs that can be started simultaneously to (the binomial coefficient) $Q = C(n, M_t - r)$ alternatives, where $r < M_t$ is the number of machines "committed" in period t as a consequence of the partial scheduling of the first $k - 1$ jobs. Suppose we enumerate all such feasible "bunches" of activities, and denote them by $S_{t,1}, S_{t,2}, \ldots, S_{t,Q}$. Then we may branch to $Q + 1$ subsets corresponding to the Q ways in which activities may be initiated at time t, plus the state in which none are initiated.

Comparing the two procedures, it is evident that in the latter the various subsets need not be mutually

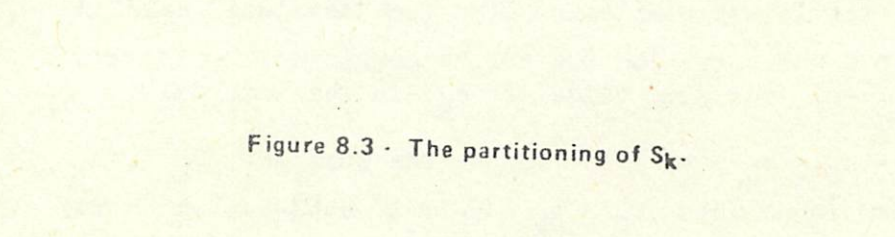


Figure 8.3 · The partitioning of S_k.

exclusive and hence the subdivision of S_k is not a partition. Moreover, the first procedure leads to a tree of N levels, while the second to a tree of H levels.

Example 8.2. Consider the well-known traveling salesman problem (TSP) over N cities, modeled by the following ILP.

$$\text{minimize} \quad \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1 \quad \text{for all } i \text{ , } \sum_i x_{ij} = 1 \quad \text{for all } j$$

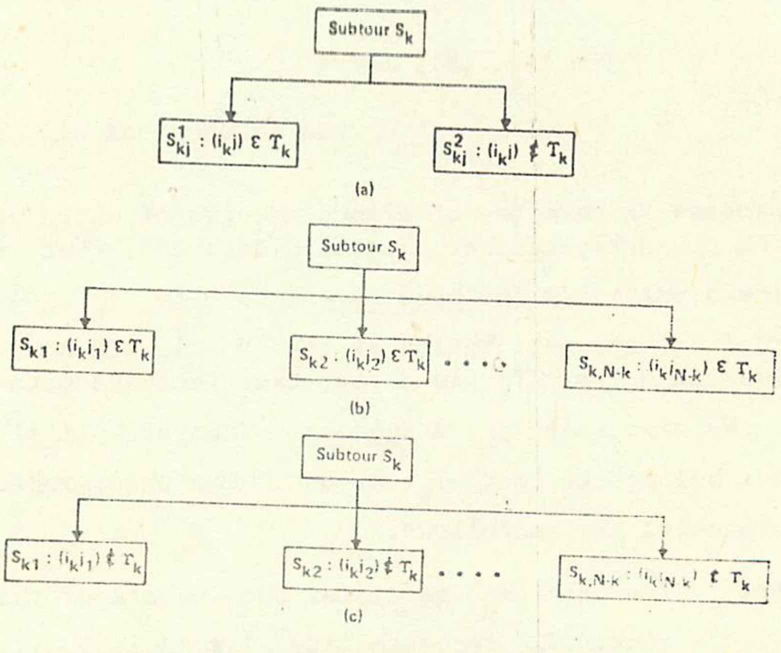
$$u_i - u_j + N x_{ij} \le N - 1 \text{ , } i = 2,\dots,N \text{ ,}$$

$$j = 2,\dots,N \text{ , } i \ne j$$

$$x_{ij} = 0,1 \text{ , } u_i \ge 0 \quad \text{and integer for all } i$$

We propose three modes of branching, two of which partition the subspaces but the third does not. Let S_k define a path from the home city 1 to city i_k. A tour T is any permutation of the $N - 1$ non-home cities and a tour T_k is a tour that includes path S_k. We also call S_k a subtour. Then arc $(i_k j)$ either belongs to tour T_k or not. The three modes of branching are as follows.

(i) The path S_k generates two subsets of the tours T_k for each city $j \notin \{1,i_1,i_2,\dots,i_k\}$: the tours wherein S_k is extended by continuation on arc $(i_k j)$ which we denote by S^1_{kj} ; and the tours denoted by S^2_{kj} wherein S_k appears but arc $(i_k j)$ does not. Figure 8.4(a) illustrates this mode of branching which is basically the branching rule proposed by Little et al. [8.37]. The branch S^1_{kj} eliminates from consideration (due to infeasibility) the row i_k in the distance matrix and any other entry (ji_r) that would form a subtour with the current partial schedule $(i_r \varepsilon \{1, i_1, \ldots, i_k\})$. On the other hand, the branch S^2_{kj} simply eliminates the entry $(i_k j)$ in the distance matrix. Recalling the bounding method of [8.37], it is obvious that the positive assertion of S^1_{kj} is more potent than the negative assertion of S^2_{kj}.



Subtour S_k
$S^1_{kj} : (i_k j) \varepsilon T_k$
$S^2_{kj} : (i_k j) \notin T_k$
(a)
Subtour S_k
$S_{k1} : (i_k i_1) \varepsilon T_k$
$S_{k2} : (i_k i_2) \varepsilon T_k$
$S_{k,N-k} : (i_k i_{N-k}) \varepsilon T_k$
(b)
Subtour S_k
$S_{k1} : (i_k i_1) \notin T_k$
$S_{k2} : (i_k i_2) \notin T_k$
$S_{k,N-k} : (i_k i_{N-k}) \notin T_k$
(c)

Figure 8.4 - Three modes of branching.

(ii) Still assuming that S_k defines a path to city i_k, specify the subsets according to the next arc included in the tour T_k. Thus $S_{k1} = S_k \cap (i_k j_1)$; $S_{k2} = S_k \cap (i_k j_2), \ldots,$ $S_{k,N-k} = S_k \cap (i_k j_{N-k})$, and there are as many branches from S_k as there are cities still to be visited. This mode is illustrated in Figure 8.4(b).

(iii) Specify the subsets of S_k by the arcs not in the tour. Then we write S_{k1} to denote the set of all tours T_k in which the arc $(i_k j_1)$ does not appear; S_{k2} denotes the set of all tours T_k in which the arc $(i_k j_2)$ does not appear; and so on. Clearly this is not a partition of the tours T_k since, for instance, both subsets S_{k1} and S_{k2} contain all completions of the subtour S_k which contain neither city j_1 nor city j_2. Figure 8.4(c) illustrates this mode.

II. The Desirability of Nonredundancy of Completions. Undoubtedly, a desiratum would be that the branching scheme generates a sequence of nonredundant partial solutions; that is, that no completion of a partial solution in the sequence ever duplicates a completion of a previous partial solution that was fathomed. To heed this second dictum, it is obviously necessary and sufficient to have in all future subsets S_v , $v > k$, at least one element "complementary" to one in S_k . This, in turn, is indeed satisfied if we store S_k and generate the new subset S_{k+1} to be exactly S_k but with its last element the complement of the last element of S_k , and indicate in some fashion that S_k was fathomed. (The storing of S_k is to comply with the requirement of (implicitly) enumerating all points in the solution space.) Compliance with this dictum is extremely difficult, and is rarely accomplished except in those instances where "complementarity" is obvious, such as 0,1 ILPs. The alternative to modeling the problem as a 0,1 ILP is to store all subsets in the tree (notjust the unbranched-from subsets) and compare them to each newly-generated subset to eliminate duplication. (This is of fundamental importance in dynamic programming. In some sense, B&B relaxes this requirement, and the price paid for such relaxation is the possibility of duplication.) Our example to illustrate this concept is indeed taken from solutions to 0,1 ILPs.

Example 8.3. In the Geoffrion-Glover Approach to 0,1 ILP [8.26, .29], the problem is stated as follows for $c > 0$.

$$\text{minimize } cX$$

$$\text{s.t. } AX + b \geq 0 \ ; \ x_j = 0,1 \ .$$

An S_k is a set of variables whose values have been assigned as either 0 or 1. The remaining variables are called "free variables." Suppose $S_k = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$, and S_k is fathomed. Then create $S_{k+1} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{k-1}}, \underline{X_{i_k}}\}$ where $X_{i_k} = 1 - x_{i_k}$, and the underlining of X_{i_k} is simply a visual indication that S_k has been fathomed. The following illustrates the two concepts of fathoming and nonredundant completion. The general logic may be shown schematically as in Figure 8.5. The application of this logic to the ILP proceeds as follows where the step number refers to the box number in Figure 8.5.

1. The most effortless completion of S_k, ignoring feasibility, is to put $x_j = 0$ for all the free

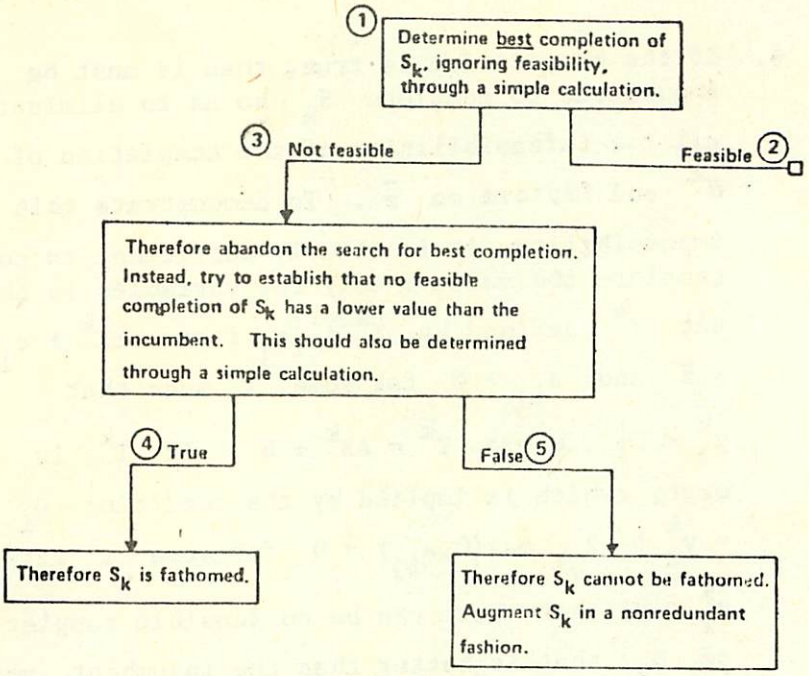

1
Determine best completion of S_k, ignoring feasibility, through a simple calculation.
2
Feasible
3
Not feasible
Therefore abandon the search for best completion. Instead, try to establish that no feasible completion of S_k has a lower value than the incumbent. This should also be determined through a simple calculation.
4
True
5
False
Therefore S_k is fathomed.
Therefore S_k cannot be fathomed. Augment S_k in a nonredundant fashion.

Figure 8.5 - General logic for fathoming.

variables. (Recall that we assumed all $c_j > 0$.)

2. Clearly, if the resultant solution is feasible, then indeed it is the best completion of S_k. Its value is easily determined, say z_k. If $z_k \leq \bar{z}$ (the value of the incumbent, assuming one exists; otherwise $\bar{z} = \infty$), then we adopt the current feasible solution as the best, and put $\bar{z} = z_k$. Otherwise, $z_k > \bar{z}$ and, a fortiori, no feasible completion of S_k has a lower value (of the objective function) than the incumbent; hence S_k is dominated, and again it is fathomed.

3. If the completion is not feasible, then instead of seeking the best completion of S_k (which may require extensive calculations) we try to establish the proposition stated in node 3.

4. If the proposition is true, then it must be impossible to complete S_k so as to eliminate all the infeasibilities of the completion of S^k and improve on $\bar{z}$. To demonstrate this impossibility, it is clearly sufficient to contemplate the value 1 only for variables in the set T^k defined by $T^k \triangleq \{x_j$ free: $cX^k + c_j < \bar{z}$ and $a_{ij} > 0$ for some i such that $y_i^k < 0\}$, where $Y^k = AX^k + b$. If T^k is empty (which is implied by the condition $D_i = y_i^k + \Sigma_{j \varepsilon T^k} \max(0, a_{ij}) < 0$ for some i with $y_i^k < 0$) then there can be no feasible completion of S_k that is better than the incumbent, and S_k is fathomed.

5. If the proposition is false, then S_k cannot be (easily) fathomed. Then the partial solution S_k must be augmented. Here, heuristics are adopted such as: add the variable that reduces total infeasibility the most; or add the variable that reduces infeasibility in the most number of constraints; or add the variable with the smallest $c_j > 0$ that still reduces infeasibility of at least one constraint; and so on. The application of these concepts to an ILP is given in the tree diagram contained in Figure 8.6. There the variables are denoted by their numbers; writing j as, for example in $S_3 = \{3,\underline{-2}\}$ means that $x_j = 1$ while $-j$ means $x_j = 0$; the underlining of a variable means that its complement was fathomed. Notice finally that the first of the stated procedures in 5. is used in Figure 8.6.

<u>III. Alternate Criteria for Branching</u>. In relating

$$\begin{array}{ll} \text{minimize} & 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5 \\ \text{s. t.} & x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 - 2 \geq 0 \\ & -2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0 \\ & - x_2 + 2x_3 - x_4 - x_5 - 1 \geq 0 \\ & x_j \geq 0;\ \text{integer} \end{array}$$

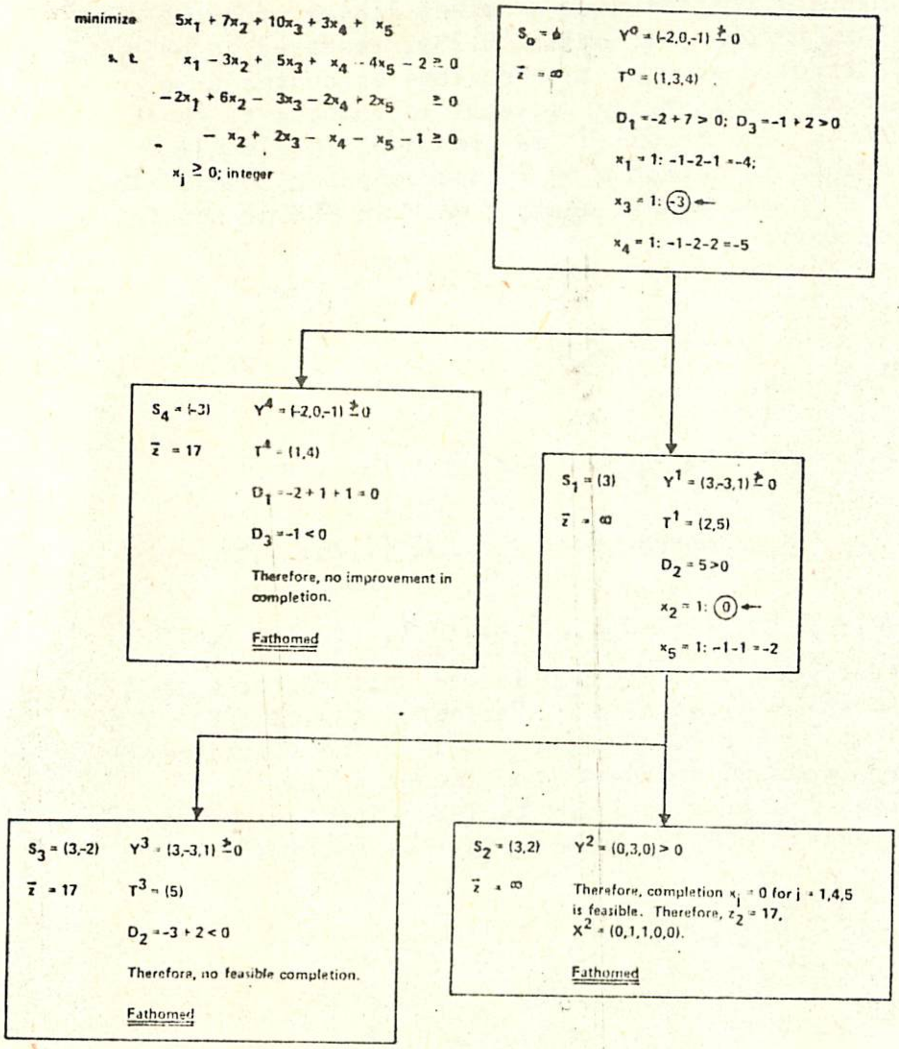

$S_o = \phi$ $Y^o = (-2,0,-1) \not\geq 0$
$\bar{z} = \infty$ $T^o = (1,3,4)$
$D_1 = -2+7>0;\ D_3 = -1+2>0$
$x_1 = 1: -1-2-1 = -4;$
$x_3 = 1: \boxed{-3} \leftarrow$
$x_4 = 1: -1-2-2 = -5$
$S_4 = (-3)$ $Y^4 = (-2,0,-1) \not\geq 0$
$\bar{z} = 17$ $T^4 = (1,4)$
$D_1 = -2+1+1 = 0$
$D_3 = -1 < 0$
Therefore, no improvement in completion.
Fathomed
$S_1 = (3)$ $Y^1 = (3,-3,1) \not\geq 0$
$\bar{z} = \infty$ $T^1 = (2,5)$
$D_2 = 5 > 0$
$x_2 = 1: \boxed{0} \leftarrow$
$x_5 = 1: -1-1 = -2$
$S_3 = (3,-2)$ $Y^3 = (3,-3,1) \not\geq 0$
$\bar{z} = 17$ $T^3 = (5)$
$D_2 = -3+2 < 0$
Therefore, no feasible completion.
Fathomed
$S_2 = (3,2)$ $Y^2 = (0,3,0) > 0$
$\bar{z} = \infty$
Therefore, completion $x_j = 0$ for $j = 1,4,5$ is feasible. Therefore, $z_2 = 17$, $X^2 = (0,1,1,0,0)$.
Fathomed

Figure 8.6 - Solution of an ILP.

their experience with UMPIRE, a proprietary computer package for the solution of mixed linear programs (MLP), Forrest, Hirsh, and Tomlin [8.23], referred to below as FH&T, treated some of the problems encountered in branching and suggested several approaches to their solution. Their insight may prove of invaluable assistance in structuring future computer codes. To set the stage, we are dealing with an MLP of the following form.

Program Q

maximize x_0

$$\text{s.t.} \quad Ax = b, \; x_j \geq 0 \quad \text{for} \quad j = 1,\ldots,n$$

$$x_j \quad \text{integer for} \quad j \in I \subset \{1,2,\ldots,n\}$$

where $A = [a_{ij}]$ $(i=0,1,\ldots,m; j=0,1,\ldots,n)$, row 0 is the cost row, and column 0 is the unit vector with 1 in the first (0th) position. Variables may be subject to simple or generalized upper bounds. The solution of the above MLP as an ordinary (continuous variable) LP results in a final simplex tableau which may be stated (in standard simplex terminology) as follows.

$$x_0 = \bar{a}_{00} + \sum_j \bar{a}_{0j} \; (-x_j)$$

$$x_i = \bar{a}_{i0} + \sum_j \bar{a}_{ij} \; (-x_j) \quad \text{for} \quad i = 1,\ldots,m$$

By the simplex criterion for optimality, we have

$\bar{a}_{i0} \geq 0$ and $\bar{a}_{0j} \geq 0$ for all i and j. For every $i \in I$, let

$$\bar{a}_{i0} = \left[\bar{a}_{i0}\right] + f_{i0}, \; 0 \leq f_{i0} < 1$$

where $\left[\bar{a}_{i0}\right]$ is the largest integer not exceeding $\bar{a}_{i0}$.

The solution of Program Q by B&B involves maintaining a list of LP problems or subproblems derived from the original MLP, obtained by imposing tighter bounds on the integer variables and always recording the best integer solution obtained so far and its value x_0^c. The basic steps in branching are the following.

1. Problem (Node) Selection. Select a problem with fractional values from the list whose objective function satisfies $x_0 > x_0^c$. If none exists, terminate.

2. Choose A Branching Variable (the "Arbitrated" variable). Among the variables in the selected problem, choose a fractional variable for branching. Denote the chosen variable by x_p.

3. Branch (Arbitrate). Create two new subproblems (nodes) by adding the following new restrictions.

$x_p \leq \left[\bar{a}_{p0}\right]$ to yield subproblem p1

$x_p \geq \left[\bar{a}_{p0}\right] + 1$ to yield subproblem p2

The procedure is "straightforward" except for the fact that each step is in need of operational definition (which is the subject of the FH&T paper).

Consider Step 3 first. One may solve each descendant subproblem, or one may solve only one of the two LPs, postponing the solution of the other to later in the hope that it never need be done because of dominance considerations. Alternatively, one may "jockey" between the two new descendant subproblems p1 and p2; the authors refer to the alternation between the two nodes as "node swapping." For example, one may investigate the other subproblem as soon as the degradation in the first branch is found to exceed the "penalty" of the second as explained below.

Three questions are to be resolved before Step 2 is successfully executed. How does one choose the branching variable? How does one decide on which branch to postpone (if branch postponement is desirable)? How does one bound its solution? To this end, FH&T discuss the following approaches.

(i) The Penalties Method. Penalties give a lower bound on the change (degradation) in the objective function as a consequence of forcing a currently non-integer variable to its adjacent integer values. From the theory of parametric LP it is evident that, assuming no change in basis, the imposition of a new ℓ.b. of $\left[\bar{a}_{p0}\right] + 1$ on x_p must decrease the objective function at least by the "up penalty"

$$U_p = \min_{j,\bar{a}_{pj}<0} (1-f_{p0})\, \bar{a}_{0j}/\left(-\bar{a}_{pj}\right)$$

If $\bar{a}_{pj} \geq 0$ for all j, let $U_p = \infty$ and x_p is a monotone decreasing variable. Similarly, the minimum "down penalty" incurred by placing an upper bound $\left[\bar{a}_{p0}\right]$ on x_p is given by

$$D_p = \min_{j,\bar{a}_{pj}>0} f_{p0}\, \bar{a}_{0j}/\bar{a}_{pj}$$

If $\bar{a}_{pj} \leq 0$ for all j, let $D_p = \infty$, and x_p is a monotone increasing variable. The penalties are lower bounds on the decrease in x_0 because we assumed no change in basis; hence the value of the objective function for these two branches must be bounded from above by $\bar{a}_{00} - U_p$, $\bar{a}_{00} - D_p$, respectively. If the descendant node is not dominated as a consequence of the new bound, then the above penalty calculations may be used in the selection of the subproblem to be solved, postponing the other to later. Presumably, the subproblem giving the smallest degradation is the one selected.

The authors argue against this method since in many instances it leads to the wrong decision, thus prolonging the search. Garfinkel and Nemhauser [8.25] recommend the capitalization on monotone variables to reduce the size of the search tree. In particular, if row p is chosen as the partitioning row and x_p is monotone (increasing or decreasing), then it will have only one successor, and one need only to consider $x_p > \left[\bar{a}_p\right] + 1$ or $x_p > \left[\bar{a}_{p0}\right]$. They also point out, correctly, that the penalties U_p and D_p were derived without taking into account the integrality requirement on the nonbasic variables. Taking such requirements into account would generate new bounds on the penalties. For instance, in order to have an integer solution, some nonbasic variable must become positive and therefore not less than one. This immediately yields the ℓ.b. on penalty $\min_j a_{0j}$; which, incidentally does not depend on the partitioning row. One may wish to carry the idea of penalties a little further and determine a stronger bound on the penalty incurred. Two approaches suggest themselves.

(a) Assume that the current basis does not change, and determine a feasible solution when x_p is rounded up or down.

(b) Assume that the basis will change with the introduction of some nonbasic variable at a positive level. Determine the cheapest such transformation that retains primal feasibility, and its associated cost.

Naturally, the price paid for improved bounds is the additional computing. The efficacy of such approach is currently under investigation by the authors. Also see Breu and Burdet [8.6].

(ii) The Method of Priorities. Priorities are accorded to variables a priori, and are based primarily on the analyst's knowledge of the physical problem. Then one would select and branch on that variable not within some tolerance of an integer value (the "arbitration level") which has the highest priority.

(iii) The Best Projection Method. While this method is more appropriately related to Step 1 (Node Selection) rather than Step 2 (Variable Selection), it finds its place here because of the bearing it has on the next method of variable selection. The logical justification of this method is rather lengthy, albeit intuitively appealing. But its statement is rather simple. Suppose that an estimate of x_0^* can be made. Let s denote the "sum of integer infeasibilities,"

$$s \triangleq \sum_{j \in I} \min(f_{j0};1-f_{j0})$$

and let

$$\lambda \triangleq \frac{x_0^0 - x_0^*}{s^0} \geq 0$$

where x_0^0 is the value of the objective function of Program Q when solved as LP, and where s^0 is its corresponding sum of integer infeasibilities. (Note that the x_0's measure degradation from Program Q.) Then, for any node k with objective value x_0^k and sum value s^k, we define the "projection"

$$p^k = x_0^k - \lambda s^k$$

The best projection (BP) criterion for Step 1 of the B&B algorithm is now to choose the outstanding node with the largest value of p_k. The rationale for this is that p_k measures the approximate value of the integer solution we can expect to attain from node k.

The term "projection" stems from the fact that λ essentially projects the sum of integer infeasibilities s^k on the $s = 0$ axis in a particular direction (namely, a direction parallel to the line joining the points $\left(s_0^0, x_0^0\right)$ and $\left(0, x_0^*\right)$ in the $s - x_0$ domain).

(iv) The Pseudo-Costs Method. A close scrutiny of the definition of λ in the above method reveals that it can be interpreted as the "cost" of removing one unit of infeasibility. In fact, the last equation may be rewritten as

$$x_0^k - \lambda s^k = x_0^k - \sum_{i \in I} \min\{\lambda f_{i0}; \lambda(1-f_{i0})\}$$

Hence, in using this expression to estimate an obtainable integer solution we are implicitly "costing" the change in the variable at the same cost per unit change (whether up or down). Since this may not be true in general, we are led to the new estimate

$$e^k = x_0^k - \sum_{i \in I} \min\{d_i f_{i0}; u_i(1-f_{i0})\}$$

where d_i and u_i are the estimated costs per unit decrease or increase in variable x_i, respectively. The determination of the values d_i and u_i, as well as their revision as the iterations proceed, are discussed at length in the paper of FH&T.

(v) The Percentage-Error (P.E.) Method. We have the definition

$$\text{P.E.} = 100\left(x_0^c - e^k\right) / \left(x_0^k - x_0^c\right)$$

for each node k. Essentially, it measures the degree of error if the current solution x_0^c is not optimal. If the P.E. is large and positive, it implies that a better integer solution is very unlikely to be found from this branch.

8.4 Bounding

We are always interested in the greatest lower bound (least upper bound) in the case of minimization (maximization). Unfortunately, this is oftentimes achieved at the heavy price of extensive calculation. Hence, there is the ever-present trade-off between a tight bound obtained at a considerable cost, and a loose one that is easily calculated. The only dictum that can be stated relative to this choice is that it may be worthwhile to put the effort in bounding nodes "higher up the tree," because then if fathomed we would save the enumeration of all their descendants. On the other hand, it is always advisable to obtain both upper and lower bounds on the value of z^* and preferably the tightest such bounds. Since a feasible solution always provides an u.b. (a ℓ.b.) in minimization (maximization) problems, it behooves the analyst to start the search procedure after having obtained as "good" a solution as possible, without the expenditure of an inordinate amount of effort in obtaining such a solution. Bounds on the value of the optimum are obtained by relaxing one constraint, or several constraints, of the original problem since the optimum of the relaxed problem is a bound on z^*. In certain instances the constraint to be relaxed is almost self-evident--such as relaxing the integer requirements in an ILP problem and solving as an ordinary LP. This is the relaxation adopted by Land and Doig [8.35] in their pioneering work. Oftentimes, though, the constraint to be relaxed requires insight into the problem to gain the "most mileage," by removing the more complicating constraint, without too much sacrifice in the value of the objective function.

We illustrate the concept of bounding with the following three examples. The first is, more or less, straightforward; the second exemplifies how a bound may be improved, that is, made tighter; the third exemplifies the need for the judicious choice of the constraint to be relaxed.

Example 8.4. Consider the problem of minimizing the total "makespan" in scheduling N jobs on three machines in series. We shall develop the ℓ.b. established by Lomnicki [8.38], which was apparently arrived at independently by Ignall and Schrage [8.33]. It is well-known that for three machines an optimal schedule permits no "passing" and hence the order of jobs will be preserved on all three machines. Let $W = (w_1, w_2, \ldots, w_N)$ denote a permutation of the numbers $1, 2, \ldots, N$ and let $f(w_i, j)$ represent the "earliest finish time" of job w_i on machine j, $j = 1,2,3$. Let the processing time of job w_i on machine j be denoted by $y(w_i, j)$. Then, clearly, for a given permutation W,

$$f(w_i, j) = \max\ [f(w_{i-1}, j); f(w_i, j-1)] + y(w_i, j)$$

since to complete job w_i on machine j the time $y(w_i, j)$ must elapse after the machine became free from job w_{i-1}, or the job w_i became available from the previous machine $j-1$ whichever happens last. For ease of notation, denote $y(w_i, 1)$ by $a(w_i)$, $y(w_i, 2)$ by $b(w_i)$, and $y(w_i, 3)$ by $c(w_i)$. Consider any partial schedule $w_1, w_2, \ldots, w_k$ which specifies the sequence of the first k elements, $k < N$. Then it is evident that the completion of all the remaining jobs consumes no less time than any of the following three values.

$$g' \equiv f_W(w_k, 3) + \sum_{j=1}^{N-k} c(w_{k+j})$$

$$g'' \equiv f_W(w_k, 2) + \sum_{j=1}^{N-k} b(w_{k+j}) + \min_{1 \le j \le N-k} c(w_{k+j})$$

$$g''' \equiv f_W(w_k, 1) + \sum_{j=1}^{N-k} a(w_{k+j}) + \min_{1 \le j \le N-k} [b(w_{k+j})$$

$$+ c(w_{k+j})]$$

Lomnicki puts the lower bound $\max(g', g'', g''')$ on the partial schedule.

Example 8.5. Consider the problem of scheduling N jobs on M identical machines. Each job j has a fixed-processing time y_j and a penalty coefficient p_j. A penalty $p_j t$ is incurred if job j is completed at time t (in other words, all jobs have a due date equal to zero, and they accumulate penalty starting from that time). Eastman, Even, and Isaacs [8.10], referred to below as EE&I, derived a lower bound on the optimum as follows. Let C_i be the symbol for the optimal cost of scheduling the N jobs on i machines, $1 \leq i \leq M$. Thus C_1 is the known optimal cost on one machine and C_N is the known optimal cost on N machines. In particular, the minimal cost (of scheduling N jobs on a single machine when a linear penalty is accumulated starting at time zero) is given by scheduling the jobs in their natural order, that is, in order of nonincreasing values of the ratio p_j/y_j. If the jobs are so numbered (if we have $p_1/y_1 \geq p_2/y_2 \geq \cdots \geq p_N/y_N$) then the minimal cost is given by

$$C_1 = \sum_j p_j \sum_{i \leq j} y_i$$

On the other hand, C_N is evidently given by $\sum_j p_j y_j$. Then EE&I assert that the desired lower bound on C_M is given by

$$\frac{1}{M}\left[C_1 + \frac{M-1}{2} C_N\right]$$

A sharper ℓ.b. was developed by Elmaghraby and Park [8.16] for the slightly more generalized problem in which each job j has a due date $d_j = y_j$. Their development was based on the remark that EE&I's ℓ.b. is based on the assumption that all the machines are available at time zero. Clearly, given any partial schedule S_k of k jobs on the M machines, the times of earliest availability of each machine may be different from zero. This leads immediately to interest in developing a sharper ℓ.b. when the machines are available at time $T > 0$. To this end, let $C^{n,v}(T)$ denote the optimal cost of scheduling n jobs on v machines when all machines are available at time $T \geq 0$. Let T_j denote the time of completion of job j under the partial schedule S_k ; m_ℓ is the last job on machine m ; T_{m_ℓ} is the time of completion of the last job on machine m ; $T_{min} = \min_m T_{m_\ell}$; and, $\bar{S}_k = N - S_k$. Then a ℓ.b. on the cost of the completion of $\bar{S}_k$ is given by

$$\sum_{j\in S_k} p_j(T_j - d_j) + \max\left\{0; \frac{1}{M} C_{k-N,1}(T_{min}) + \frac{M-1}{M} T_{min} \sum_j p_j - \frac{M-1}{M} C_{N-k,N-k}(0)\right\}$$

where $C_{N-k,N-k}(0) = \sum_{j\in\bar{S}_k} p_j y_j$. The first term is the cost incurred in scheduling the k jobs in S_k . The partial schedule S_k leaves the M machines with earliest availabilities $T_{1_\ell}, T_{2_\ell}, \dots, T_{M_\ell}$. The smallest value of (earliest) availabilities is T_{min} , which was conservatively taken to be the availability of all machines. The second term is essentially EESI's ℓ.b. corrected for the jobs already included in S_k and the earliest availability of the machines.

Example 8.6. In treating the problem of scheduling lots on a single facility over a finite horizon, Elmaghraby and Mallik [8.14] addressed themselves to the following specific version. There are N items to be produced on the same facility. In any period (say, day or week), the facility may be devoted to the production of only one item. Item i is produced at the rate p_i per period, but is continuously consumed at the rate r_i per period where $r_i < p_i$. Given the initial "on hand" stock of each item, and the desired terminal inventory at the end of a planning horizon of length H, determine the minimum cost schedule (if one exists), where cost is defined in terms of inventory cost and backorder penalty, which vary from item to item. The constraint that Elmaghraby and Mallik chose to relax in their B&B approach is the noninterference constraint. Then the items are independent, which implies that the facility is devoted to the production of one item only. The determination of the optimal schedule under such an assumption is an intriguing problem in its own right, and was treated by Elmaghraby and Dix [8.15]. Fortunately, it proved to be of extremely simple form which requires a nominal amount of computing.

In relation to bounding, we wish to advance several dicta.

IV. The Use of the Previous ℓ.b. Calculations (for the Parent Node) as Lead-Off to the New ℓ.b. (of the Descendant Node). The pertinence of this concept increases with the amount of effort required in the calculation of the ℓ.b. The concept was used by Land and Doig [8.35] in their treatment of ILP, and by Elmaghraby [8.13] in his treatment of the problem of scheduling activities subject to resource constraints. In both instances the ℓ.b. at a node is determined by solving a (continuous) linear program. The ℓ.b. of a node is constructed from the parent optimal basis.

Another excellent example of capitalizing on the optimal solution of the previous iteration was provided by Srinivasan and Thompson [8.44] in their treatment of the traveling salesman problem (TSP) by the so-called "operator theory." The reader will recall that the TSP is a restricted assignment problem, restricted to assignments that are tours. Consequently, the search for the optimal tour in the TSP may be viewed as the search for the optimal assignment (in the assignment problem) that is a tour. (This is not the only way to interpret the TSP. For example, Held and Karp [8.31] interpret it as a one-tree problem, hence the search for the optimal tour in the TSP is reduced to the search for the optimal one-tree. B&B methods are then used, and they report excellent computing results.) By utilizing the established properties of parametric linear programming, specialized to the assignment problem, the authors achieve the capability of probing the consequences of increasing a nonbasic variable (at the expense of a basic variable) in the optimal solution of the assignment problem, without in fact undertaking such changes. Because of the ease with which bounds can be established on such probes, the authors report excellent computing results.

V. Choose an Easy-To-Calculate Bound. The value of this concept rests on the fact that bounds are evaluated a large number of times over the life of a search, and if it is time-consuming it will render the search impractical.

Example 8.7. Nowhere is this concept more apparent than in the solution of the (linear) knapsack problem by the so-called "Greedy Algorithm." The setting of the problem is as follows.

$$\text{maximize} \quad \sum_i v_i x_i$$

$$\text{s.t.} \quad \sum_i a_i x_i \le b \ , \ x_i = 0,1 \quad \text{for all} \quad i$$

and where the v_i , the a_i , and b are given positive integer constants. The rationale for the Greedy Algorithm is the well-known observation that, in the absence of the integer requirements, the optimum is readily obtained by renumbering the variables in order of non-increasing v_i/a_i , and putting $x_j = \min\left(1, b - \sum_{i=1}^{j-1} a_i\right)$, $j = 1,2,\ldots,n$. The Greedy Algorithm approach to the knapsack problem proceeds in exactly the same fashion, but then it branches on the fractional variable, denoted by x_r, thus creating two subsets corresponding to $x_r = 0$ and $x_r = 1$. Of these two nodes, we investigate (branch on) the node with the best (fractional) solution, which is an u.b. on the optimal value. In case of ties, apply any heuristic to break it, such as random choice among the tied nodes. Continue the process of dichotomizing the solution space; each time an integer solution is achieved it provides a new ℓ.b. on the value of the optimum (if it is better than the incumbent). If all the u.b.'s of the unbranched-from nodes are smaller than the current (integer) ℓ.b., it is also optimal. Otherwise, branching and bounding continues from other nodes until the optimum is achieved.

To illustrate, consider the following knapsack problem.

$$\text{maximize} \quad z = 5x_1 + 4x_2 + 3x_3 + 2x_4$$

$$\text{s.t.} \quad x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$$

$$x_j = 0,1 \quad \text{for all } j .$$

The search tree is shown in Figure 8.7. In this example we explicitly enumerated only 8 solutions out of the possible 16 solutions. In larger problems, the savings are, fortunately, more pronounced than in this small example.

<u>VI. Relax the Objective Function Instead of a Constraint</u>. Sometimes bounds are easily computed by relaxing the form of the objective function, rather than a constraint. Perhaps the following example illustrates this concept best.

<u>Example 8.8</u>. In the field of project planning and control, a problem that has been extensively studied is that of reducing the duration of a project (the so-called project "crashing" or "compression") at minimal cost. The optimal reduction under the assumption of

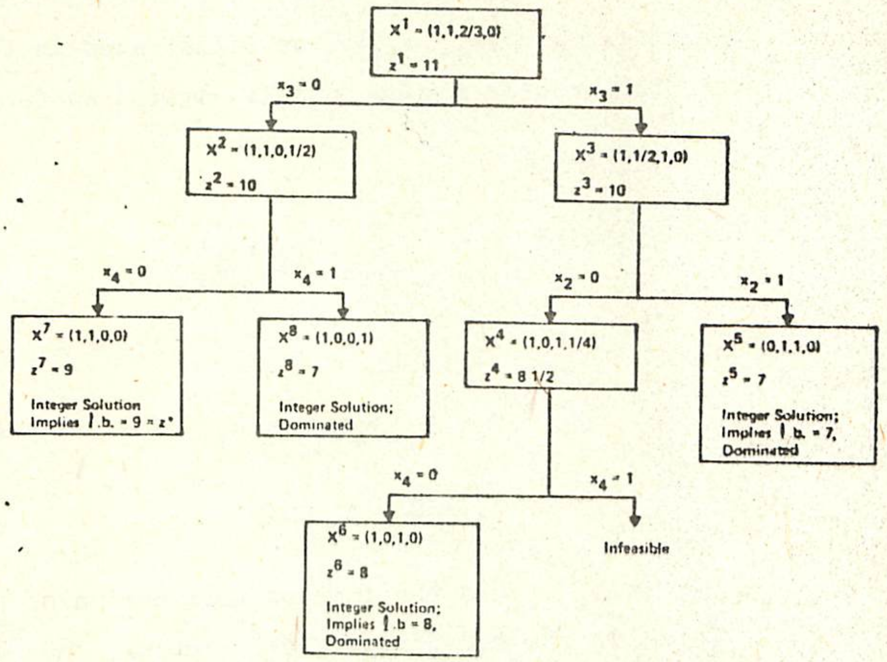


Figure 8.7 - Search tree for a knapsack problem.

linear or convex costs for the individual activities has been treated by several researchers, see Clark [8.7], Elmaghraby [8.12], and Fulkerson [8.24], among others, who used algorithmic approaches. However, the case of concave cost functions was treated by Falk and Horowitz [8.21] using a rather ingenious B&B approach, first proposed by Rech and Barton [8.43]. Their basic idea was to relax the objective function into the largest linear function that underestimates the concave cost of each activity. This reduces the problem to an LP problem, which can be easily solved; whose optimum is a ℓ.b. on the optimum cost desired. Also, by virtue of it being a feasible solution to the original problem, it also provides an u.b. More importantly, the LP solution suggests the partitioning of the solution space, the branching process, which is proved to terminate in a finite number of steps. Briefly, the procedure is as follows.

Let the duration of activity (ij) be denoted by y_{ij} , and assume its upper and lower limits are denoted by u_{ij} and ℓ_{ij} , respectively. The cost of activity

(ij) is given by $c_{ij}(y_{ij})$, assumed concave in the interval $[\ell_{ij}, u_{ij}]$, as illustrated in Figure 8.8. The problem may be formally stated as follows.

Program P

$$\text{minimize } C = \sum_{(ij)\varepsilon A} c_{ij}(y_{ij})$$

$$\left.\begin{aligned} \text{s.t. } & t_i + y_{ij} \leq t_j \quad \text{for } (ij) \ \varepsilon \ A \\ & 0 \leq \ell_{ij} \leq y_{ij} \leq u_{ij} \quad \text{for } (ij) \ \varepsilon \ A \\ & t_1 = 0 \ , \ t_n = T \end{aligned}\right\} F$$

Here, t_i is the time of realization of node i, A is the set of activities (arcs), and T is the specified duration of the projects. The approach is simply the following. Suppose that the concave cost function of Figure 8.8(a) is approximated by a linear function as shown by the dotted line in (b), which is the highest linear function which underestimates $c_{ij}(y_{ij})$. Denote such a linear cost function by $c^1_{ij}(y_{ij})$. Then we may take $C^1(Y)$ as a first (lower) approximation to the objective function of the Program P, and formulate the first "estimating problem" Q^1 as follows.

Program Q^1

$$\text{minimize } C^1(Y) = \sum_{(ij)\varepsilon A} c^1_{ij}(y_{ij})$$

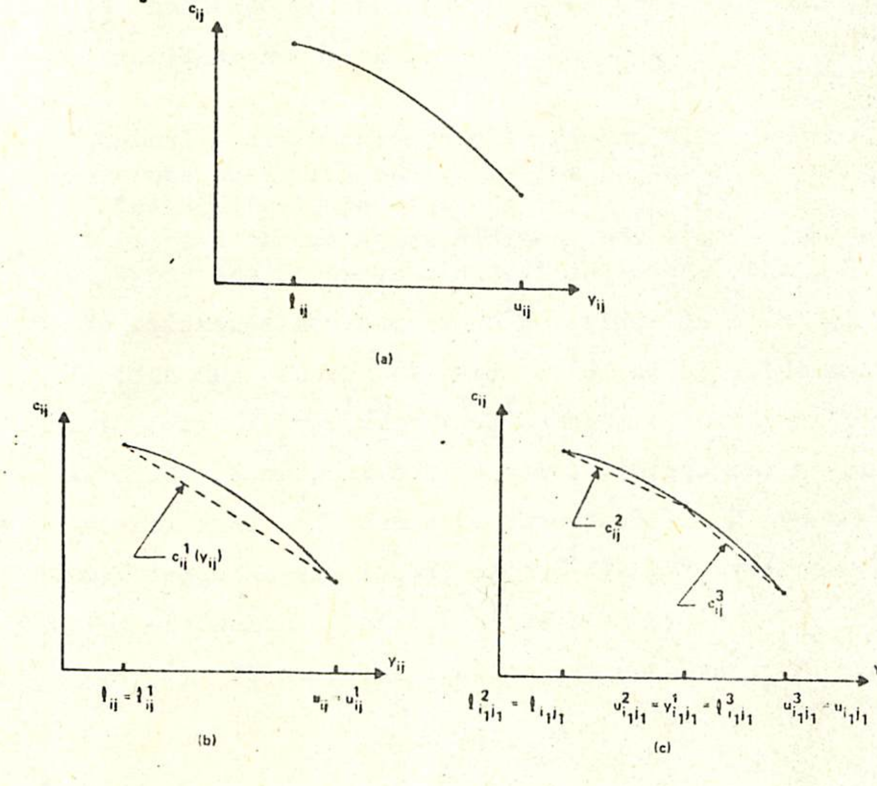


Figure 8.8 - Approximating a concave cost function.

$$\left.\begin{array}{ll} \text{s.t.} \quad t_i + y_{ij} \leq t_j & \text{for } (ij) \in A \\ \ell^1_{ij} \leq y_{ij} \leq u^1_{ij} & \text{for } (ij) \in A \\ t_1 = 0 \,,\ t_n = T & \end{array}\right\} F^1$$

Here, the upper and lower bounds on the duration y_{ij} are $\ell^1_{ij} = \ell_{ij}$ and $u^1_{ij} = u_{ij}$, as shown in Figure 8.8(b).

Interestingly enough, the Program Q^1 is a linear program which can be solved by the Fulkerson approach detailed in [8.24]. For the sake of simplicity of notation, denote the feasible space of the Program P by F, and denote the feasible space of the Program Q^1 by F^1. Clearly, $F^1 = F$ and consequently, if P is feasible, so is Q^1. Let Y^1 denote the optimal solution of Q^1; then $C^1(Y^1) = \Sigma c^1_{ij}(y^1_{ij})$ is a lower bound on the optimal value of the Program P, which we denote by C^*. Furthermore, since Y^1 is a feasible solution for P, $C(Y^1) = \Sigma c_{ij}(y^1_{ij})$ is an upper bound on C^*. Thus we have succeeded in bounding the optimal value C^* based on the optimal solution of Q^1 as follows.

$$C^1(Y^1) \leq C^* \leq C(Y^1) \tag{8.1}$$

Clearly, if equality holds throughout (8.1) then the current trial solution is optimal. This remark holds for all subsequent iterations, and hence will not be repeated. Now suppose that strict inequality holds above; then there is room for improvement. This is accomplished by producing a closer (albeit still an underestimating) linear approximation to the cost function $c(y)$. Consider the difference

$$c_{ij}(y^1_{ij}) - c^1_{ij}(y^1_{ij}) \tag{8.2}$$

which is always ≥ 0 and suppose that activity (i_1j_1) yields the maximum such difference, that is,

$$c_{i_1j_1}\left(y^1_{i_1j_1}\right) - c^1_{i_1j_1}\left(y^1_{i_1j_1}\right) = \max_{(ij)\varepsilon A} \left[c_{ij}\left(y^1_{ij}\right) - c^1_{ij}\left(y^1_{ij}\right)\right] > 0$$

Such an activity must exist, for otherwise the differences of (8.2) are zero for all activities, implying equality in (8.1) which is a contradiction. (In case of ties, any tied activity will do.) Divide the feasible domain of $y_{i_1j_1}$ into two subintervals: $\left[\ell_{i_1j_1}, y^1_{i_1j_1}\right]$ and $\left[y^1_{i_1j_1}, u_{i_1j_1}\right]$. (Recall that $y^1_{i_1j_1}$ is the value of the duration of activity (i_1j_1) obtained from the optimal solution of the Program Q^1.) Construct the two cost functions: $c^2_{i_1j_1}\left(y_{i_1j_1}\right)$ and $c^3_{i_1j_1}\left(y_{i_1j_1}\right)$ as the maximum linear approximations which underestimate the original cost function $c_{i_1j_1}\left(y_{i_1j_1}\right)$ in the two subintervals $\left[\ell_{i_1j_1}, y^1_{i_1j_1}\right]$ and $\left[y^1_{i_1j_1}, u_{i_1j_1}\right]$ illustrated as in Figure 8.8(c). Now define two linear programs as follows.

Program Q^2

minimize $\sum_{(ij)\neq(i_1j_1)} c^1_{ij}(y_{ij}) + c^2_{i_1j_1}\left(y_{i_1j_1}\right)$

$$\left.\begin{array}{ll} \text{s.t.} & t_i + y_{ij} \le t_j \quad \text{for all} \quad (ij) \in A \\ & \ell_{ij} \le y_{ij} \le u_{ij} \quad \text{for} \quad (ij) \ne (i_1j_1) \\ & \ell_{i_1j_1} = \ell^2_{i_1j_1} \le y_{i_1j_1} \le u^2_{i_1j_1} = y^1_{i_1j_1} \\ & t_1 = 0 \;,\; t_n = T \end{array}\right\} F^2$$

Program Q^3

$$\text{minimize} \sum_{(ij)\ne(i_1j_1)} c^1_{ij}(y_{ij}) + c^3_{i_1j_1}\left(y_{i_1j_1}\right)$$

$$\left.\begin{array}{ll} \text{s.t.} & t_i + y_{ij} \le t_j \quad \text{for all} \quad (ij) \in A \\ & \ell_{ij} \le y_{ij} \le u_{ij} \quad \text{for} \quad (ij) \ne (i_1j_1) \\ & y^1_{i_1j_1} = \ell^3_{i_1j_1} \le y_{i_1j_1} \le u^3_{i_1j_1} = u_{i_1j_1} \\ & t_1 = 0 \;,\; t_n = T \end{array}\right\} F^3$$

The logic of these two programs rests on the observation that the duration of the activity (i_1j_1) must lie in either of the two subintervals of Figure 8.8(c). Note that the only difference between these two programs is in the definition of the range of the duration $y_{i_1j_1}$.

Both programs are feasible since the point Y^1 is still a feasible point of either of them. Let F^2 denote the space of feasible solutions for the Program Q^2 and let F^3 denote the space of feasible solutions for the Program Q^3. Clearly, F^2 and F^3 are defined by

$$F^2 = F^1 \cap \left\{(Y,t): \ell^1_{i_1j_1} \le y_{i_1j_1} \le y^1_{i_1j_1}\right\}$$

$$F^3 = F^1 \cap \left\{(Y,t): y^1_{i_1j_1} \le y_{i_1j_1} \le u_{i_1j_1}\right\} \tag{8.3}$$

Furthermore,

$$c^2_{ij}(y_{ij}) = c^3_{ij}(y_{ij}) = c^1_{ij}(y_{ij}) \quad \text{for } (ij) \ne (i_1j_1) \tag{8.4}$$

Therefore, Problems Q^2 and Q^3 may be succinctly stated as:

Program Q^2: minimize $C^2(Y)$ s.t. $(Y,t) \in F^2$

Program Q^3: minimize $C^3(Y)$ s.t. $(Y,t) \in F^3$

Now both problems Q^2 and Q^3 may be solved by the Fulkerson algorithm, yielding optimal durations Y^2 and Y^3, respectively. By virtue of the fact that the cost functions C^2 and C^3 serve as tighter underestimates of C over their domains, and that the feasible space $F \equiv F^1 = F^2 \cup F^3$, we have

$$C^1(Y^1) \le \min\left\{C^2(Y^2), C^3(Y^3)\right\} \le C^* \le \min\left\{C(Y^1), C(Y^2), C(Y^3)\right\} \le C(Y^1)$$

The rightmost inequality follows from (Y^2,t^2) and (Y^3,t^3) being feasible solutions to Problem P. We have thus achieved improved bounds on the optimal value C^*.

From this point onwards, the algorithm proceeds in a series of stages. The zeroth stage, just detailed above, consists of Problem Q^1 and its solution Y^1. The first stage consists of Problems Q^2 and Q^3 and their solutions Y^2 and Y^3. The kth stage consists of Problems Q^{2k} and Q^{2k+1} and their solutions. The process is conveniently depicted by a typical binary-search tree whose nodes correspond to the Problems Q^s. Figure 8.9 depicts such a tree with four stages and nine nodes. Branching occurs when a particular activity is selected to have one of its (duration) intervals divided into two subintervals, as exemplified in Figure 8.8(c) with costs and bounds redefined as in (8.3) and (8.4). A branching node s may be selected according to some heuristic rule; for example, it may be done by choosing that Problem Q^s whose optimal value $C^s(Y^s)$ is minimal over all cost values associated

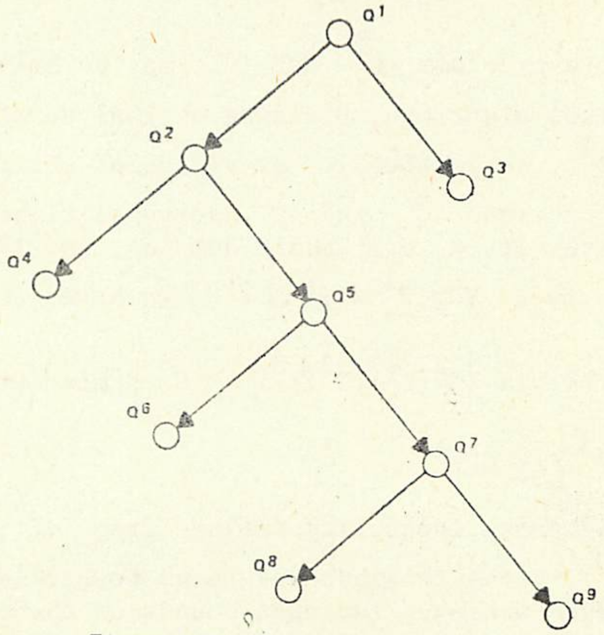


Figure 8.9 - A binary search tree.

with intermediate nodes (nodes from which no branching has occurred, denoted by $I(k)$ at stage k). The rationale here is that Problem Q^s has the smallest lower bound on C^* and, hopefully, the feasible space F^s will contain a point yielding a value to C approximately equal to $C^s(Y^s)$. Another possible heuristic rule is as follows: choose that Problem Q^s whose optimal solution Y^s yields the smallest interval of uncertainty on C^* . In any event, having chosen a Problem Q^s at stage k from which to branch, we create the Problems Q^{2k} and Q^{2k+1} in a manner exactly analogous to the manner in which Q^2 and Q^3 were created from Problem Q^1 . The feasible spaces F^{2k} and F^{2k+1} are thus defined by

$$F^{2k} = F^s \cap \left\{(Y,t): \ell_{i_s j_s} \leq y_{i_s j_s} \leq y^s_{i_s j_s}\right\}$$

$$F = F^s \cap \left\{(Y,t): y^s_{i_s j_s} \leq y_{i_s j_s} \leq u_{i_s j_s}\right\}$$

Furthermore, $c^{2k}_{i_s j_s}$ is the highest linear function underestimating $c_{i_s j_s}$ over the subinterval $\left[\ell_{i_s j_s}, y^s_{i_s j_s}\right]$ while $c^{2k+1}_{i_s j_s}$ is the highest linear function underestimating $c_{i_s j_s}$ over subinterval $\left[y^s_{i_s j_s}, u_{i_s j_s}\right]$. The sets F^{2k} and F^{2k+1} are feasible, since the point (Y^s, t^s) lies in both sets. The new problems Q^{2k} and Q^{2k+1} are linear problems with network constraints similar to those of Problem P and thus may be solved by the Fulkerson algorithm. The optimal value C^* is found at stage $k + 1$ by

$$v^{k+1} = \min_{s\in I(k+1)} \{C^s(Y^s)\} \leq C^* \leq \min_{s=1,2,\ldots,2k+1} \{C(Y^s)\}$$

$$= w^{k+1}$$

The process continues until either $v^k = w^k$ at some stage k, or when the interval bounding C^* is deemed small enough for practical purposes.

Three remarks should now be made. First, the choice of the subintervals on the duration $y_{i_s j_s}$ of Problem Q^s is arbitrary; the division indicated above seems to be a reasonable heuristic. Second, the above approach is clearly applicable to piecewise linear functions, whether concave or convex; in fact, the numerical example worked out in Falk and Horowitz [8.21] contained arcs in both categories. Third, the proof that the algorithm is finite rests on the fact that the function C is concave and defined over a convex polytope: it must assume its minimum at one of the finite vertices of the feasible space F. Most of the subproblems Q^s have their solutions at vertices of F. But since new vertices are created when new upper and lower bounds u^s_{ij} and ℓ^s_{ij} are added, some Q-problems may have solutions at points which are not vertices of F. The proof that only a finite number of such problems exist is given in Reference [8.22].

VII. Local Optima Provide Excellent Bounds. The dictum seems pedantic, yet its application, where possible, yields significantly improved results. To some degree this dictum, which advises the analyst to seek the optimum of the subspace under consideration, seems to be antithetical to Dictum V which advises against such optimization. Nevertheless, this dictum should be taken to read: if one can easily derive a local optimum, then it is worth implementing.

Example 8.9. One of the more recent illustrations of this dictum was provided by Mitten and Tsou [8.41] referred to below as M&T. They addressed themselves once more to the problem of scheduling N tasks on a single facility to minimize an objective cost expression. Their approach was to combine simple bounding with local optimality to achieve efficiency and fast convergence. We introduce their terminology and notation.

S	:	a finite set of N elements (the tasks)
$\sigma \subset s$	:	a subset of elements of S (a subset of the tasks)
P_σ	:	the set of all permutations of σ, with $P = \bigcup_\sigma P_\sigma$
$p \in P$	:	a permutation in P, $p = (w_1, w_2, \ldots, w_n)$, $n \leq N$, in which w_i is the task occupying the ith position in the sequence
S_p	:	the unordered set of elements contained in p, and $\bar{S}_p = S - S_p$
p^m	:	$(w_1, w_2, \ldots, w_m)$; the set of the first m elements of $p \in P$ with p^0 the null permutation
(p,q)	:	a permutation formed by the concatenation of the two dispoint permutations p and q, both in P
p_i	:	the ith element in the permutation $(w_1, w_2, \ldots, w_m)$

To each element $x \in S$ there are two given real and finite constants: $c_x > 0$, denoting the cost of task

x discounted to its start time, and $d_x \geq 0$ denoting its duration. For any permutation $p \varepsilon P$, let

$$D_p = \sum_{x \varepsilon S_p} d_x \quad \text{with} \quad D_{p^0} = 0$$

Assume that "now" is the start time of the scheduling process. Let $f(\xi) \geq 0$ be a real-valued bounded function, say a discount factor. Let $C(p,t)$ denote the cost associated with the permutation p when initiated at time t. For any real constant D_0, $0 \leq D_0 < \infty$, the cost associated with permutation $p \varepsilon P$ is given by

$$C(p,D_0) = \sum_{m=1}^{n} c_{p_m} f\left(D_0 + D_{p^{m-1}}\right)$$

Note that

$$C[(p,q),0] = C(p,0) + C(q,D_p)$$

The optimum is defined by

$$C^*(D_0) = \min_{p \varepsilon P_S} C(p,D_0)$$

which corresponds to some $p^* \varepsilon P^*(D_0)$.

Let us now turn to the generation of bounds. It is well known that a condition for local optimality of a permutation is that any contiguous binary switch (CBS) does not lead to improved objective value. The statement of this condition in M&T terminology is as follows. Let q and r be two disjoint permutations not containing the two distinct elements x and y. Let $p = (q,x,y,r)$ and $p' = (q,y,x,r)$; that is, p' is the permutation p with the elements x and y interchanged. Then

$$C(p,0) - C(p',0) = c_y[f(D_q+d_x) - f(D_q)] - c_y[f(D_q+d_y) - f(D_q)]$$

Let $R_w(D) = [f(D+d_w) - f(D)]/c_w$ for any $w \varepsilon S$ and $0 \leq D < \infty$. Then it can be seen that

$$R_x(D_q) \leq R_y(D_q)$$

is equivalent to

$$C(p,0) \leq C(p',0)$$

from which M&T obtain the following CBS condition

$$p \varepsilon P^* \text{ only if } R_x(D_q) \leq R_y(D_q) \qquad (8.5)$$

Furthermore, p is locally optimum if condition (8.5) holds for every adjacent pair of elements in p. This condition leads immediately to the following construction to establish an upper bound B on the optimal value. Construct a complete permutation $g = (g_1, g_2, \ldots, g_N)$ by iteratively choosing g_i satisfying

$$R_{g_1}(0) \leq R_x(0) \quad \text{for all } x \varepsilon S$$

$$R_{g_i}\left(D_{g^{i-1}}\right) \leq R_x\left(D_{g^{i-1}}\right) \quad \text{for all } x \varepsilon S - g^{i-1}.$$

Clearly, the value of any complete permutation is an upper bound; but this value $C(g,0)$ of the particular permutation g is usually a very good upper bound, if not optimal value. It is equally well-known that the cost of any partial permutation p is itself a lower bound b on the value of the optimum. However, if $f(\xi)$ is monotone in ξ, which is usually the case, we can do better. For, let σ be a nonempty proper subset of S which contains no elements of p.

Suppose that $f(\xi)$ is nonincreasing in ξ (the case of a discount factor). Consider the two partial permutations u and v defined as follows

$$u = (u_1, u_2, \ldots, u_n) \in P \text{ with } c_{u_1} \le c_{u_2} \le \cdots \le c_{u_n}$$

$$v = (v_1, v_2, \ldots, v_n) \in P \text{ with } d_{v_1} \ge d_{v_2} \ge \cdots \ge d_{v_n}$$

Then a sharper ℓ.b. is given by

$$b(D_p) = C(p,0) + \sum_{i=1}^{n} c_{u_i} \, f\left(D_p + D_{v^{i-1}}\right)$$

In case $f(\xi)$ is monotone nondecreasing, reverse the order of elements in both u and v.

Thus the u.b. is established on the basis of local optima. The ℓ.b. is based on a well-known inequality that presumes the relaxation of the parameters of the problem. The reader has just been introduced to a third form of relaxation!

8.5 Dominance and Feasibility

The ultimate in efficiency of search techniques is to be able to rule out, or eliminate from the set of contenders, all points but one based on logical arguments alone. Unfortunately this happy state of affairs rarely occurs, and when it does, it rules out the very need for an iterative search procedure such as discussed here under B&B. Still, the concept is appealing and eminently useful. We have already witnessed such elimination (fathoming) through the use of bounds. The two other avenues are: the establishment of dominance relations between subsets of the feasible space, and the establishment that certain completions of partial solutions are infeasible. This latter consideration arises as a consequence of the fact that, more often than not, the original space being searched is "enriched" through the inclusion of infeasible points. Thus the words "dominance and infeasibility" refer to the dual activity of weeding out infeasible points and, among the feasible ones, demonstrating that some subset is preferred to another.

The above use of dominance and infeasibility is essentially a process of elimination. Such elimination is motivated by one of two considerations: by inclusion, or by exclusion and decomposition and it is accomplished through the use of cuts, surrogate constraints, and relations. The following discussion elaborates on these notions.

VIII. Adaptive Cuts: Value Cuts and Configuration Cuts. By a "cut" is meant an additional constraint to the original statement of the problem. Such cuts are usually "adaptive" because their form and content depend on the stage of iteration and the particular partial solution being considered. The concept is based on the elementary observation that having arrived at a partial solution the analyst should, and usually can, augment the set of constraints on the basis of additional information available. These additional constraints "cut out" portions of the original feasible space. This results in sharper (lower or upper) bounds and reduced search effort. The concept is not new: it was first used by Little et al. [8.37] in their solution of the TSP. For instance, they asserted that given a partial-permutation $(1,i_2,i_3,\ldots,i_k)$ of the first k cities in a tour, then any completions which contain a city i_r in the subset of cities $1,i_2,\ldots,i_k$ are inadmissible; that is, such completions are "cut out" of the feasible space since the complete permutation would then contain a subtour. The generation of cuts in B&B procedures demands ingenuity and insight into the problem since the nature and form of cuts vary from problem to problem.

In general, there are two kinds of cuts: "value cuts" and "configuration cuts." The former type of cut is generated from knowledge of the value of the objective function, or knowledge of its bounds. The latter is gleaned from either the constraining set of equations or the set K of "other" restraints. Value cuts are illustrated by the following two examples.

Example 8.10. Consider the following general MLP problem which crops up regularly in the field of facility location-and-allocation problems:

Program L

$$\text{minimize} \quad \phi = fy + cx$$

$$\text{s.t.} \quad A_1 y + A_2 x = b$$

$$x \geq 0 \ , \ y_i = 0,1 \ \text{ and } \ y \in K$$

Here, f , c are given vectors, A_1 and A_2 are given matrices, and K represents additional constraints on the y_i . For a physical interpretation of this model, one may consider the binary variable y_i to correspond to the dichotomy: have a facility in location i $(y_i=1)$ or do not $(y_i=0)$. The fixed cost for establishing the facility is f_i independent of its size. The vector x may represent a level of the activities, with corresponding cost vector c . The matrices A_1 and A_2 are the coefficients of the variables in the constraining equations. The set K represents, for example, the so-called "bunching constraints" of the form

$$\sum_{i \in S_k} y_i = 1$$

which reflect the need for one, and only one, facility in a given subset of locations S_k . For ease of notation, denote by B the set of additional constraints imposed on the y and x vectors

$$B \triangleq \{y,x\colon x \geq 0 \ , \ y_i = 0,1 \ , \ y \in k\}$$

To illustrate the concept of value cuts we proceed to establish a sequence of lower and upper bounds on costs.

(a) A ℓ.b. on the total cost (LBTC). Solve the LP

$$\text{minimize} \quad \psi = fy + cx$$

$$\text{s.t.} \quad A_1y + A_2x = b$$

$$y \,,\, x \in B'$$

$$B' \triangleq \{y,x\colon x \geq 0 \;,\; 0 \leq y_i \leq 1 \;,$$

$$y \in K\}$$

In other words, the LBTC is obtained by solving the original problem but with the integer requirements on the y_i relaxed.

(b) A ℓ.b. on the cost associated with the continuous part (LBCC). Solve the MLP

$$\text{minimize} \quad \gamma = cx$$

$$\text{s.t.} \quad A_1y + A_2x = b$$

$$x,\, y \in B$$

Note that here we are demanding that y_i be a binary variable. In some cases the retention of this requirement poses no computational difficulty (see [8.17]); otherwise, substitute the set B' for B.

(c) An u.b. on the cost associated with the integer part (UBIC)

If at any stage of the search tree there is available an estimate of the upper bound on the total cost (UBTC), such as a complete feasible solution, it can be utilized to calculate an UBIC as follows. Define TIC (TCC) as the total integer cost (total continuous cost) associated with any solution vector y of the integer part. Clearly, we are interested in considering only the vectors y which could improve the current UBTC, that is, those which yield

$TIC \leq UBTC - TCC$

Since $TCC \geq LBCC$, then, a fortiori,

$TIC \leq UBTC - LBCC$

and the quantity UBTC-LBCC is an u.b. on the value of the integer costs (UBIC). We have generated a value cut of the form

$fy \leq UBIC\ (=UBTC-LBCC)$

This cut is updated once a better UBTC and/or a better LBCC is obtained. The cut is incorporated, upon its generation, in the restraining set.
Example 8.11. The second form of value cuts is exemplified by the well-known Benders' cuts [8.5] which are based on a partitioning approach.. Because of their frequent use in MLP problems, we briefly review their construction in the context of the above facility location MLP. Consider the mixed Program L stated at the beginning of Example 8.10 and let y^0 be any non-negative integer vector. The resulting problem is an ordinary (continuous) LP.

Program LC

$$\text{minimize} \quad \phi(y^0) = fy^0 + cx$$
$$\text{s.t.} \quad A_2x = b - A_1y^0$$
$$x \geq 0$$

The dual is

Program DLC

$$\text{maximize} \quad \psi\,(y^0) = fy^0 + u(b-A_1y)$$
$$\text{s.t.} \quad uA_2 \leq c$$

$u \geq 0$

If DLC has no feasible solution, then the vector y^0 is inadmissible. On the other hand, if DLC has an unbounded solution, then the primal LC is infeasible for all x, and again y^0 is declared inadmissible. This leaves y^0 admissible only if DLC possesses a finite maximum. Assume that to be the case, and suppose that it corresponds to $u^0(y^0)$. Clearly

$$\phi(y^0) = \psi^*(y^0) \geq \phi^*$$

and we may write

$$g^* = \min fy + \max u(b-A_2y)$$

where "min" is s.t. $y = 0,1$, and admissible, and "max" is s.t. $uA_2 \leq c$, $u \geq 0$. If T denotes the set of extreme points of the constraint set of (DLC) then T is of finite cardinality and

$$\psi^*(y^0) = fy^0 + \max_{u \in T} u(b-A_1y^0).$$

In order to restrict attention only to admissible y, we must guarantee that the solution to DLC is bounded. Suppose that it is not. Then it must be true that $u(b-A_1y^0)$ increases along some extreme ray. In other words, there would exist an extreme point u' and a direction v such that every point on the extreme ray

$$u' + \theta v, \quad \theta \geq 0$$

is feasible for DLC and $(u'+\theta v)(b-A_1y^0)$ increases with θ, or $\theta v(b-A_1y^0)$ increases with θ, implying that $v(b-A_1y^0) > 0$. To prevent this eventuality, let

$R = \{v: u + \theta v, \theta \geq 0$ is an extreme ray for some

$u \in T\}$

Then we require y^0 to satisfy

$v(b-A_1y^0) \leq 0$ for every $v \in R$

Incorporating this into the original MLP we obtain

$g^* = \min fy + \max u(b-A_1y)$

where "min" is s.t. $y = 0,1$, and "max" is s.t. $u \in T$ and $v(b-A_1y) \leq 0$ for every $v \in R$. The original MLP may be transformed into an equivalent MLP by introducing the objective function of DLC as a variable ψ such that

$\psi \geq fy^0 + u(b-A_1y^0)$ for $u \in T$.

Then the equivalent MLP is given by

Program EL

$$\phi^* = \text{minimum } \psi$$

$$\text{s.t. } \psi \geq fy + u(b-A_1y) \quad \text{for } u \in T$$

$$0 \geq v(b-A_1y) \quad \text{for } v \in R$$

$$y = 0,1$$

This "mixed" program has all integer variables but one, namely ψ . It is usually approached as an ILP with the understanding that any algorithm for ILPs would have to be modified slightly to solve it. Essentially, the MLP of L has been partitioned into the continuous linear program LC and the "almost integer" linear program EL. Theoretically, EL can be solved only if all the extreme points and extreme rays of DLC are known. Practically, EL can be solved iteratively by generating constraints only when they are needed.

Let $\bar{\phi}$ (possibly $+\infty$) be an upper bound on ϕ^*, obtained perhaps by a feasible solution. Let T^k and R^k be any subsets of extreme points and directions of extreme rays of $\{u: uA \leq c, u \geq 0\}$ known at iteration k (at the start, both T^0 and R^0 are empty). Since $\phi^* \leq \bar{\phi}$ and

$$\phi^* \leq fy + u(b-A_1y) \quad \text{for every} \quad u \in T^k$$

then we may impose the constraint:

$$\bar{\phi} \geq fy + u(b-A_1y) \quad \text{for every} \quad u \in T^k$$

which replaces the first set of constraints in EL. To restrict the enumeration to admissible y, we further have

$$v(b-A_1y) \leq 0 \quad \text{for every} \quad v \in R^k$$

Equivalently, we have

$$(-f+uA_1)\,y \geq -\bar{\phi} + ub \quad \text{for every} \quad u \in T^k$$

$$vA_1y \geq vb \quad \text{for every} \quad v \in R^k$$

$$y = 0,1$$

The sets T^k and R^k increase in size (implying additional constraints) as iterations proceed, which is one possible drawback of the Benders' partitioning algorithm. Of course, the number of iterations is limited by 2^n, the number of possible binary values of the vector y. However, empirical evidence shows that termination is achieved long before the sets

T^k and R^k contain all feasible extreme points and extreme rays.

Configuration cuts are usually implicitly embedded in the definition of the problem, residing either in the constraints set or in the definition of the set K. However, if a cut in the solution space of the integer is sought, these implied restraints must be made explicit. For instance, in capacitated plant location problems, the set of constraints may specify that all demands at the various destinations must be met. Thus the following constraint is implied

$$yq \geq RTDMN$$

where q is the vector of capacities at the various locations, and RTDMN is the residual total demand. The efficiency of this cut is data-dependent, but it has proved to be most powerful in some applications [8.17]. For another instance, the set K implies mainly the "multiple choice" constraints of the form

$$\sum_{i \varepsilon S_k} y_i \leq 1 \quad \text{for all } S_k, \ k = 1,2,\ldots$$

where S_k is a set of possible choices for a facility. These "bunching constraints" are adaptively updated as the search proceeds and partial solutions are obtained.

IX. Surrogate Constraints. Surrogate constraints do not augment the set of restraining equations, but rather substitute for them. The surrogate usually increases the search space. Hopefully, the solution of the problem with the surrogate is much easier than the original problem, which more than compensates for the loss in efficiency due to the expansion of the search space. Of course we are interested in the "tightest" surrogate, in the sense of minimal enlargement of the search space. If that is achieved, the resultant would be a tighter bound, which is always desirable.

Example 8.12. To fix ideas, consider the following ILP where $c \leq 0$ and $b \geq 0$.

Program R

$$\text{maximize} \quad z = cx$$

$$\text{s.t.} \quad Ax \leq b$$

$$x = 0,1$$

Suppose that at the kth stage of iteration we have a partial solution: some of the variables fixed at 0 form the set S_k^0 ; others fixed at 1 form the set S_k^1 ; and the remaining variables are still free and form the set F_k . At that particular node in the search tree, the problem may be stated as follows.

Program R_k

$$\text{maximize} \quad z_k = \sum_{j \varepsilon F_k} c_j x_j + \sum_{j \varepsilon S_k^1} c_j$$

$$\text{s.t.} \quad \sum_{j \varepsilon F_k} a_{ij} x_j \leq b_i - \sum_{j \varepsilon S_k^1} a_{ij} = s_i , \qquad (8.6)$$

$$i = 1,\ldots,m$$

$$x_j = 0,1 , \; j \; \varepsilon \; F_k$$

Suppose we substitute for the inequality constraints (8.6) the single constraint

$$\sum_{i=1}^{m} u_i \sum_{j \varepsilon F_k} a_{ij} x_j \leq \sum_{i=1}^{m} u_i s_i \qquad (8.7)$$

where $u = (u_1, u_2, \ldots, u_m) \geq 0$. Clearly, (8.7) is weaker than (8.6) since any set of x_j's satisfying the latter also satisfy the former but the converse need not be true, that is,

$S_k(u) \supseteq S_k$

where S_k and $S_k(u)$ are the set of binary feasible solutions to (8.6) and (8.7), respectively. Clearly, if $S_k(u) = \phi$, then $S_k = \phi$ also. To achieve the best surrogate constraint, we seek the multipliers u^* which yield

$$\min_{u \geq 0} \max_{x_j \varepsilon S_k(u)} \sum c_j x_j$$

$$x_j = 0,1$$

Let $g(u) = \max \sum_{j \varepsilon F_k} c_j x_j \; ; \; x \; \varepsilon \; S_k(u)$ and let $g'(u)$ denote the maximum with the integrality restrictions on the x_j's dropped:

$$g'(u) = \text{maximum} \sum_{j \varepsilon F_k} c_j x_j$$

$$\text{s.t.} \sum_{i=1}^{m} u_i \sum_{j \varepsilon F_k} a_{ij} x_j \leq \sum_{i=1}^{m} u_i s_i$$

$$0 \leq x_j \leq 1 \; , \; j \; \varepsilon \; F_k$$

Then the best multipliers to this continuous version of Q_k provide an excellent estimate of the multipliers u^*. If we denote the best multipliers by u^0, then

$$g'(u^0) = \min_{u \geq 0} g'(u)$$

It is easy to demonstrate that u^0 is given by the optimal (dual) variables to the dual LP to Program Q_k

with the integrality requirements removed, that is, the optimal dual variables to the LP

$$\text{minimize} \quad \sum_{i=1}^{m} u_i s_i + \sum_{j \in F_k} w_j$$

$$\text{s.t.} \quad \sum_{i=1}^{m} a_{ij} u_i + w_j \geq c_j \quad \text{for} \quad j \in F_k$$

$$u_i \, , w_j \geq 0 \quad \text{for all} \quad i \, , j$$

Once the (best) surrogate constraint is generated, it can be used to generate bounds and to fathom nodes as explained above.

X. Introduce Relations Where None Existed. While constraints and relations complicate mathematical programming problems, in the sense of increasing their complexity and time to solution, constraints and relations may in fact be of great assistance in B&B methods because they limit the space of search. After all, considerations of infeasibility are among the most powerful methods in the B&B approach.

Consequently, one always seeks to establish constraints and relations among the unknown variables of the problem in order to reduce the space of search. Cuts and surrogate constraints, discussed above, are such devices. Here we wish to highlight the concept of generation of cuts that are based on derived relations from logical arguments, where none existed before. The reader may wish to view the (dominance) relations emphasized here as added examples of cuts, which indeed they are. However, there is a generic difference between the cuts discussed here and, say, those based on Benders' decomposition: these latter are based on mathematical programming arguments, while the former are based on permutational arguments without recourse to the mathematical programming model. Two applications illustrate what is meant here. The first is due to Emmons [8.18] and the second is due to Elmaghraby and Park [8.16]. We briefly review the latter.

<u>Example 8.13</u>. In their study of scheduling N jobs on M machines in parallel, Elmaghraby and Park [8.16] developed a set of dominance relations that helps reduce the search effort drastically. Their approach exemplifies the concept of dominance by inclusion, namely, a schedule with a particular property dominates all others that do not possess such property. (Below under Dictum XI we discuss an approach that exemplifies dominance by exclusion.) Suppose job j has processing time p_j and due date equal to its processing time. Furthermore, suppose there is a linear penalty of tardiness given by the product of π_j and max $(0; T_j-p_j)$ for job j. It is desired to determine the schedule that minimizes the total penalty for tardiness. We use the following notation.

$Q_m = (m_1, m_2, \ldots, m_k)$ denotes a sequence of k jobs on machine m, $m = 1,2,\ldots,M$

Subscript m_j refers to the jth job on machine m. The subscript m_ℓ refers to the <u>last</u> job on machine m

s_i: the start time of job i. The start time of the last job on machine m is denoted by s_{m_ℓ}

T_i: the completion time of job i in a given sequence. The completion time of the last job on machine m is denoted by T_{m_ℓ}

$r_i = \pi_i/p_i$ and we assume that the jobs are numbered in nonincreasing ratio r_i, the so-called natural order

The dominance relations are presented in terms of properties of the optimal schedules or as precedence relations between pairs of jobs. (These latter assertions can be easily translated into dominance relations.)

(i) For a schedule Q to be optimal, it is necessary for $Q_m = (m_1, m_2, \ldots, m_k)$ to be sequenced in the natural order.

(ii) There exists an optimal schedule in which job 1 is scheduled first.

(iii) For a schedule Q to be optimal it is necessary that

$T_{i_\ell} \geq s_{j_\ell}$ for every pair of machines i and j

(iv) Using the convention, job i precedes job j means that $s_i \leq s_j$, we have that if

$p_i \leq p_j$ and $\pi_i \geq \pi_j$

then job $\underline{i}$ precedes job $\underline{j}$ in an optimal schedule.

(v) If $\pi_j = \pi$, a constant for all jobs, then there exists an optimal schedule in which jobs are assigned in their natural order from machine 1 to machine M in rotation.

(vi) Let $H = \sum_{i=1}^{N} p_i/M$, representing the processing interval if job splitting (including parallel operation on two or more machines) were permitted. Then in an optimal schedule on two machines, job $\underline{h}$ precedes job $\underline{k}$ if the following two conditions are satisfied.

(a) $r_h > r_{h+1}$ and $r_{k-1} > r_k$

(b) $$\sum_{1}^{h-1} p_i \leq H - \frac{1}{2} \max_{i \epsilon N} p_i - \sum_{k}^{n} p_i$$

As an example of translating the above statement into a dominance statement, consider (iii) above. Suppose we have a partial schedule of k jobs, denoted by $P(k)$, on three machines as shown by the solid lines in Figure 8.10. Then the completion of $P(k)$ which has job $(k+1)$ on machine 2 (the broken segment) dominates all other completions. In fact, if p_{k+1} (the processing time of job $k+1$) is such that $T_{2_\ell} + p_{k+1} < \min\left(T_{1_\ell}, T'_{3_\ell}\right)$, which is the case represented in the figure, then the completion of the partial schedule $P(k+1)$ itself which has job $k+2$ placed on machine 2 dominates all other completions!

XI. Exclusion and Decomposition. Exclusion and decomposition constitute another form of derived relations based on permutational arguments. The only

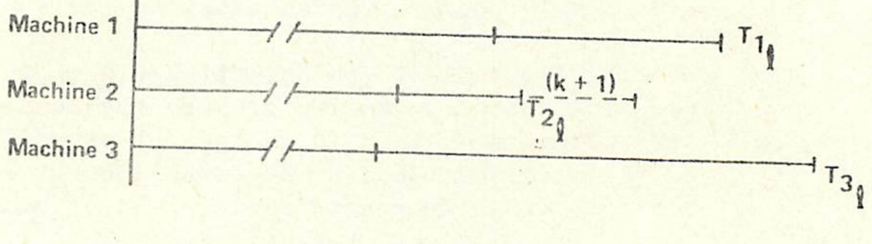


Figure 8.10 - A Gantt chart.

difference between exclusion cuts and the cuts discussed in Dictum X above is that here dominance is based on excluding a possibility. This is a weaker condition than the inclusion arguments discussed in X. On the other hand, decomposition arguments essentially "break up" the original space into subspaces with the links between them established optimally. Clearly, such decomposition reduces the burden of enumeration considerably. For instance, in a ten-job scheduling problem on a single facility the original space has $10! = 3,628,800$ points to be enumerated. If it can be established that optimality requires that a subset of four jobs must precede a subset of six jobs, then the enumeration is reduced to only $4! \times 6! = 17,280$ alternatives!

Example 8.14. An excellent illustration of such exclusion arguments and decomposition procedures was provided by Mitten and Tsou [8.41], referred to below as M&T, and whose paper was previously discussed in Example 8.9. They utilized the CBS condition (8.5) to establish some exclusion conditions as well as a condition for decomposition. Let $p = (p_1, p_2, \ldots, p_n)$ be a partial permutation of n elements $(n<N)$. Let $z \in \bar{S}_p$, where $\bar{S}_p = S - p$. The issue is to establish the conditions under which z is not a candidate for the (n+1)st position after p. Let σ_z be the set of elements of $\bar{S}_p$ whose R-value is not less than R_z at D_p:

$$\sigma_z = \{w \in \bar{S}_p : R_z(D_p) \leq R_w(D_p)\}$$

Clearly, $z \in \sigma_z$ and $\bar{S}_p - \sigma_z$ is the set of elements of $\bar{S}_p$ whose R-value is less than R_z at D_p. Let

$A_z = \sum_{w\in\sigma_z} d_w - \min_{w\in\sigma_z} d_w$. Then A_z is a ℓ.b. on the duration of σ_z beyond D_p and the following can be established. If

(a) $\sigma_z \neq \bar{S}_p$, that is, if there exist elements $u \in \bar{S}_p$ for which $R_z(D_p) > R_u(D_p)$, and if

(b) $R_x(D) < R_y(D)$ for all $x \in \bar{S}_p - \sigma_z$, $y \in \sigma_z$, and all D s.t. $D_p \leq D \leq D_p + A_z$ (that is, the elements of $\bar{S}_p$ whose R-value is $< R_z$ retain that inequality for all time periods in the interval $[D_p, D_p + A_z])$

then $(p,z,q) \notin P^*(0)$ for any permutation q of the remaining elements of the set $\bar{S}_p - \{z\}$. In particular, if $\sigma_z = \{z\}$ then z cannot follow p in position $n+1$. The immediate consequence of this result is the formation of a preference table that gives for each value of D (there are only finitely many) the elements ranked in order of increasing $R_x(D)$. A check on conditions (a) and (b) is then facilitated, and would normally result in the elimination of candidates (branches in the B&B search) for position $n+1$. To achieve decomposition, let $p \in P$ be a partial permutation and σ a nonempty subset of $\bar{S}_p$ to be partitioned into two nonempty subsets σ' and σ'' . As before, let $A = \sum_{w \in \sigma} d_w - \min_{w \in \sigma} d_w$, and $D' = \sum_{w \in \sigma} d_w$, Finally, suppose there exist two partial permutations $q^* \in P_{\sigma'}$ and $r^* \in P_{\sigma''}$ for which the following hold.

$C\left(q^*, D_p\right) \leq C(q, D_p)$ for all $q \in P_{\sigma'}$

$$C\left(r^*, D_p + D'\right) \leq C\left(r, D_p + D'\right) \quad \text{for all} \quad r \in P_{\sigma''},$$

$$R_x(0) < R_y(0) \quad \text{for all} \quad x \in \sigma', \; y \in \sigma'' \quad \text{and all}$$

$$D \in [D_p, D_p + A_\sigma]$$

Then we conclude that $C[(p,q^*,r^*),0] \leq C[(p,u),0]$ for all $u \in P_{\bar{S}}$. In other words, the completion of p has been decomposed into two parts, q^* and r^*, with q^* definitely preceding r^*. Naturally, the check for such decomposition is guided by the (sufficient) condition of its realization. In particular for any given $\sigma \in \bar{S}_p$ one determines the two subsets σ' and σ'' based on the R-values; then one proceeds to seek the subsequences q^* and r^*. M&T report excellent computing results using the approaches outlined above.

8.6 Miscellaneous Dicta

In this final section we present three dicta.

XII. It Is Sometimes Better To Use a Not-So-Economical Model Because It Is More Amenable To Computing "Tricks". Geoffrion and Graves' treatment [8.28] of a multicommodity distribution system provides an illustration.

Example 8.15. The problem statement from [8.28] is as follows. There are several commodities produced at several plants with known production capacities. There is a known demand for each commodity at each of a number of customer zones. This demand is satisfied by shipping via regional distribution centers (DCs), with each customer zone being assigned exclusively to a single DC. There are lower as well as upper bounds on allowable total annual throughput of each DC. The possible locations for the DCs are given, but the particular sites to be used are to be selected so as to result in the least total distribution cost. The DC costs are expressed as fixed charges (imposed only on the sites actually used) plus a linear variable charge which is a function of the "throughput" at the DC. Transportation costs are taken to be linear.

The mathematical model adopted was the following MLP.

minimize $x \geq 0$ and $y,z = 0,1$

$$\sum_{i,j,k,\ell} c_{ijk\ell}\, x_{ijk\ell} + \sum_{k} \Big[f_k z_k + v_k \sum_{i,\ell} D_{i\ell}\, y_{k\ell} \Big] \qquad (8.8)$$

s.t.

$$\sum_{k,\ell} x_{ijk\ell} \leq S_{ij} \quad \text{all } i, j \qquad (8.9)$$

$$\sum_{j} x_{ijk\ell} = D_{i\ell}\, y_{k\ell} \quad \text{all } i, k, \ell \qquad (8.10)$$

$$\sum_{k} y_{k\ell} = 1 \quad \text{all } \ell \qquad (8.11)$$

$$\underline{V}_k z_k \leq \sum_{i\ell} D_{i\ell}\, y_{k\ell} \leq \bar{V}_k z_k \quad \text{all } k \qquad (8.12)$$

Linear configuration constraints on one or both of y and z (8.13)

The notation is explained as follows.

$x_{ijk\ell}$	:	the amount of commodity i shipped from plant j through DC k to customer zone ℓ
$c_{ijk\ell}$	:	average unit cost of production and shipping corresponding to $x_{ijk\ell}$
z_k	:	a 0,1 variable, equal to 1 if a DC is acquired at site k, and 0 otherwise
f_k	:	the fixed cost associated with acquiring the facility at site k

$y_{k\ell}$	:	a 0,1 variable, equal to 1 if DC k serves customer zone ℓ, and 0 otherwise
$D_{i\ell}$	:	demand for commodity i in zone ℓ
v_k	:	variable unit cost of throughput in a DC at site k
S_{ij}	:	production (supply) capacity for commodity i at plant j
$\underline{V}_k, \bar{V}_k$	:	minimum and maximum allowed total annual throughput for a DC at site k

It is not difficult to see that this model is too liberal in the number of equations, a condition usually avoided by workers in LP (continuous, integer or mixed). There is an obvious opportunity to economize on the number of constraints of type (8.10) without changing the logical content of the model in any way: replace (8.10) by

$$\sum_{j,k} x_{ijk\ell} = D_{i\ell} \quad \text{all } i, \ell \tag{8.10a}$$

$$\sum_{i,j} x_{ijk\ell} = \left(\sum_{i} D_{i\ell}\right) y_{k\ell} \quad \text{all } k, \ell \tag{8.10b}$$

This formulation handles the two functions of (8.10) separately, ensuring that all demands are met, and forcing the appropriate logical relationship between the x's and y's. But in any particular application it presents a saving in the number of constraints over (8.10). For instance, suppose $i = 100$, $j = 10$, $k = 20$, and $\ell = 1000$. Equations (8.10) would yield (100)(20)(1000) = 2,000,000 equations; while (8.10a) and (8.10b) would yield only (100)(1000) + (20)(1000) = 120,000 equations! However, the program of (8.8)-(8.13) is actually easier to solve than the program with (8.10a) and (8.10b) substituted for (8.10). The reason lies in the application of the Benders' decomposition algorithm which yields excellent "cuts" and easy-to-solve transportation-type subproblems in the first case, and extremely weak "cuts" in the second case!

XIII. Do Not Optimize the Relaxed Problem; Just Obtain a Bound on Its Solution. In spite of the rather elementary nature of the concept, its implementation may spell the difference between an operational scheme and a difficult, albeit correct, one. The dictum is best illustrated by Eastman [8.9] and that of Little et al. [8.37] in the calculation of the bounds for the TSP. Both treatments approached the problem from the point of view of relaxing the tour constraint, which leaves the standard assignment problem. While Eastman solved the assignment problem to obtain its optimum, Little et al. were content with a ℓ.b. on its value. Clearly

$z^* \geq$ opt. value of assignment problem $\geq$ ℓ.b. on assignment problem

XIV. Preconditioning: Limit the Size of the Original Space To Be Searched. This is a manifestation of the well-known motto: "an ounce of prevention is worth a pound of cure." It draws attention to the need for the proper "conditioning" of the search space before the search is initiated. Two examples are provided in the literature. The first is due to Held, Karp and Sherashian [8.32] in the study of assembly line balancing, and the second is due to Elmaghraby [8.11] in a scheduling problem. We briefly discuss this latter.

Consider the scheduling of N jobs on a single facility, in which each job j has a processing time p_j, due date d_j, and a penalty function $\pi_j \max(0;T_j-d_j)$ which penalizes tardiness, where T_j is the completion time of job j deduced from the schedule. It is desired to determine the schedule that minimizes the total cost of tardiness. At the outset there are $N!$ different sequences, and as many points to be enumerated. However, through developing a dynamic programming model of the problem, the size of the search space was reduced to only 2^N. Branch-and-Bound was then applied to this smaller space, with excellent results.

References

[8.1] Agin, N. (1966). Optimum seeking with branch and bound. Management Sci. 13 B176-B185.

[8.2] Ashour, S. (1972). Sequencing Theory. Lecture Notes in Economics and Mathematical Systems 69. Springer.

[8.3] Ashour, S., and M. N. Quraishi (1969). Investigation of various bounding procedures for production scheduling problems. International Journal of Production Research 7 249-252.

[8.4] Beale, E. M. L. (1965). A survey of integer programming. Operational Res. Quart. 16 219-229.

[8.5] Benders, J. F. (1962). Partitioning procedures for solving mixed-variable programming problems. Numer. Math. 4 238-252.

[8.6] Breu, R., and C. A. Burdet (1973). Branch and bound experiments in 0-1 programming. Management Science Research Group Report, Carnegie-Mellon University (August).

[8.7] Clark, C. E. (1961). The optimum allocation of resources among the activities of a network. Journal of Industrial Engineering 12 11-17.

[8.8] Davis, E. W. (1973). Project scheduling under resource constraints--historical review and categorization of procedures. American Institute of Industrial Engineering Transactions 5 257-313.

[8.9] Eastman, W. L. (1968). A solution to the traveling salesman problem. Paper presented at the Summer Meeting of the Econometric Society, Cambridge, Massachusetts (August).

[8.10] Eastman, W. L., S. Even, and J. M. Isaacs

(1964). Bounds for optimal scheduling of jobs. Management Sci. 11 268-279.

[8.11] Elmaghraby, S. E. (1968a). The one machine sequencing problem with delay costs. Journal of Industrial Engineering 19 105-108.

[8.12] Elmaghraby, S. E. (1968b). The determination of optimal activity duration in project scheduling. Journal of Industrial Engineering 19 48-51.

[8.13] Elmaghraby, S. E. (1968c). The sequencing of n jobs on m parallel processors with extensions to scarce resource problem of activity networks. in Proceedings of the Inaugural Conference of the Scientific Computation Center and the Institute of Statistical Studies and Research, Cairo, Egypt, 230-255.

[8.14] Elmaghraby, S. E., and A. K. Mallik (1973). The scheduling of a multi-product facility. in S. E. Elmaghraby (ed.) Symposium on the Theory of Scheduling and its Applications. Lecture Notes in Economics and Mathematical Systems 86. Springer. 244-277.

[8.15] Elmaghraby, S. E., and L. P. Dix (1973). Scheduling a single facility under constant demand and fixed production rate. OR Report No. 87, North Carolina State University (May).

[8.16] Elmaghraby, S. E., and S. Park (1974). Scheduling jobs on a number of identical machines. American Institute of Industrial Engineering Transactions 6 1-13.

[8.17] Elshafei, A. N. (1974). On solving the capacitated facility location problem with concave cost functions. OR Reports Nos. 92, 93, North Carolina State University (June).

[8.18] Emmons, H. (1969). One machine sequencing to minimize certain functions of job tardiness. Operations Res. 17 701-715.

[8.19] Faaland, B. H. (1972). Estimates and bounds on computational effort in the accelerated bound-and-scan algorithm. Technical Report No. 35, Department of Operations Research, Stanford University (May).

[8.20] Faaland, B. H., and F. S. Hillier (1972). The accelerated bound-and-scan algorithm for integer programming. Technical Report No. 34, Department of Operations Research, Stanford University (May).

[8.21] Falk, J. E., and J. L. Horowitz (1972). Critical path problems with concave cost-time curves. Management Sci. 19 446-455.

[8.22] Falk, J. E., and R. M. Soland (1969). An algorithm for non-separable non-convex programming problems. Management Sci. 15 550-569.

[8.23] Forrest, J. J. H., J. P. H. Hirst, and J. A. Tomlin (1974). Practical solution of large mixed integer programming problems with UMPIRE. Management Sci. 20 736-773.

[8.24] Fulkerson, D. R. (1961). A network flow computation for project cost curve. Management Sci. 7 167-178.

[8.25] Garfinkel, R. S., and G. L. Nemhauser (1972). Integer Programming. Wiley.

[8.26] Geoffrion, A. M. (1967). Integer programming by implicit enumeration and Balas method. SIAM Rev. 9 178-190.

[8.27] Geoffrion, A. M. (1970). Elements of large scale mathematical programming. Parts I and II. Management Sci. 16 652-691.

[8.28] Geoffrion, A. M., and G. W. Graves (1974). Multicommodity distribution system design by Benders decomposition. Management Sci. 20 822-844.

[8.29] Glover, F. (1965). A multi-phase dual algorithm for the zero-one integer programming problem. Operations Res. 13 879-919.

[8.30] Greenberg, H., and R. L. Hegerick (1970). A branch and bound algorithm for the knapsack problem. Management Sci. 16 327-332.

[8.31] Held, M., and R. M. Karp (1970). The traveling salesman problem and minimum spanning trees. Operations Res. 18 1138-1162.

[8.32] Held, M., R. M. Karp, and R. Sherashian (1963). Assembly line balance--dynamic programming with precedence constraints. Operations Res. 11 442-459.

[8.33] Ignall, E., and L. Schrage (1965). Application of the branch-and-bound technique to some flow shop scheduling problems. Operations Res. 13 400-412.

[8.34] Kan, A. H. G. Rinnooy (1973). The machine scheduling problem. The Mathematical Center, Amsterdam (August).

[8.35] Land, A. H., and A. G. Doig (1960). An automatic method for solving discrete programming problems. Econometrica 20 497-520.

[8.36] Lawler, E. L., and D. E. Wood (1966). B&B methods: a survey. Operations Res. 14 699-717.

[8.37] Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel (1963). An Algorithm for the traveling salesman problem. Operations Res. 11 972-989.

[8.38] Lomnicki, Z. A. (1965). A branch-and-bound algorithm for the exact solution of the three machine scheduling problem. Operational Res. Quart. 16 89-100.

[8.39] Mitten, L. G. (1970). Branch-and-bound methods: general formulation and properties. Operations Res. 18 24-34.

[8.40] Mitten, L. G. (1973). Branch and bound: a general formulation and survey of the literature. Research Memorandum, University of British Columbia (August).

[8.41] Mitten, L. G., and C. A. Tsou (1973). Efficient solution procedures for certain scheduling and sequencing problems. in S. E. Elmaghraby (ed.) Symposium on the Theory of Scheduling and Its Applications. Lecture Notes in Economics and Mathematical Systems 86. Springer 127-142.

[8.42] Mitten, L. G., and A. R. Warburton (1973). Implicit enumeration procedures. Research Memorandum, University of British Columbia (March).

[8.43] Rech, P., and L. G. Barton (1970). A non-convex transportation algorithm. in E. M. L. Beale (ed.) Applications of Mathematical Programming Techniques. American Elsevier. 250-260.

[8.44] Srinivasan, V., and G. L. Thompson (1973). Solving scheduling problems by applying cost operations to assignment models. in S. E. Elmaghraby (ed.) Symposium on the Theory of Scheduling and its Applications. Lecture Notes in Economics and Mathematical Systems 86. Springer 399-425.