Netherlands school of economics

ECONOMETRIC INSTITUTE

BRANCHING FROM THE LARGEST UPPER BOUND: FOLKLORE AND FACTS

B.L. FOX, J.K. LENSTRA, A.H.G. RINNOOY KAN
and L.E. SCHRAGE

REPORT 7722/0

BRANCHING FROM THE LARGEST UPPER BOUND: FOLKLORE AND FACTS

B.L. FOX

Université de Montréal, Canada

J.K. LENSTRA

Mathematisch Centrum, Amsterdam, The Netherlands

A.H.G. RINNOOY KAN

Erasmus University, Rotterdam, The Netherlands

L.E. SCHRAGE

University of Chicago, U.S.A.

ABSTRACT

Branch-and-bound algorithms are widely used to solve combinatorial maximiza-
tion problems. At each step of such an algorithm a search strategy selects
an active subset of feasible solutions for examination. In this paper we
discuss the formal properties and the practical value of search strategies
based on branching from the largest upper bound (BLUB strategies). We inves-
tigate conditions under which BLUB strategies are optimal in the sense that
they minimize the number of subsets generated. Counterexamples show that the
conditions given in the literature are not strong enough and a correct opti-
mality condition is formulated. Finally, we argue that the practical objec-
tions raised against BLUB strategies are not necessarily convincing.

1. INTRODUCTION

Let us consider branch-and-bound algorithms as applied to maximization prob-
lems. One of the components of such an algorithm is a search strategy that
at each step selects an active subset of feasible solutions for examination.
The search strategies based on *branching from the largest upper bound* (BLUB
strategies) are often credited with certain optimality properties. For
instance, Lawler and Wood [7,p.712] comment on BLUB strategies:

> Suppose that for any given problem the set of new bounding problems is
> uniquely determined. Then this policy has the advantage that the total
> amount of computation is minimized, in the sense that any branching
> operation performed under this policy must also be performed under any
> other policy.

Garfinkel and Nemhauser [5,p.149] give the following exercise:

> Assume that all aspects of a branch and bound algorithm have been
> determined, except for the branching rule. One is interested in devising
> a branching rule that minimizes the number of vertices considered. If
> the original problem is to find all optimal solutions to
>
> max z(x), x ∈ S
>
> show that the appropriate rule is "branch to a vertex having the largest
> upper bound."

In Section 2, counterexamples show that neither of these characterizations
is completely accurate and a correct optimality condition is formulated. In
Section 3, we consider the practical value of BLUB strategies and argue
against some traditional criticisms.

2. FORMAL PROPERTIES OF BLUB STRATEGIES

The essential components of a branch-and-bound algorithm for maximization
of a real objective function z are the following.

- A *branching rule* β ("separation rule" in [5]) associates a family β(S) of subsets to a subset S of feasible solutions such that $U_{S' \in \beta(S)}$ S' = S; the subsets S' are the *children* of the *parent* subset S.
- A *lower bounding rule* LB provides a lower bound LB(S) on the value of each solution in a subset S generated by β.
  - A *global lower bound* LB^* on the value of the optimal solution is provided by the largest LB(S) found so far.
  - Computation of LB(S) is assumed to include feasibility test (if any) and may lead to *improvement* of LB^*.
- An *upper bounding rule* UB provides an upper bound UB(S) on the value of each solution in S.
  - If S is a *singleton* {x}, then LB(S) = UB(S) = z(x).
  - *Elimination* of S occurs if UB(S) ≤ LB^*.
- A *search strategy* σ ("branching rule" in [5]) selects a currently active subset for examination, where:
  - *active* subsets are generated subsets which have neither been eliminated nor led to branching;
  - *examination* of S is understood to imply determination of β(S) and computation of LB(S') and UB(S') for all S' ∈ β(S).

The set of all feasible solutions is assumed finite. Two common types of search strategies are *jumptrack* strategies, whereby any active subset can be selected, and *backtrack* strategies, which, after examination of S' ∈ β(S), examine all children S" ∈ β(S') before returning to the other members of β(S). A jumptrack strategy is called a *BLUB strategy* if it prescribes the examination of an active subset with largest upper bound; a strategy is *optimal* if it minimizes the number of subsets generated to find an optimal solution (including verification of its optimality).

An actual branch-and-bound computation can be conveniently represented by means of a *search tree*. Initially, the tree consists of a single node corresponding to the set of all feasible solutions. The generation of a subset is represented by the creation of a node corresponding to S and an edge between this node and its parent. Each node may have two labels, corresponding respectively to the values of the lower bound and the upper bound.

In this section we will investigate the optimality of BLUB strategies. In the first place, the presence of *ties among upper bounds* can lead to
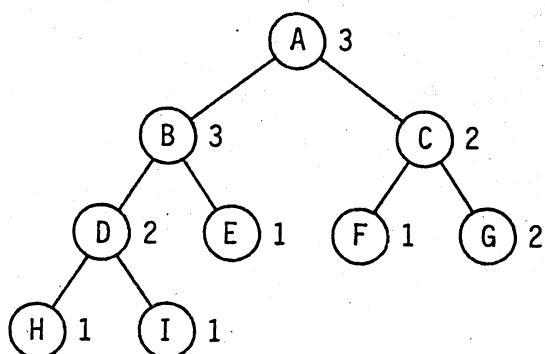
Figure 1 Search tree for the example.

situations in which *some* BLUB strategies are non-optimal, as demonstrated by the following eaxmple.

Example. Consider the search tree given in Figure 1. The nodes correspond to subsets denoted by capitals and are labelled only by the values of their upper bounds. Let us assume that the unique optimal solution with value 2 is contained in G and is found during computation of LB(G). Consider the following search strategies:

- σ_1 examines A,B,C in that order and generates B,C,D,E,F,G;
- σ_2 examines A,B,D,C in that order and generates B,C,D,E,H,I,F,G;
- σ_3 examines A,C,B in that order and generates B,C,F,G,D,E.

Obviously, σ_1 is an optimal BLUB strategy, σ_2 is a non-optimal BLUB strategy, and σ_3 is an optimal non-BLUB strategy. □

In the second place, there are situations in which *all* BLUB strategies may be non-optimal. At each stage of the tree search, let the *complete history* refer to, roughly speaking, all information obtained so far and the *direct history* to the information obtained along the path from the initial node to the node under examination. Up to now, we have tacitly assumed that branching and bounding rules only depend on the direct history. Let us quote two instances of actual branch-and-bound algorithms for which this assumption is false.

- Geoffrion's mixed integer programming algorithm [6] (see also [5, Section 4.7]) computes strong upper bounds via the appropriate linear programming relaxations at, say, every sixteenth node; at all other nodes only a derived surrogate constraint is used. In this case, UB depends on the complete history.

- The MPSX/MIP algorithm [2;8] allows the use of a variation of "pseudo-costs". For example, the attractiveness of branching on a certain variable may be measured by the average reduction in upper bound over all previous nodes in which branching on that variable took place. Thus, β is allowed to depend on the complete history.

It is easily checked that dependence of β, LB or UB on the complete history may destroy optimality with respect to all BLUB strategies.

Example (continued). Suppose that at the third node generated a weak upper bound is computed instead of a strong one. Suppose further that the weak and strong bounds are the same except at node D, where the weak bound is 3 and the strong bound is 2, as above. The unique BLUB strategy is the non-optimal strategy σ_2 and the unique optimal strategy is the non-BLUB strategy σ_3. $\square$

Similar examples involving β and LB are easily constructed. In the latter case, a non-optimal BLUB strategy can exist only if ties occur among upper bounds.

When *is* branching from the largest upper bound optimal? Consider the following conditions:
(1)   no ties occur among upper bounds;
(2) β, LB and UB do not depend on σ.
Condition (2) implies that any search strategy generates a subtree of a fixed, doubly-labelled, complete enumeration tree. It also implies that β, LB and UB depend only on the direct history, since there exists a search strategy that first generates the path from the initial node to the node under examination.

If (1) and (2) hold, then all BLUB strategies are optimal. If (1) is violated but (2) holds, then we have that:
- at least one BLUB strategy is optimal;
- if a BLUB strategy generates n nodes, e of which have upper bounds equal to the optimal solution value and d of which are children of such nodes but have smaller upper bounds, then any search strategy has to generate at least n-(d+e-1) nodes, examining at least n-(e-1) of them.

The latter observation implies that, under condition (2), any BLUB strategy

is not far from optimal, unless there are many tied bounds. The proofs of
the above results are simple and can be left to the reader. It is possible
to strengthen the bound on BLUB performance by assuming more information
about the unexamined portion of the search tree.

Condition (2) may seem overly restrictive. It does hold, however, for
most specialized branch-and-bound algorithms. In [3], it holds automatically
and there is no advantage in weakening it. In any case, (2) appears to be
an essential hypothesis for any statement about the theoretical merits of
BLUB strategies.

3.   PRACTICAL VALUE OF BLUB STRATEGIES

If condition (2) holds, the above propositions suggest the use of BLUB
strategies. Even if (2) does not hold, BLUB may be a good heuristic. The
case for BLUB seems especially strong if we terminate when the difference
between the current largest upper bound and LB^* is less than a given
tolerance. Yet no commercial code that we know of uses BLUB even periodi-
cally. It is true that MPSX/MIP [8,p.111] and FMPS [9,pp.8-18] have a BLUB
option, but their authors recommend against using it.

Traditional objections raised against BLUB strategies are that they
require:
(A)   excessive time to find an active node with largest upper bound;
(B)   excessive bookkeeping;
(C)   excessive storage.
Objection (A) seems hardly serious. By the use of existing list processing
algorithms, the family of currently active nodes can be kept ordered accord-
ing to nonincreasing upper bounds, and deletion and insertion of nodes can
be accomplished in time proportional to the logarithm of the family size
[4;1].

In contrast to backtrack strategies, jumptrack strategies such as BLUB
cannot be implemented in a straightforward recursive manner. This makes BLUB
strategies relatively complex to program and adds to the bookkeeping over-
head. We would grant that on small-scale or one-shot applications BLUB does
not appear worthwhile for this reason. Let us assume then that we are con-
templating the design of a large-scale production code.

Probably, the most involved bookkeeping arises when it would be very inefficient to compute bounds from scratch at each node. For example, suppose that bounds are computed via linear programming. For each generated node, we either store the corresponding basis inverse directly or provide sufficient information so that it can be easily reconstructed. Inverses for adjacent nodes generally differ only little, so that relatively few (strategically placed) inverses need to be stored explicitly.

In such situations, objection (B) may be valid. Still, it would require actual computational experiments to investigate if the trade-off between tree size and bookkeeping overhead works out to the disadvantage of BLUB strategies.

Similar remarks apply to objection (C). In general, jumptracking will require more storage than backtracking. This is a disadvantage only if both:

(a) a backtrack strategy permits everything to be done in core;

(b) accessing secondary storage takes a considerable amount of time (*i.e.*, buffering with negligible swap times is not feasible), or secondary storage space cannot effectively be regarded as infinite.

With respect to some large-scale applications, (a) does not hold and it seems that (b) has not been seriously investigated. Buffering may be facilitated and secondary storage requirements reduced if in a hybrid fashion we alternate between jumptracking and backtracking.

To summarize: we do not find the objections raised against BLUB strategies and their hybrid variants necessarily convincing and we believe that they merit serious empirical investigation.

ACKNOWLEDGEMENTS

REFERENCES

1. A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN (1974) *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.

2. M. BÉNICHOU, J.M. GAUTHIER, P. GIRODET, G. HENTGÈS, G. RIBIÈRE, O. VINCENT (1971) Experiments in mixed-integer linear programming. *Math. Programming* 1,76-94.

3. E.V. DENARDO, B.L. FOX (1977) Shortest route methods: 2. group knapsacks, expanded networks and branch-and-bound. To appear in *Operations Res.*

4. B.L. FOX (1970) Accelerating list processing in discrete programming. *J. Assoc. Comput. Mach.* 17,383-384.

5. R.S. GARFINKEL, G.L. NEMHAUSER (1972) *Integer Programming*. Wiley, New York.

6. A.M. GEOFFRION (1969) An improved implicit enumeration approach for integer programming. *Operations Res.* 17,437-454.

7. E.L. LAWLER, D.E. WOOD (1966) Branch-and-bound methods: a survey. *Operations Res.* 14,699-719.

8. (1971) MPSX/MIP program description. International Business Machines Corporation.

9. (1975) Sperry Univac 1100 series functional mathematical programming system (FMPS) programmer reference. Sperry Rand Corporation.

REPORTS 1977

7700	List of Reprints, nos. 179-194; List of Reports, 1976.
7701/M	"Triangular - Square - Pentagonal Numbers", by R.J. Stroeker.
7702/ES	"The Exact MSE-Efficiency of the General Ridge Estimator relative to OLS", by R. Teekens and P.M.C. de Boer.
7703/ES	"A Note on the Estimation of the Parameters of a Multiplicative Allocation Model", by R. Teekens and R. Jansen.
7704/ES	"On the Notion of Probability: A Survey", by E. de Leede and J. Koerts.
7705/ES	"A Mathematical Theory of Store Operation", by B. Nooteboom.
7706/ES	"An Analysis of Efficiency in Retailing, by B. Nooteboom.
7707/S	"A Note on Theil's Device for choosing a Blus Base", by C. Dubbelman
7708/E	"A General Market Model of Labour Income Distribution: An Outline", by W.H. Somermeyer.
7709/E	"Further Results on Efficient Estimation of Income Distribution Parameters", by T. Kloek and H.K. van Dijk.
7710	"List of Reprints, nos. 195-199; Abstracts of Reports First Half 1977".
7711/M	"Degenerating Families of Linear Dynamical Systems I", by M. Hazewinkel.
7712/M	"Twisted Lubin-Tate Formel Group Laws, Ramified Witt Vectors and (Ramified) Artin-Hasse Exponential Mappings", by M. Hazewinkel.
7713/EM	"An Efficient Way in Programming 'Eaves' Fixed Point Algorithm" by R. Jansen and A.S. Louter.
7714/E	"An Alternative Derivation and a Generalization of Nataf's Theorem", by W.H. Somermeyer and J. van Daal.
7715/M	"Application of Non-Linear Programming to Plane Geometry", by R. Stroeker.
7716/S	"Stochastic Compactness of Sample Extremes" by L. de Haan and G. Ridder.
7717/ES	"Cross-Section Studies of Efficiency in Retailing, Part I: Individual Grocers and Butchers", by B. Nooteboom.
7718/O	"On Redundancy in Systems of Linear Inequalities", by J. Telgen.
7719/O	"On R.W. Llewellyn's Rules to Identify Redundant Constraints; A Detailed Critique and Some Generalizations", by J. Telgen.
7720/O	"Redundant and Non-Binding Constraints in Linear Programming Problems", by J. Telgen.
7721/MO	"The Complexity of the Network Design Problem", by D.S. Johnson, J.K. Lenstra and A.H.G. Rinnooy Kan.
7722/O	"Branching from the Largest Upper Bound: Folklore and Facts", by B.L. Fox, J.K. Lenstra, A.H.G. Rinnooy Kand and L.E. Schrage.