RC22890 (W0309-010) September 3, 2003
Mathematics

IBM Research Report

Branch and Bound, Integer and Non-Integer Programming

J. J. H. Forrest
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

J. A. Tomlin
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Branch and Bound, Integer, and Non-Integer Programming

J.J.H. Forrest
IBM Watson Research Center, Yorktown Heights, NY 10598
J. A. Tomlin
IBM Almaden Research Center, San Jose, CA 95120
{forrest@watson.ibm.com,tomlin@almaden.ibm.com}

Abstract

In this note we review the development of the first commercial branch and bound codes at CEIR/Scicon, with particular emphasis on those features not commonly emphasized today—that is the use of branch and bound to handle discrete constructs, which, while often formally representable in terms of zero-one integer variables, can be handled more elegantly and effectively by applying branch and bound to the actual dichotomies in the model.

keywords: Branch and bound, Special Ordered Sets, Integer Programming

1 Introduction

The purpose of this note is threefold:

1. To give a brief survey of the early development of branch and bound codes at CEIR, which subsequently became Scicon, Ltd.

2. To celebrate the seminal contributions of our mentor E.M.L. (Martin) Beale to these developments.

3. To emphasize the importance and flexibility of the branch and bound framework for models and algorithms *not* requiring explicit integer variables.

The potential power of an ability to solve mixed integer linear (MIP) problems had been known since the mid 50's (see Markowitz and Manne (1957) and Dantzig (1963)), but remained unrealized, when Martin Beale, then visiting Princeton University in 1957/8, encountered R.E. Gomory and explained to him that solving integer programming problems was "impossible" (Gomory (1991)). As Gomory was to show, solution of integer programs in a finite number of iterations was possible, but as he was also heard to remark, finite numbers can be very large indeed. Thus, although Gomory's method of integer forms (Gomory (1963)), and later Glenn Martin's Accelerated Euclidean Algorithm (Martin (1963)), were able to solve some significant IP problems, they were unsatisfactory in general practice.

It was the seemingly pedestrian approach of Branch and Bound, based on the work of Land and Doig (1960), which proved successful in practice, and was implemented in the CEIR LP/90/94 system, then the most powerful of its day, by E.M.L. Beale and his colleagues (see Beale and Small (1965), Beale (1968)) . Experience with this code is also reported by Shaw (1969), and the present authors had some experience with the system before it was superseded by UMPIRE. Some familiarity with the concepts of linear programming and branch and bound will be assumed for the remainder of this paper.

There were some notable variations of the original branch and bound ideas in the CEIR LP/90/94 implementation. Firstly, the tree search was depth-first (or last in first out—LIFO), dictated by the need

to save postponed branches sequentially on tape. Second, Driebeck penalties (Driebeck (1966)) were used to guide the tree search, and thirdly, even at this early stage it was realized that multiple choice zero-one variables (then called "covering variables") could benefit from special treatment. The difficulty is that when there are, say, p zero-one variables constrained to sum to 1, and some are fractional, the branch which sets one of them to 0 is very "weak"—that is unlikely to bring us closer to an integer solution—while setting it to 1 forces the remaining $p-1$ variables to 0. The C-E-I-R LP/90/94 system (see Beale (1968)) attempted to finesse this by always postponing the branch associated with the weaker choice and pursuing the stronger one. The danger is that it may be too strong, as we discuss below.

2 Special Ordered Sets and UMPIRE

It is often thought that Special Ordered Sets (SOS) were primarily designed to handle multiple choice integer variables, but this is not the case. They were inspired by a model involving piecewise-linear non-convex (actually concave) costs, described in a paper modestly titled "Two Transportation Problems" (Beale (1963)). The hand guided procedure used (described in Beale (1963) and reviewed in Tomlin (1988)) laid the groundwork for the more general special ordered set concept. Thus what came to known as SOS of type 2 (or S2 sets) actually preceded what came to be known as S1 sets, when they were implemented for the first time in Scicon's UMPIRE system. It should also be noted the idea of "ordered sets" of variables was very appealing, because Generalized Upper Bounds (GUB) were important in the design of UMPIRE (Beale (1970)), and thus the variables were already divided into sets—a convenient feature for SOS, which we now describe.

Let us consider any application which embeds a piecewise-linear non-convex (usually, but not always concave) cost function, with breakpoints (a_i, b_i) defined for $(z, f(z))$. Attaching variables $\lambda_0, \lambda_1, \ldots, \lambda_N$ to the origin and endpoints of the segments, we express the function algebraically as:

$$f(z) - \sum_{j=0}^{N} b_j \lambda_j = 0 \tag{1}$$

$$z - \sum_{j=0}^{N} a_j \lambda_j = 0 \tag{2}$$

$$\sum_{j=0}^{N} \lambda_j = 1 \tag{3}$$

where at most two of the variables may be nonzero, and these must be adjacent, in standard separable programming fashion (Beale (1968)).

When the LP relaxation of such a model is solved we normally obtain an *inadmissible* solution which interpolates a cost below the actual piecewise-linear curve. Typically (but not necessarily) this involves the first and last variable. Let the current solution values be λ_j^* and compute the weighted sum of the a_j:

$$w = \sum_j a_j \lambda_j^*. \tag{4}$$

Let us suppose this value satisfies

$$a_r \leq w < a_{r+1}, \tag{5}$$

then (a_r, a_{r+1}) is called the *current interval*. We may now say that any solution which satisfies the requirements on the λ-variables must be on *either* the $(r+1)^{\text{th}}$ through last segments of the curve or on the first through r^{th} segments. This is equivalent to saying that:

$$\textit{Either} \quad \lambda_0 = \lambda_1 = \ldots = \lambda_{r-1} = 0 \quad \textit{or} \quad \lambda_{r+1} = \lambda_{r+2} = \ldots = \lambda_N = 0. \tag{6}$$

One can make a similar statement centered around the right end of the interval:

$$\textit{Either} \quad \lambda_0 = \lambda_1 = \ldots = \lambda_r = 0 \quad \textit{or} \quad \lambda_{r+2} = \lambda_{r+2} = \ldots = \lambda_N = 0. \tag{7}$$

These dichotomies may be used as a basis for branching in a branch and bound scheme in the same way that an integer variable with current value $x = n + f$, where f is fractional, is used to create a dichotomy: *either* $x \leq n$ *or* $x \geq n + 1$.

It is important to note that there are two dichotomies defined above. We would normally choose the one which gives the "strongest" branch consistent with the overall branch and bound scheme, but it is possible that there is no choice. This is because the right end point of the current interval (a_{r+1}) may be the only break point to the right of w with a nonzero weight attached. In that case, the second alternative of the second dichotomy does not exclude the current solution, and therefore does not create a valid branch. Similarly, if a_r is the only break point to the left of w with a nonzero weight attached, the first alternative in the first dichotomy does not exclude the current solution. We therefore need the option of making either pair of branches.

The analogy of the above procedure with branch and bound on general integer variables is striking, and is one of the reasons it proved easy to integrate in mathematical programming systems. One is thus tempted to ask whether the same thing could not be equally well accomplished using integer variables. The answer is that it could be accomplished, but not as well. For example one can force the λ-variables above to have the correct properties by introducing new zero–one variables δ_j, and (using 3 segments for illustrative purposes) the constraints:

$$\begin{aligned} \lambda_0 &\leq \delta_1 \\ \lambda_1 &\leq \delta_1 + \delta_2 \\ \lambda_2 &\leq \delta_2 + \delta_3 \\ \lambda_3 &\leq \delta_3 \\ \delta_1 + \quad \delta_2 \quad &+ \delta_3 = 1 \end{aligned}$$

(see Dantzig (1963) and Beale (1968)). This not only increases the model size, but leaves us with the problem of deciding on a branching strategy for the δ-variables.

A set of zero–one variables summing to one (such as we see above) is a *multiple–choice* set, which have remarked on before, and which occur frequently in many contexts. A good example is a choice of plant size to build — none, small, medium, and large — when there are economies of scale. If we represent the (size, cost) alternatives by (a_j, b_j) we obtain a representation exactly as in (1)–(3), except that now the rule is that at most one of the λ-variables may be nonzero. We shall call this an example of a special ordered set of type 1 (or S1 set). For obvious reasons, when two adjacent members may be nonzero the special ordered sets are considered of type 2 (or S2 sets).

The process of branching on the members of a multiple–choice set individually can be very inefficient. Once again the LP relaxation of the model is likely to produce a solution with two nonzero members (often the first and the last). If we choose to branch on one of these (say λ_0), we are faced with a rather weak choice. The two branches correspond to the decisions to build no plant ($\lambda_0 = 1$) or to build "something" ($\lambda_0 = 0$). Similarly, with λ_N we have the choice of building the maximum size plant or some indeterminate smaller size (if any). Neither of these branches may take us much closer to a valid solution.

If we compute the weighted solution (4) the LP is really telling us that it wants a plant with capacity in the current interval (a_r, a_{r+1}), though not necessarily at the full price. We may therefore consider the more useful alternatives of building a plant of size at least a_{r+1}, or at most a_r. This is expressed by the dichotomy:

$$\textit{Either} \quad \lambda_0 = \lambda_1 = \ldots = \lambda_r = 0 \quad \textit{or} \quad \lambda_{r+1} = \lambda_{r+2} = \ldots = \lambda_N = 0. \tag{8}$$

The branching procedure generated by such a dichotomy is even more obviously analogous to branching on a general integer variable. Notice that we have *not* required the variables in an S1 set to be integer. All that is important is that only one be nonzero. Integrality will naturally occur as a by-product if the λ_j and required to sum to one.

It should also be noted that the SOS branching rules specified above say nothing about the sum of the λ_j either. In most applications it is natural for them to sum to 1, but there are real applications in which we may not want the variables in a set to sum to any particular quantity (see Beale (1970)). The general special ordered set algorithm then computes a true weighted average:

$$w = \frac{\sum_j a_j \lambda_j^*}{\sum_j \lambda_j^*}$$

before finding the current interval (5). The a_j values (where $a_1 \leq a_2 \leq \ldots \leq a_N$) are known as *reference entries* and may be explicitly included in a user designated *reference row* of type (2) or otherwise supplied or generated. The dichotomy generated by (6) or (7), or (8), is then available as one of the choices in the branch and bound algorithm. The actual branches take the form of adjusting the first and last members of the set permitted to be nonzero, in the same way that lower and upper bounds of integer variables are modified.

The initial branching strategies implemented in UMPIRE, for both ordinary integer variables and SOS, were essentially the same as those implemented in LP/90/94— depth first (LIFO) search coupled with use of penalties to choose branching variable and branch. These were initially successful, but as problems grew larger and more complex, the limitations of these strategies became apparent. Even though the original penalties(see Beale (1968), Driebeck (1966)) could be slightly strengthened (Tomlin (1970)), they were often not even of the same order of magnitude as the actual degradation in the objective function when a branch was made on the variable. J.P.H Hirst of British Petroleum (which at that time owned Scicon) suggested new branching strategies (Forrest et al (1974)). These included abandonment of the LIFO strategy (since random access store could now be used to store the tree), and use of the "Best Projection (BP)" criterion, for evaluating nodes, later generalized to "pseudocosts" (Benichou et al (1977)). These techniques considerably extended the problem solving power of UMPIRE and it became dominant in the UK service bureau environment of its day. A version was also developed for use by Computer Science Corporation (CSC) for use with their time-sharing system in the US.

3 SCICONIC and Global Optimization

UMPIRE had been written for the Univac 1108 computer system. As this became obsolescent, Scicon developed a new Mathematical Programming System, SCICONIC, again under direction of Martin Beale, the bulk of the actual implementation being carried out by J.J.H. Forrest, and subsequently R.C. Daniel.

SCICONIC omitted GUB, which had heavily influenced the design of UMPIRE, but introduced a number of new discrete features, including semi–continuous variables (variables required to be either zero or in some positive interval), several new bound types in the input, and a much more elaborate treatment of SOS intended to enable the global optimization of separable functions. This work was described in two papers by Beale and Forrest(1976, 1978), and carries through the early work done on separable programming, with automatic refinement (Beale (1968, 1970)) to the non-convex case.

In addition to new methods for interpolating non-convex functions (so that S2 sets could be refined), this work introduced a new idea which seems to have virtually disappeared. This was the idea of "pseudo shadow prices", for branch choice with SOS. Essentially this idea generalized pseudo-costs, which are very effective for ordinary integer variables, but not so obviously applicable to SOS (Forrest et al (1974)). Pseudo-costs were developed because the penalties previously used for branching seriously underestimated

the cost of removing integer infeasibilities. Now in principle, if an SOS (say of type S2) is unsatisfied, the LP relaxation will have constructed a convex combination of members of the set, $a^U(w_k)$,which we can think of as the contribution of the kth set for the interpolated value w_k defined by the reference row. If we evaluate the set correctly (without linear interpolation) at w_k we would obtain a representative column $a^C(w_k)$. If the shadow prices of the LP relaxation are π_i, then we might estimate the change in objective when the set is satisfied to be:

$$\sum_{i=1}^{n}(a_i^U(w_k) - a_i^C(w_k))\pi_i$$

This idea is attractive in the same way as penalties are attractive—it uses data (the π_i) which fall out of the LP relaxation—but suffers from a similar weakness. The π_i give only the cost of infinitesimal changes, which may be small or even zero, when the cost of a significant change is required. Beale therefore proposed using "pseudo shadow prices" p_i^+ and p_i^- to estimate the cost per unit change (up or down) of each $(a_i^U(w_k) - a_i^C(w_k))$. We shall not go into further detail here (see Beale and Forrest (1976)) and Beale (1985)), but point out that the new S2 set facilities enabled SCICONIC to effectively solve some challenging global optimization problems arising in the oil industry (Beale and Forrest (1978)).

4 Further development of Special Ordered Sets

The SOS set concept has proved remarkably amenable to extensions. The developers of MPSX/370 (see Benichou et al (1977)) generalized the zero—one multiple—choice concept to cover variables satisfying

$$\sum_{j\in J}\delta_j - \sum_{h\in H}\delta_h = 1 - |H|$$

and called them "special ordered set", though they really have a rather tenuous connection. These are now sometimes known as Special Ordered Sets of type 3 (S3 sets). Several practitioners also realized that additional branching power could by gained by allowing SOS to overlap.

Extension of the SOS concept beyond separable nonlinearities was to prove a major theme in Beale's work in his later years. Working with R.C. Daniel and J.J.H. Forrest, Beale proposed "Linked Ordered Sets" as a means of globally optimizing problems containing product terms $x_u f_u(y_j)$ (see Beale (1979, 1980), Beale and Daniel (1979)). Even more general application of the SOS concept proved possible. Extension to non-separable non-convex functions via simplicial sub-division is proposed in Tomlin (1981). In his last published paper on integer programming, Beale (1985) described an alternative approach—"Chains of Linked Ordered Sets"—actually implemented in SCICONIC/VM. Again we shall not go into more detail here, but observe that these further developments of SOS were inspired, as was so much of Beale's work, by the desire to solve real world customer problems more effectively.

5 Conclusion

E.M.L. Beale made many seminal contributions to Mathematical Programming, but one of his most important must surely be the early practical implementation of integer programming by branch and bound, and the subsequent development of so many powerful extensions of the branch and bound concept to other discrete and non-convex domains. These ideas were also notable for their real-world implementation in commercial mathematical programming systems and their application to customer problems. We are proud to have been his colleagues in some of these endevours.

References

[1] Beale, E.M.L. (1963). "Two transportation problems", in *Proceedings of the Third International Conference on Operational Research*, (G. Kreweras and G. Morlat, eds.), Dunod, Paris and English Universities Press, London, 780–788.

[2] Beale, E.M.L. (1968). *Mathematical Programming in Practice*, Pitman's, London, and Wiley, New York.

[3] Beale, E.M.L. (1970). "Advanced algorithmic features for general mathematical programming systems", in *Integer and Nonlinear Programming*, (J. Abadie, ed.), North Holland, Amsterdam, 119–137.

[4] Beale, E.M.L. (1979). "Branch and bound methods for mathematical programming systems", *Annals of Discrete Mathematics* **5**, 201–219.

[5] Beale, E.M.L. (1980). "Branch and bound methods for numerical optimization of non-convex functions", *COMPSTAT 80 Proceedings in Computational Statistics*, (M.M. Barritt and D. Wishart, Eds.), 11–20.

[6] Beale, E.M.L. (1985). "Integer programming", in *Computational Mathematical Programming*, (K. Schittkowski, ed.), NATO ASI Series F: Computer and System Sciences, Vol. 15, Springer-Verlag, Berlin, 1–24.

[7] Beale, E.M.L. and R.C. Daniel (1979). " Chains of linked ordered sets: A new formulation for product terms", Presented at the 10th International Symposium on Mathematical Programming, Montreal.

[8] Beale, E.M.L. and J.J.H. Forrest (1976). " Global optimization using special ordered sets", *Math. Prog.* **10**, 52–69.

[9] Beale, E.M.L. and J.J.H. Forrest (1978). " Global optimization as an extension of integer programming", in *Towards Global Optimization 2*, (L.C.W. Dixon and G.P. Szego, Eds.), 131–149, North Holland, Amsterdam.

[10] Beale, E.M.L. and R.E. Small (1965). "Mixed integer programming by a branch and bound technique", *Proc. of the IFIP Congress 1965*, (W.A. Kalenich, Ed.), 450–451, MacMillan, London and Spartan Press, Washington, DC.

[11] Beale, E.M.L., and J.A. Tomlin (1970). " Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables", in *Proceedings of the Fifth International Conference on Operational Research*, (J. Lawrence, ed.), Tavistock Publications, London, 447-454.

[12] Benichou, M., J.M. Gauthier, G. Hentges, and G. Ribière (1977). "The efficient solution of large-scale linear programming problems — some algorithmic techniques and computational results", *Math. Prog.* **13**, 280–322.

[13] Dantzig, G.B. (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.

[14] Driebeck, N.J. (1966). "An algorithm for the solution of mixed integer programming problems", *Management Sci.* **12**, 576–587.

[15] 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.

[16] Gomory, R.E. (1963). "An algorithm for integer solutions to linear programs", in *Recent Advances in Mathematical Programming* (R.L. Graves and P. Wolfe, Eds.), McGraw-Hill, NY, 269–302.

[17] Gomory, R.E. (1991). "Early integer programming", in *History of Mathematical Programming* (J.K. Lenstra et al., Eds.), North Holland, Amsterdam, 55–61.

[18] Land, A.H. and A.G. Doig (1960). "An automatic method for solving discrete programming problems", *Econometrica* **28**, 497–520.

[19] Markowitz, H.M. and A.S. Manne (1957). "On the solution of discrete programming problems", *Econometrica* **25**, 84–110.

[20] Martin, G.T. (1963). "An accelerated Euclidean algorithm for integer linear programming", in *Recent Advances in Mathematical Programming* (R.L. Graves and P. Wolfe, Eds.), McGraw-Hill, NY, 311–317.

[21] Shaw, M. (1969). "Review of computational experience in solving large mixed integer programming problems", in *Applications of Mathematical Programming Techniques* (E.M.L. Beale, Ed.), English Universities Press, London, 406–412.

[22] Tomlin, J.A. (1970). "Branch and bound methods for integer and non-convex programming", in *Integer and Nonlinear Programming*, (J. Abadie, ed.), North Holland, Amsterdam, 437–450.

[23] Tomlin, J.A. (1981). "A suggested extension of special ordered sets to non-separable non-convex programming problems", in *Studies on Graphs and Discrete Programming*, (P. Hansen, ed.), North Holland, Amsterdam, 359–370.

[24] Tomlin, J.A. (1988). "Special ordered sets and an application to gas supply operations planning", *Mathematical Programming 42*, 69–84.