

**A GENERALISATION OF SPECIAL ORDERED SETS TYPE 1:
A RESEARCH PROPOSAL**

Nicholas Beaumont, Alan Farley & Lee Gordon-Brown

*Working Paper 56/05
August 2005*

**DEPARTMENT OF MANAGEMENT
WORKING PAPER SERIES
ISSN 1327-5216**



Abstract on next page

Abstract

Many practical optimization problems (such as aircraft scheduling) can be expressed as Mixed Integer Programs (MIPs). Many important MIPs resist general solution methods, proving tractable only when their special structures are exploited. One common special structure models choosing one of a set of mutually exclusive actions (a worker is allocated only one of several tasks). We propose to generalise this structure to decisions that involve selecting a small number of non-mutually exclusive choices (NMECs) (a worker is allocated up to (say) three of several tasks).

Many practical MIPs are very difficult to solve. A common approach entails creating and examining a hierarchy of sub-problems. At each stage of the solution process there are a number of sub-problems that can next be chosen for examination. Critical to the effective solution of practical MIPs is choosing the right sequence of sub-problems to explore. If good choices are made, solving the original problem is easy. Conversely, if good choices cannot be made, the problem may prove intractable. Many criteria guiding choices have been proposed but an individual criterion can work well or badly on different kinds of problems, it is very difficult to anticipate what criteria will work well on a new kind of problem.

The research programme will concentrate on finding criteria appropriate to the NMEC problem but we anticipate that some criteria, especially those based on the properties of constraints and variables, will have more general application.

By utilizing this project's results, Australian business and governments will be able to deploy resources more efficiently and/or improve service, thus helping Australia remain competitive in a global economy whose barriers to trade are eroding. Enhanced analytical techniques will help governments allocate and price scarce resources such as water and social services optimally, and help quantify the opportunity cost of regulations e.g. those governing pollution and anti-competitive practices.

A GENERALISATION OF SPECIAL ORDERED SETS TYPE 1: A RESEARCH PROPOSAL

BACKGROUND

Single and Multiple Choice Constraints

One form of constraint (the single choice constraint or SCC) occurring frequently in Mixed Integer Programs (MIPs) models the allocation of a resource to one of several possible tasks. For example, it models a worker being assigned to one of several possible jobs; an investment being made in one of several years; or a shipment being carried by one of several different vehicles. Beale and Tomlin (1970) demonstrated that this kind of constraint could be handled in an efficient way. If one worker is to be allocated to one of 20 possible tasks, it is sometimes possible to implicitly and economically eliminate some possibilities without explicitly considering each of the 20 possible allocations. Existing methods rely on rather unsophisticated methods of dichotomizing the set of 20 tasks. We will investigate more systematic methods.

We propose to extend Beale and Tomlin's technique to the multiple choice constraint (MCC) exemplified by a worker being assigned a small number (say 2 or 3) of 20 jobs. It will similarly be possible to implicitly eliminate some combinations with computational advantage.

Criteria

The search amongst the myriad possible solutions for the optimal solution to an MIP is guided by a number of criteria (a random search would be hopeless). Most of these criteria are heuristics, based on "commonsense", not strong theoretical considerations. If the criteria were perfect there would be no difficult MIPs. Solving a difficult MIP invariably involves experimenting with different criteria, fine tuning those that seem to work best. Some criteria are specific to SCCs and we will generalise these to MCCs.

Most of the criteria now used are based on the properties of variables. This is because older software, unlike its modern counterparts, did not make it easy to access or modify constraints during the solution process. We believe that useful new criteria guiding the search for optimal solutions can be obtained by considering the properties of constraints. Such criteria would have general application.

Significance

This research is significant because many MIP problems have practical importance and their solution would be economically beneficial. Faster solution of a problem allows exploration of more scenarios. Some problems expressible as MIPs pertain to the location of facilities (e.g. factories, oil wells, shops, community facilities and military bases); optimal design of circuits and communication networks; scheduling of aircrew, aircraft, factory production, transport, and power generation; timetabling and rostering; blending (exemplified by oil refineries); and design of financial portfolios (Williams, 1993).

Three current areas of interest are: the location of public facilities (e.g. schools) so as to (inter alia) optimize citizens' access; rationalising a set of facilities (e.g. factories, shops, branches, and transport arrangements); and ascertaining the opportunity cost of regulations pertaining to safety, pollution, or competition. The demand for the second kind of analysis has been increased by takeovers, globalisation, and diminished barriers to trade. The solution of hitherto unsolved MIPs would, by allowing optimal allocation of resources manifest for example in improved production schedules, location of facilities, staff scheduling, and the scheduling of public transport; have large and diffuse global benefits.

These benefits arise because the research will result in:

Easier methods of preparing the required input data. Preparing and formatting the input data for a large MIP is time-consuming and error prone. The proposed methodology will simplify the expression of MCCs in input data.

Faster solution of MIPs containing MCCs.

The solution of hitherto unsolved MIPs containing MCCs.

The development of criteria allowing a more efficient search of the possible solutions. Some of these criteria will have general application.

The faster solution of MIPs containing MCCs will be partly attributable to our exploiting special structures and partly attributable to developing improved criteria guiding the search amongst solutions. Some of these criteria will be specific to MCCs. By considering the properties of constraints, we can develop a set of new criteria that could be applied in all MIPs.

APPROACH AND METHODOLOGY

The Standard MIP Model

A MIP can be expressed as:

Max $z = \mathbf{c}\mathbf{x}$ subject to:

$$\mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\delta} \leq \mathbf{b}$$

$\mathbf{x} \geq 0, \boldsymbol{\delta} \in \{0,1\}$ where

c, x, δ, A, B, b are $1 \times n, n \times 1, p \times 1, m \times n, m \times p, m \times 1$ respectively. (1)

This model (1) can be easily extended to incorporate \geq or equality constraints, negative or free variables, minimization and general integer variables. For example, a general integer variable $0 \leq z \leq 14$ can be re-expressed as $z = \delta_1 + 2\delta_2 + 4\delta_3 + 7\delta_4$ $\delta_i \in \{0,1\}$ $i = 1, 2, 3, 4$.

Algorithms for Solving MIPs

MIPs are potentially difficult to solve because even if $p=20$ (a very small number by modern standards) there are potentially a million problems (2^{20}) to be investigated. Although other methods have been considered (Geoffrion & Marsten, 1972) the standard generic method of solving MIPs is "Branch and Bound" (BB) perhaps better expressed as "Branch and Cut". A BB algorithm is:

Step 0. Form the relaxation of the original problem by replacing the binary constraints $\delta_i \in \{0,1\}$ $i \in M$ by $0 \leq \delta_i \leq 1$ $i \in M$. U , the set of unsolved problems, initially comprises the relaxation alone. Solve this problem, if it is infeasible or unbounded, exit and reformulate, otherwise set $BEST = -\infty$.

DO steps 1, 2, 3 until U is empty.

Step 1 [Selection]. Choose a problem (denoted by P) in U , replace U by $U - P$ and optionally modify P as described in "Problem modification" below. Solve P denoting its solution by $\mathbf{x}, \boldsymbol{\delta}$ and its optimal function value by z ($z = -\infty$ if P is infeasible).

Step 2. If $z > BEST$ and all δ are binary, set $BEST = z$ and store the solution \mathbf{x}, δ . Optionally modify some or all of the problems in U as described in “Problem Modification” below.

Step 3. [Arbitration] If $z > BEST$ and not all δ are binary, choose an element of δ that is not binary (say $\delta_i, i \in M$) and form two problems P_1 and P_2 in which the bound $0 \leq \delta_i \leq 1, i \in M$ is replaced by $\delta_i = 0$ and $\delta_i = 1$ respectively (this is called arbitrating a variable). Add P_1 and P_2 to U .

Step 4. If $BEST = -\infty$ then no integer feasible solution has been found, the problem is integer infeasible. If $BEST$ is finite, the most recently stored solution (\mathbf{x}, δ) is an optimal feasible solution.

There are 2^p potential “twigs” in the solution “tree”. “Branches” can be “pruned” i.e. not considered further, by demonstrating that they are either an integer solution, infeasible, or bounded, i.e. cannot yield a solution better than one already found in Step 2. It is desirable to find a good solution early so that most branches are pruned.

Two choices in the algorithm can profoundly affect solution times: The problem chosen in step 1 and the variable selected for arbitration in Step 3. If perfect choices could be made, all integer feasible MIP problems would be easy. In practical problems there may be hundreds or thousands of entities to arbitrate and the set U can become extremely large (its potential size increases exponentially with the number of binary variables and SOS1s).

Problem Modification

Especially when an integer solution has been found, it may be useful to modify some or all problems in U by:

- (1) Tightening variables’ bounds (Hoffman & Padberg, 1991; Williams, 1993, pp 213-214; Zionts, 1969). In some cases this may allow variables to be fixed, constraints to be replaced by bounds, and/or constraints recognised as redundant. Most commercial LP packages have a “reduce” option that automates some of these tasks.
- (2) Tightening constraints (Lougee-Heimer, 2001; Ogryczak, 1996; Williams, 1993, pp 213-214).
- (3) Expressing a lower bound on the objective function as a constraint.
- (4) Introducing general cutting planes (Gomory, 1963; Wilson, 1990) and especially Balas et al. (1996).
- (5) Abandoning development of the tree, modifying the problem, and restarting the solution process (Hoffman et al., 1991).

MODIFYING THE BB ALGORITHM FOR SPECIAL ORDERED SETS (SOS1S)

SOS1 Definition

Single-choice constraints appear frequently in Mixed Integer Programs (MIPs), e.g. Williams (1993, problems 4 and 10). Their mathematical expression is:

$$\sum_{i \in M} \delta_i = 1 \text{ where:}$$

i is an index,

M is an index set,

δ_i is a binary variable: a variable that must be either 0 or 1. (2)

For example, $i_3 = 1$ might imply that an investment takes place in year 3. The variables $\delta_i, i \in M$, exactly one of which is non-zero; comprise a Special Ordered Set Type 1 (SOS1). This definition can be extended to: (i) A set of non-negative variables of which at most one is non-zero or (ii) (2) being replaced by $\sum_{i \in M} \delta_i \leq 1$ (Hummeltenberg, 1984, p 2). These variations make no essential difference to the proposed approach.

The naive way of finding which $\delta_i, i \in M$ is optimal is to test each $\delta_i, i \in M$ separately (a total of $m = |M|$ sub-problems must be solved), this does not exploit the SOS1 special structure. Beale & Tomlin (1970) found that it is sometimes more efficient to create two branches of the solution tree by augmenting (2) with the dichotomy:

$$\sum_{i \in M_1} \delta_i = 0 \vee \sum_{i \in M_2} \delta_i = 0 \text{ where } M_1 \cup M_2 = M \quad (3)$$

where M_1, M_2 have about equal sizes. Each term can be conveniently expressed by manipulating the bounds of $\delta_i, i \in M$. If necessary, M_1, M_2 can each be dichotomised into subsets $M_{12}, M_{22}, M_{21}, M_{22}$, and the process repeated. In the worst possible case this will require the solution of $2m - 1$ subproblems (not m). In practice, the set M often has a natural order. For example if the set M denotes the years 2001-2020 it may be possible to tell, after one dichotomisation, whether an investment should be made in the first or second decade. This suggests the possible computational efficiency of arbitrating subsets of variables comprising a SOS1 rather than single variables.

Modifying the BB Algorithm for Multiple Choice Constraints

An objective of the proposed research is to extend the methods used to accelerate the solution of MIPs that include SOS1s to MIPs that include constraints (multiple choice constraints) expressed as:

$$\sum_{i \in M} \delta_i = k \text{ where:}$$

i is an index,

M is an index set,

δ_i is a binary variable: a variable that must be either 0 or 1

k is an integer small compared with $|M|$.

This constraint drastically increases the number of choices that a naive algorithm must explore. If $n = |M|$, that number is C_k^n .

The set $\delta_i, i \in M$ is denoted by the term SOS(k) (the term SOS2 is already used for a different purpose). This definition can be extended in two ways with no essential change to algorithms: (i) A set of non-negative variables of which at most k are non-zero (ii) If the constraint is $\sum_{i \in M} \delta_i \leq k$, it can be converted to an equality by inserting an integer slack variable with bounds $[0, k]$.

Suppose that an organisation can initiate *two* investments in the years 2001-2020 (but not in the same year). The variables $\delta_i = 1, i \in M = \{1, 2, 3, \dots, 20\}$ if an investment is made in year i ,

otherwise $\delta_i = 0$. The constraint can be formulated as $\sum_{i \in M} \delta_i = 2, M = \{1, 2, 3, \dots, 20\}$. Analogous to (3), this SOS(2) can be efficiently arbitrated by introducing the dichotomy:

$$\sum_{i \in M_1} \delta_i = 0 \vee \left(\sum_{i \in M_1} \delta_i = 1 \wedge \sum_{i \in M_2} \delta_i = 1 \right) \vee \sum_{i \in M_2} \delta_i = 0 \quad (4)$$

where $M_1 \cup M_2 = M$ and M_1, M_2 have about equal sizes and $\sum_{i \in M} \delta_i = 2, M = \{1, 2, 3, \dots, 20\}$ is retained.

The left term and right terms of (4) are easily and efficiently arbitrated by setting variables' upper bounds to zero. The middle term is a conjunction of two Special Ordered Sets Type 1 that can be arbitrated by well know methods (Beale et al., 1970). If necessary, M_1, M_2 can be dichotomized into subsets $M_{12}, M_{22}, M_{21}, M_{22}$, and the process repeated. We anticipate that sometimes one of the terms (e.g. $\sum_{i \in M_1} \delta_i = 0$ of (4)) will be prunable and efficiency will be gained by not having to test individual $\delta_i, i \in M_1$.

The General Case

Step 2 of the algorithm can be extended to incorporate the arbitration of SOS(k)s. If an SOS(k) comprising the variables $\delta_i, i \in M$ is selected for arbitration, additional nodes of the problem tree P_0, P_2, \dots, P_k are formed by dichotomizing M into two sets M_1, M_2 where $M = M_1 \cup M_2$, of about equal size, retaining the original constraint $\sum_{i \in M} \delta_i = k$ and adding the disjunction:

$$\sum_{i \in M_1} \delta_i = 0 \vee \sum_{i \in M_1} \delta_i = 1 \vee \sum_{i \in M_1} \delta_i = 2 \vee \dots \vee \sum_{i \in M_2} \delta_i = 2 \vee \sum_{i \in M_2} \delta_i = 1 \vee \sum_{i \in M_2} \delta_i = 0. \quad (5)$$

This implies that the original SOS(k) can be expressed in terms of SOS(k)s of order $\lceil k/2 \rceil$ ($k/2$ rounded up).

We note that, for some problems, it may be appropriate to replace the bound $\delta_i \in \{0, 1\}, i \in M$ by $\delta_i \in \{0, k\}, i \in M$.

Examples of SOS(k)

Williams (1993, Part 2) gives a number of problems containing SOS(k)s (numbers 2, 4, 7, 10, 16, 19). In most of these cases the index set is too small for the proposed technique to be useful but it might be if the problems were enlarged.

Constraints of form $\sum_{i \in M} \delta_i = k$ and $\sum_{i \in M} \delta_i \leq k, k$ integer $\square |M|, \delta_{i \in M} \in \{0, 1\}$ arise frequently in MIPs.

They are a generalisation of assignment constraints (for which $k=1$) and are known as set covering constraints. They are interpretable as a resource being allocated to k different demands chosen from the set M . These kinds of constraints characterize generalized assignment problems (one resource can be assigned a limited number of tasks or one task uses a limited number of resources) and are widely used in problems of current interest such as:

- Telecommunications (number or capacity of routes available for network reliability and survivability)
- Logistics hub location (location of shops such that each outlet is within a minimum distance of k shops)
- Shift and crew assignments (k operators per shift) and machine scheduling (k machines per activity)
- Air transport capacity (number of air traffic controllers, landing and departure windows, parking or air bridge windows)
- Partitioning problems (balancing k attributes in partitions such as electorates)
- Transport scheduling (a taxi can take up to k passengers)
- Energy or utility supply scheduling (security surplus available in each supply period, number of generators operating, planned maintenance scheduling)
- Emergency services (security personnel, paramedics, emergency vehicles), and
- Health and analogous services, (nurses, theatres, wards, or consumables can each be allocated several tasks or be allocated to several different groups)

Other examples that can be modeled as SOS(k)s include:

- Blending: a blend can contain at most (or at least) k different ingredients
- Irregular polygon cutting stock problems (object positioning constraints)
- Disjunctions: at least k of n constraints must be satisfied (Sherali & Shetti, 1980)
- Some problems include constraints that are variants of the form $\sum_{i \in P} \delta_i - \sum_{i \in Q} \delta_i = 0$ $\delta_i \in \{0, 1\}$ $i \in P \cup Q$. If $\delta'_i = 1 - \delta_i$ $i \in Q$ this constraint can be rewritten as

$$\sum_{i \in P} \delta_i + \sum_{i \in Q} \delta'_i = |Q|$$

The problems listed have obvious importance: their optimal solution would enable resources to be more efficiently allocated or scheduled or for service levels to be maintained with less resources being used.

Dichotomizing Index Sets

The proposed extension to the BB algorithm introduces another arbitrary choice made at step 3 during expansion of the tree namely, how M is dichotomized. The dichotomization can profoundly affect the size of the solution tree. There is an obvious basis for dichotomization If M has a natural order based on dates or increasing capacity, size, or efficiency. Where this is not so, we intend to deduce from the original formulation or from the detail of a sub-problem, a potentially fruitful dichotomization. We note that sequencing the set M is overkill, dichotomization suffices.

In any solution, each variable belonging to an SOS(k) is represented by a column of coefficients. It may be possible to group elements on the basis of the similarity of columns (several kinds of distance metric could be used) (Lougee-Heimer, 2001; Martin & Sweeney, 1983; Ogryczak, 1996;

Yu-Hsien Lin, Edward Bricker, & L., 2002). It might be desirable, in a partial solution, to split a set of resources into those that were already substantially allocated and those that were not.

Resources and Tasks

Many of the scheduled tasks required to complete the project have three phases: conceptualization, programming and testing, and exploitation. Conceptualization entails having an idea and expressing that idea in terms of detailed specifications that a program programmer can understand and implement. Programming and testing entails writing computer programs and verifying that they work correctly by testing them on a variety of problems whose behavior is known. Exploitation entails using the programs to compare new and old algorithms' performances. Exploitation will be protracted because the performance of packages embodying the conventional and proposed algorithms will vary with parameters and amongst test problems.

It will be impossible to completely separate these activities. During exploitation it is inevitable that errors in the code will be detected and that new methods of accelerating solution will emerge.

The project can be broken into five stages:

Stage 1. Writing and testing two computer packages. The "SOS(k)" and "conventional" packages (each comprising many programs) will embody the envisaged and conventional algorithms respectively. Both packages will contain the option of applying a variety of "preprocessing" techniques at each node of the tree and will have parameters that allow the user to influence how the tree is traversed. The most important of these will influence: (i) which sub-problem will next be solved (ii) which variable will next be arbitrated and, in the case of SOS(k)s, (iii) how each set will be dichotomized. The two packages will be tested on a variety of problems whose solutions are known. Collections of problems are available on the World Wide Web e.g. at <http://www.brunel.ac.uk/depts/ma/research/jeb/info.html> (J.E.Beasley, 1990).

Stage 2. The two packages, incorporating basic selection criteria for selection and arbitration will be applied to a set of test problems (some of which will be large and demanding) and their performances compared. It is hoped that the performance of the SOS(k) suite will be markedly superior, at least for some kinds of problem. There are several possible performance criteria and difficulties in comparing the performance of different algorithms. Criteria to be used (here and stages 3 and 4) include:

- The resources (including time) required to find an optimal solution. For some complex problems, it may not be proven that the best solution found after several hours of computer time he is optimal.
- The resources required to find the first optimal solution.
- The best solution found using finite resources, for example one hour of computer time.
- The time required to prepare data files and the size of the data files.

Suppose that the new algorithm solves problems ten times faster than the conventional algorithm. The algorithms are not strictly parallel because: (i) it may not be possible to exactly translate criteria used in the conventional algorithm into similar criteria in the new algorithm. (ii) It would be agreeable to force the two packages to traverse the solution tree in an exactly parallel way. Unfortunately, this fine control is not possible if there are alternate solutions of a nodal problem. It will therefore be difficult to unambiguously allocate the improvement to the basic algorithm or to different criteria.

Stage 3. The efficacy of cutting planes derived from SOS(k)s and the effect of different ways of dichotomizing SOS1 and SOS(k) will be investigated. A set of problems will be repeatedly run with

different cutting planes (Wilson, 1990) and different methods of dichotomization. We will ascertain whether any method or combination of methods is universally superior, or superior for a particular kind of problem.

Stage 4. The effect of new arbitration criteria, predominantly based on properties of constraints will be investigated. The performance of the new criteria will be compared with extant criteria by testing them with a disparate set of problems. The new criteria might be universally superior, or superior for a particular kind of problem. We note that some criteria may facilitate finding a first integer solution but be less useful thereafter.

Stage 5. The results will be communicated to the academic and practicing operations research communities. Organizations (such as Dash Optimization and ILOG) that sell optimization software will be approached and invited to incorporate the new algorithms in their software.

BENEFITS OF THE OUTLINED RESEARCH

The outcomes comprise improved methods of solving Mixed Integer Programs. Such methods would be beneficial (by improving the output/input ratio i.e. productivity).

Some practical problems could be solved faster. This is this would be especially useful in situations requiring allocation of resources in response to demands (e.g. vehicle traffic) that change unpredictably on an hourly basis. Rational real-time allocation of resources would be possible and advantageous.

Difficult practical problems that had not been hitherto solved or for which only approximate solutions were available might yield to the proposed techniques.

The benefits would not be specific to particular industries or countries, although having in Australia researchers with the expertise to apply the new techniques and a history of successful consulting with industry is likely to give Australia an advantage. Organisations (predominantly large organisations) that have the resources to maintain their own Operations Research teams would be the prime beneficiaries. Such organizations would have the resources needed to evaluate and implement the new techniques.

At least two foreign consulting groups (ILOG www.ilog.com and Dash Optimization (www.dashoptimization.com) develop, sell, and advise on the application of mathematical programming techniques in industry. If such groups could be persuaded to incorporate the new techniques in their packages, the techniques would be available to general users.

Topics meriting publication will be:

- A comparison of the proposed and conventional methods of arbitrating SOS(k)s.
- The effect of pre-processing rules and cutting planes peculiar to SOS(k)s.
- The effect of new methods of dichotomizing SOS(k)s.
- The effect of new, constraint-based arbitration criteria.

CONCLUSION

The proposed research is based on recognizing that a particular kind of constraint (the NMEC) occurs frequently in MIPs and that NMEC constraints have a special structure that allows their efficient arbitration. The ways in which sets of variables comprising NMECs are dichotomised has

not been well researched and merits investigation. Most criteria used in arbitration are based on the properties of variables, not constraints. This might be attributable to history; because problems are stored in column order, it was difficult to extract and manipulate constraints. Modern software greatly lessens this difficulty. We will consider arbitration criteria derived from the properties of rows and variables.

REFERENCES

- Balas, E., Ceria, S., & Cornuejols, G. 1996. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9): 1229.
- Beale, E. M. L., & Tomlin, J. A. 1970. *Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables*. Paper presented at the 5th International Conference of O. R., Tavistock, London.
- Bixby, R. E. 2002. Solving real-world linear programs: A decade of progress. *Operations Research*, 50(1): 3-15.
- Geoffrion, A. M., & Marsten, R. E. 1972. Integer programming algorithms: A framework and state-of-the-art survey. *Management Science*, 18: 465-491.
- Gomory, R. E. 1963. An algorithm for integer solutions to linear programs. In R. L. Graves, & P. Wolfe (Eds.), *Recent Advance in Mathematical Programming*: 269-302. New York: McGraw-Hill.
- Hoffman, K. L., & Padberg, M. 1991. Improving LP-Representations of Zero-One Linear Programs for Branch and Cut. *Journal on Computing*, 3(2): 121-134.
- Hummeltenberg, W. 1984. Implementations of Special Ordered Sets in MP Software. *European Journal of Operational Research*, 17(1): 1-15.
- J.E.Beasley. 1990. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11): 1069-1072.
- Lougee-Heimer, R. 2001. A note on coefficient adjustment using SOS constraints. *Operations Research*, 49(1): 175-177.
- Martin, R. K., & Sweeney, D. J. 1983. An ideal column algorithm for integer programs with special ordered sets of variables. *Mathematical Programming*, 26: 48-63.
- Ogryczak, W. 1996. A note on modeling multiple choice requirements for simple mixed integer programming solvers. *Computers & Operations Research*, 23(2): 199.
- Sherali, H. D., & Shetti, C. M. 1980. *Optimisation with disjunctive constraints*. Berlin: Springer-Verlag.
- Williams, H. P. 1993. *Model building in mathematical programming* (3 ed.). Chichester, England: Joh Wiley & Sons Ltd.
- Wilson, J. M. 1990. Generating Cuts in Integer Programming with Families of Special Ordered Sets. *European Journal of Operational Research*, 46(1): 101-108.
- Yu-Hsien Lin, Edward Bricker, & L., D. 2002. Connecting special ordered inequalities and transformation and reformulation technique in multiple choice programming. *Computers & Operations Research*, 29(10): 1441-1445.
- Zionts, S. 1969. Towards a Unifying Theory for Integer Programming. *Operations Research*, 17: 359-367.