Comput. & Ops. Res. Vol. 13, No. 5, pp. 533–549, 1986
Printed in Great Britain.

0305-0548/86 $3.00 + 0.00

FUTURE PATHS FOR INTEGER PROGRAMMING AND LINKS TO ARTIFICIAL INTELLIGENCE

FRED GLOVER*

Center for Applied Artificial Intelligence, Graduate School of Business, University of Colorado, Boulder, CO 80309, U.S.A.

Scope and Purpose—A summary is provided of some of the recent (and a few not-so-recent) developments that offer promise for enhancing our ability to solve combinatorial optimization problems. These developments may be usefully viewed as a synthesis of the perspectives of operations research and artificial intelligence. Although compatible with the use of algorithmic subroutines, the frameworks examined are primarily heuristic, based on the supposition that effective solution of complex combinatorial structures in some cases may require a level of flexibility beyond that attainable by methods with formally demonstrable convergence properties.

Abstract—Integer programming has benefited from many innovations in models and methods. Some of the promising directions for elaborating these innovations in the future may be viewed from a framework that links the perspectives of artificial intelligence and operations research. To demonstrate this, four key areas are examined: (1) controlled randomization, (2) learning strategies, (3) induced decomposition and (4) tabu search. Each of these is shown to have characteristics that appear usefully relevant to developments on the horizon.

1. INTRODUCTION

Integer programming (IP) has gone through many phases in the last three decades, spurred by the recognition that its domain encompasses a wide range of important and challenging practical applications. Two of the more prominent landmarks in the development of the field have undoubtedly been the emergence of the cutting plane and branch and bound approaches. As general solution strategies, these approaches have drawn on concepts from diverse areas including number theory, group theory, logic, convex analysis, nonlinear functions, and matroid theory [1–7].

From the theoretical side, cutting planes have received the greatest attention, though from a broad perspective the distinction between cutting plane and branch and bound methods blurs. Indeed, branch and bound may be viewed as *provisional cutting*. From the practical side, the most effective general purpose methods have relied heavily on branch and bound, conceiving branch and bound in its standard (narrower) sense, where the collection of provisional cuts derives simply from constraining integer variables to satisfy lower and upper bounds. Doses of cutting plane theory have been used to improve the basic branch and bound framework, chiefly by generating cuts to be added before initiating the branch and bound process (or in some cases just prior to selecting a next branch) [8–14]. The cuts used, however, are typically those that are easily derived and generated. The more labyrinthine and esoteric derivations have not so far demonstrated great practical utility.

Implicit in cutting methods and branch and bound methods are the allied notions of problem relaxation and restriction (enlarging and shrinking the feasible region) [15–20]. Problem relaxation has found particular application by means of the Lagrangean and surrogate constraint strategies, both of which have achieved their greatest successes on problems with special structures [21–25]. Indeed, it is often noted that the chances of developing a widely effective general purpose method are slim. Many of the interesting practical IP problems have very evident special structures, and the diversity of these special structures is sufficient that a procedure designed to do well for one of them is very likely, on the basis of experience in the field, to do poorly for many others.

This is not to say that certain *principles*, adapted appropriately to different settings, may not prove of wide value. On the contrary, a growing collection of such principles marks precisely the direction in which the field of IP methodology is evolving. We have already inherited a number of these from the

*Fred Glover received the doctorate from Carnegie–Mellon in 1965, and served on the faculty at the University of California, Berkeley, and the University of Texas and the University of Minnesota. He is the author of over 150 published papers in Management Science and related fields.

fundamental early developments: as represented by the "cutting plane principle", the "Benders partitioning principle", etc. It is also increasingly evident that modeling is especially important to practical application, not only for its links to solution efficiency, but equally for its role in fostering understanding and communication. Notable contributions to this area are provided by models associated with disjunctive programming [6, 8, 15, 26–30] and network-related formulations (or "netforms") [24, 31–34].

The emergence of strategies that can be adapted to take advantage of diverse problem conditions marks what I believe to be the wave of the future in IP, giving us power to solve a significantly increased number of IP problems effectively. Current indications point to the opportunity to obtain near-optimal solutions, by tailoring principles to special structures, for problems involving many thousands or even *millions* of IP variables [21, 23, 24, 35]. Clearly, such an outcome would have been unthinkable in the not so distant past. The blend of heuristics and algorithms (viewing heuristics from a perspective that transcends some of those commonly embraced) opens the possibility of a continuing stream of "surprises" of this nature. Integer programming, seen as one of the most difficult and challenging of fields, is evolving into one that promises to yield a remarkable collection of successful case histories.

The foundation for this prediction derives, perhaps surprisingly, from the recent remarriage of two disciplines that were once united, having issued from a common origin, but which became separated and maintained only loose ties for several decades: operations research and artificial intelligence. This renewed union is highlighting limitations in the frameworks of each (as commonly *applied*, in contrast to *advocated*), and promises fertile elaborations to the strategies each has believed fruitful or approaching combinatorial complexity.

Rather than argue this point abstractly, or attempt to trace historical threads to give due credit to which strategic frameworks or perspectives owe more heavily to the influence of AI or OR, I will describe a few of the strategies that appear to hold promise based on experience currently available, and speculate on future directions.

As prelude to this undertaking, it is worthwhile to ask why theoretical advances in IP methods have not more frequently led to corresponding computational advances. One possible answer is that the critical increment of theory that will lead to computational breakthrough lies just beyond our reach, and will soon be discovered. A more likely hypothesis, I suspect, is that a procedure with special convergence properties, demonstrable by theorem and proof, must be somewhat rigidly structured—and this must impose limitations in turn on performance. Beyond a point, flexibility can be the enemy of theory, for it allows the crucial "if-then" chain to be broken. Yet rigidity can be equally the enemy of effective search. In the face of combinatorial complexity, there must be freedom to depart from the narrow track that logic single-mindedly pursues, even on penalty of losing the guarantee that a desired destination will ultimately be reached. ("Ultimately" may be equivalent to "a day before doomsday" from a practical standpoint.)

In brief, effective strategies for combinatorial problems can require methods that formal theorems are unable to justify. That is not to say such methods lack "structure". Complete flexibility has its parallel in complete randomization, which is scarcely a useful basis for complex problem solving. Methods that are "intelligently flexible" lie in some yet-to-be-defined realm to which theorems (of the familiar sort) do not apply, yet which embraces enough structure to exclude aimless wandering. We currently face the challenge of identifying features shared by the more promising members of this realm.

Perhaps the most conspicuous limitation of a heuristic method for problems involving discrete alternatives is the ability to become trapped at a local optimum. If we broadly allow a local optimum to include reference to solutions that are infeasible, then it is reasonable to say that local optimality is the crucial issue for a good heuristic approach to wrestle with.

From this point of view, it is useful to organize the exploration of good methods for discrete problems around strategies for transcending local optimality. Of the variety of heuristic classifications that might be discussed, I will focus on four: (1) controlled randomization, (2) learning strategies, (3) induced decomposition and (4) tabu search. These classifications, while not by any means exhaustive, are conjectured to be particularly relevant to developments on the horizon.

2. CONTROLLED RANDOMIZATION

Controlled randomization is perhaps the oldest strategy for seeking to overcome local optimality in combinatorial optimization, and classically takes two forms. The first is the well-known "random restart" approach, which injects a randomizing element into the generation of an initial starting point to which a heuristic is subsequently applied. (The approach may alternately be conceived as employing two heuristics, the outcome of the first providing the starting point of the second.) Depending on the nature of procedure for obtaining such a starting point, the "randomizing element" may be more systematic than random: for example, one may choose equally spaced values of a parameter along an interval rather than randomly sample such values from a selected distribution.

The second classical version of this approach is the "random shakeup" procedure which, instead of restarting, periodically generates a randomized series of moves that leads the heuristic from its customary path into a region it would not otherwise reach. (The term "random shakeup" is used in preference to "random perturbation" because the degree of departure may be more significant than perturbation usually connotes.) In the framework commonly employed, a criterion is established for differentiating a move as improving or nonimproving (though a broader "multicriteria" framework has been found useful in [35]), and the purpose of the randomizing element may be viewed as that of admitting nonimproving moves which would normally be disregarded. The point at which to activate the randomizing element (for example, waiting until no improving moves are available) is a choice rule of the procedure.

Recently a special case, or more properly a refinement, of the random shakeup approach has attracted a good deal of attention. Called *simulated annealing*, this procedure has been heralded as a new and powerful methodology for combinatorial problems, with implications for the field of artificial intelligence [36, 37]. The name "simulated annealing" derives from the intent to pattern the approach after the physical process of annealing, which is a means for reducing the temperature of a material to its low energy or ground state. Such a state may be viewed as analogous to an "optimum", whereon the energy level may accordingly be viewed as the value of an objective function to be minimized.

The annealing process begins with a material in a melted state and then gradually lowers its temperature, analogous to decreasing an objective function value by a series of "improving moves". However, in the physical setting the temperature must not be lowered too rapidly, particularly in its early stages. Otherwise certain "locally suboptimal" configurations will be frozen into the material and the ideal low energy state will not be reached. To allow a temperature to move slowly through a particular region corresponds to permitting nonimproving moves to be selected with a certain probability—a probability which diminishes as the energy level (objective function value) of the system diminishes. Thus, in the analogy to combinatorial problem solving, it is postulated that the path to an optimal state likewise begins from one of diffuse randomization, somewhat removed from optimality, where nonimproving moves are initially accepted with a relatively high probability which is gradually decreased over time.

The form of the process for the purpose of simulation may be specified as follows [37, 38]. Potential moves of the system, viewed as small displacements from its present state, are examined one at a time, and the change c in the objective function value is calculated. If $c < 0$, indicating an improvement, the move is automatically accepted. Otherwise the move is accepted with probability $P(c) = \exp(-c/kT)$, where T is the current objective function value (temperature) and k is a constant adapted to the application (Boltzmann's constant in the physical setting). The heuristic adaptation of k at different levels of T is referred to as creating an "annealing schedule".

In spite of the publicized success of this method, an interesting study [39] discloses that it has noteworthy limitations in application to the traveling salesman and p-median problems. Intuition strongly suggests, however, that a successful adaptation must be done with a somewhat charitable latitude to depart from the publicized framework. For example, in the implementation of [39], which was highly faithful to the simulated annealing framework, the solutions used for a basis of comparison were those obtained after a certain elapsed time, rather than the best obtained through all stages of the process. While a material may stabilize at an "optimum" ground state after a

sufficient duration, an analogous stabilized condition for general combinatorial systems seems quite unlikely. In particular, fluctuations away from an optimal solution are apt to be significant even in the "combinatorial vicinity" of such a solution. If this supposition is true, simulated annealing would undoubtedly benefit by a modification that does not rely on a strong stabilizing effect over time. (Under assumptions which imply an appropriate stabilizing effect exists, simulated annealing and "Markov generalizations" can be shown to yield an optimal solution with probability 1 after a sufficient number of iterations at each level [40]. The implicit required effort has not, however, been shown better than that of complete enumeration.) The approach of [36], for example, acknowledges the shakiness of the stability assumption by introducing a final phase to see if any improved solutions appear during some additional time increment after reaching the supposedly stable area.

There are also other reasons, however, to maintain a healthy skepticism about the appropriateness of simulated annealing for discrete optimization—a fact which makes the approach all the more interesting if the claims of its effectiveness are widely substantiated. In spite of its heralded relevance to artificial intelligence, there seems little about the method that is "intelligent" in the usual sense. Reflection on the way a human might proceed, for example, leads to a somewhat different organization. It is reasonable to postulate that human behavior resembles the simulated annealing process in one respect: a human may take "nongoal directed" moves with greater probability at greater distances from a perceived destination. (Similarly, an animal on the hunt, when confronted with a relatively weak scent, is inclined to wander to pick up additional scents rather than to zero in immediately on a particular target.) Yet the human fashion of converging upon a target is to proceed not so much by continuity as by thresholds. Upon reaching a destination that provides a potential "home base" (local optimum), a human maintains a certain threshold—not a progressively vanishing probability—for wandering in the vicinity of that base. Consequently, a higher chance is maintained of intersecting a path that leads in a new improving direction.

Moreover, if time passes and no improvement is encountered, the human threshold for wandering is likely to be increased, the reverse of what happens to the probability of accepting a nonimproving move in simulated annealing over time. On the chance that humans may be better equipped for dealing with combinatorial complexity than particles wandering about in a material, it may be worth investigating whether an "adaptive threshold" strategy would prove a useful alternative to the strategy of simulated annealing. At this time the question is open.

3. LEARNING STRATEGIES

The ability to learn is generally regarded as a fundamental attribute of intelligence. Learning studies that were applied to discrete optimization problems as long ago as the 1960s uncovered interesting conclusions, not widely known, which suggest fruitful directions for investigation.

Scheduling

An early study devoted to the job shop scheduling problem [41] examined two types of learning schemes. In the first, probabilities were assigned to selecting various local decision rules at different stages of the solution process. The approach then generated multiple solutions to the same problem, sampling from the decision rules on the basis of the given probabilities. By comparing the quality of solutions obtained, the procedure "learned" how to improve its own performance—which it accomplished simply by revising the probabilities to favor rules that were selected more often in obtaining the better solutions. Success of the probabilistic learning scheme was verified by its ability to find solutions superior to those obtained either by applying its component decision rules in isolation or by simple random sampling among these rules.

The second type of learning strategy was prompted by a "bow and arrow" analogy which suggested possible limitations to the probabilistic learning scheme. In this strategy, evaluation criteria underlying local decision rules were viewed analogous to criteria that might be taken into account in archery—e.g. wind direction, distance from the target, draw of the bow and so forth. To be effective in archery, all such relevant criteria would be taken into account simultaneously, rather than in a piecemeal "probabilistic" fashion that accommodated only a single criterion at any given time. (The latter approach would likely assure that an archer would never succeed in hitting the target!) Correspondingly, following this principle, a composite decision rule was sought for the job shop

scheduling problem that would *simultaneously* accommodate the various criteria implicitly contained in standard local decision rules. This was achieved by a parametric method for integrating the different component rules, and the strategy undertook to learn good parameter settings by the approach of systematically varying their values over successive trials, keeping track of the combinations and directions that worked best. The composite strategy vindicated its premise by matching or improving on the best known solutions for each of the test problems of the study. (More recent support for using composite frameworks has been provided by an approach of [35] that allows different criteria to "vote" on potential moves.)

Traveling salesman problems

A somewhat different type of learning strategy has been applied to the traveling salesman problem in [42]. To generate trial solutions that provide the raw material for learning, the approach used the standard 2-OPT heuristic for traveling salesman problems [43], but other methods to produce trial solutions could be used as well (e.g. [44–47]). The approach was based on an adaptation of the familiar notion that "good things share common features". More specifically, it was supposed that at least one exceedingly high quality solution would contain subsequences of nodes in common with some subset of the best heuristic trial solutions (obtained by applying the heuristic for different random starting points). As a test of this notion the procedure was designed to select the three best trial solutions obtained in an initial series of runs, and then to identify their intersecting subsequences. These subsequences were compelled to lie in each starting solution generated thereafter, and to remain intact throughout a given run until no improving moves were otherwise available. The resulting exploitation of "learned commonalities" produced solutions superior to those obtained by an equal number of runs of the basic heuristic on its own.

An expanded framework for this type of learning is embodied in the characterization of "strongly determined" and "consistent" variables in [48]. A variable may be called strongly determined, relative to some form of sensitivity or "post solution" analysis, if an assigned value (or assigned bounds) cannot be changed except by producing a significant deterioration in solution quality. By extension, a variable may be called consistent if it is strongly determined at a particular value or within a narrow range in some collection of good trial solutions. (For the type of constructive procedure that makes projections from incomplete assignments, as by solving problem relaxations to obtain trial values for unassigned variables, the "good" solutions may not satisfy all conditions of feasibility at early stages of construction.)

There is clearly a healthy latitude in the possible operational specification of what makes up a consistent variable. Experimentation and context determine whether a variable should be regarded as more consistent if it strongly receives a particular value in three out of five cases, or if it less strongly receives a value in four out of five cases. Segregation of types of solutions may accommodate the possibility that a variable may be strongly determined at one value in some solutions, and strongly determined at another value in other solutions. Consideration of clusterings and interactions among subsets of variables provides logical ways of extending the concept. (It is of course possible to bypass reference to strongly determined variables and simply define consistent variables in terms of the statistical frequency of particular value assignments in selected solutions.)

The motivation for identifying consistent variables, by extension of the rationale underlying the traveling salesman application of [42], is based on three conjectures: (1) a variable that is highly consistent over a subset of good solutions is very likely to receive its preferred value—or to lie within its preferred range—in optimal and near-optimal solutions; (2) once some variables are assigned specific values or constrained to narrow ranges, other variables that seemed not particularly consistent will now become a good deal more so; (3) the operation of imposing narrow restrictions on selected variables will yield increasingly reliable measures of the relative consistency of remaining variables, given the imposed restrictions. These ideas lay a foundation for constructing more ambitious learning procedures both in the traveling salesman setting and in broader discrete optimization contexts.

An IP goal programming application

Still another learning approach invites consideration, particularly for the fact that it has proved uniformly *unsuccessful* in a number of variations. Experience with the approach provides a useful

object lesson about the sensitivity of a learning method to the right blend of components. The context of this application is an integer goal programming problem involving the minimization of deviations from a set of targets. The problem may be paraphrased as that of assigning a collection of weighted objects to boxes so each box receives a total weight that matches its target weight as closely as possible. (An elaboration of this problem is considered in the section on tabu search.)

A learning procedure for this problem was superimposed on a class of simple constructive heuristics of the following form:

Step 1. Choose the next object to be assigned to a box (e.g. by preordering the objects).
Step 2. Choose a box to receive the selected object whose target weight minus its current weight is maximum. (Some of these values may be negative.)

The quality of the solution obtained at the completion of such a procedure is indicated by the sum of the absolute values of the discrepancies between the final assigned weights and the target weights. A natural learning process, therefore, is to take these discrepancies into account on a renewed application of the procedure. A box that receives too much weight should, for example, have its target correspondingly adjusted to appear smaller than it actually is, inducing the method to compensate by assigning the box less weight than before. A straightforward application of this principle was embedded in a learning approach that adjusted each target by exactly the amount of the discrepancy produced on the previous pass, observing that if the method were to compensate by this amount, a perfect solution would result.

While seemingly plausible, this approach did not improve the quality of solutions obtained. More careful reflection suggested that the adjustment on a given pass should not depend simply on the assignment of the preceding pass, but also on the previous targets that led to this assignment. Upon altering the procedure to take this observation into account, however, the results were again unimpressive.

Continued search for refinement led to the supposition that as the boxes were more nearly "filled", the adjusted targets should be phased out and replaced by the original targets. (At the extreme, if a box were to be assigned just one more item, then its original target would appropriately become the basis for deciding the weight of that item.) Thus, the adjustments determined by learning were further modified by introducing a decay factor, which permitted the influence of these adjustments to diminish as the weight (and expected number of elements) of a box approached the original target value. This strategy, too, did not yield improved solutions.

Finally, it was hypothesized that the decay should not be a continuous function of the current weight or number of items in a box, but rather governed by a threshold, such that the full adjustment should apply before reaching this threshold weight and the decay factor should only be allowed to operate thereafter. Upon integrating this last consideration with the others, the learning strategy abruptly succeeded and a marked improvement in solution quality resulted.

The moral of this anecdote is that learning is not a process with a single form, nor is the first plausible form necessarily effective. More attention needs to be given to components of learning strategies that prove useful in specific contexts in order for the value of such strategies to be realized in practical application.

A learning experiment

Before leaving the subject of learning, there is a final area that deserves consideration. From one perspective, learning may be viewed as a form of hindsight analysis. Yet the current studies in discrete optimization disregard one of the most conspicuous forms of information on which hindsight may be based: a pre-determined optimal solution (or collection of alternative optima). From another perspective, learning may be viewed as pattern recognition with a time dimension. Integer programming provides a natural tool for modeling problems in pattern recognition, and current studies often also overlook the potential to employ IP in this way as a component of the learning process.

These two elements may be conveniently elaborated in the context of branch and bound solution strategies. Intelligent learning within the branch and bound setting may draw on several sources of information, such as

(i) alternative types of branching penalties,

(ii) alternative means of evaluating mutually exclusive branches,
(iii) easily specified logical interdependencies (as in multiple choice and variable upper bound problems),
(iv) historical data of the search,
(v) various methods for generating trial "completions" of the current solution.

Given the enormous wealth of potential information to account for, and the many ways it can be integrated, today's research customarily settles on hypothesizing the relevance of a few types of information and a handful of decision rules, thus enabling comparisons of only a small number of options. But there is nothing to guide the creation of these options except relatively blind intuition—i.e. there is no reference to sophisticated models for combining available information and, equally importantly, there is no reference to knowledge of a solution, or set of solutions, one is trying to reach.

The following alternative scenario invites consideration. To begin, a massive solution effort is applied to a representative sample of a given class of problems, with the goal of obtaining optimal or exceptionally high quality solutions. To achieve this goal, time and resources are expended in amounts beyond those normally considered appropriate in a practical, day-to-day operating environment. (For the sake of economy, the effort could begin by testing smaller problems, and checking for the possibility of extrapolating the results to larger ones. Alternatively, attention might first be given to settings where optimal solutions can be constructed in advance.)

The solutions determined in the initial stage of investigation then become a source of information for the next. At this point, discrete optimization is introduced to aid in identifying connections that may enhance the solution process—employing a bootstrapping operation in which a mixed IP pattern recognition model determines which evaluative factors are most relevant, and the weights and conditional relationships (including thresholds) that provide effective composite rules. (An AI perspective would not frown on introducing a human "pattern recognizer" into the loop to facilitate the process. Although perhaps not a mainstream tradition, OR practitioners have for years similarly found it useful to draw on human intervention to enhance the performance of their models and algorithms [13, 45, 49].

The first objective of the resulting rules is simple: to identify at least one of the currently available branches that will lead to an optimal solution. (Such branches are known since an optimal solution has already been determined.) The creation and testing of the model would occur by progressing through a sequence of solution steps in which the evaluation of the model is challenged to prescribe an appropriate next move at each juncture.

There is, however, a second level consideration. Although a variety of branches may lead to an optimal solution, some will typically lead to solution states that are exceedingly poor for differentiating subsequent good and bad alternatives. In other words, the amount of new information gained by branching to such a state may be very slight. (Generally speaking, greater marginal information is gained by branches that more radically change the state of the system.) Thus, the time path of generating and testing the model will be crucial to its success.

A number of strategic elements that are relevant to accommodating this consideration are suggested by proposals of the last dozen years. For example, techniques and criteria for identifying influential variables, shrinking the branch and bound tree, performing "branch reversals", imposing branches by degrees, employing pseudo costs and bounds, and so forth, have already shown promise [50–56], and would undoubtedly yield further benefits by means of a "hindsight" learning study to identify their most useful interconnections. (It is also interesting to note that some of these proposals provide concepts that can be used to extend the parallel AI search strategies such as A^*, Z^*, etc. [57–62], which in turn broaden some elements of the branch and bound perspective traditionally employed in OR.)

Implementation of a learning study of this type would clearly be a challenging and ambitious undertaking. On the other hand, such an effort would help to answer many questions that current research has not thoroughly resolved: (1) Which information is most important for incorporation into decision rules? (2) What parameters and thresholds should be used to integrate this information? (3) What gain may be achieved by allowing the inclusion of information that is more expensive to generate or store? (4) What is the influence of the conditional time path of the solution (and what parameters help identify moves that lead to more informative states)? (5) To what extent does the

discovery that a previous move should be "redecided" play a role in the process? (6) How are the answers to these questions affected by the class of problems examined? (7) How are outcomes influenced by the choice of a "representative" sample (e.g. should the class be further divided into subclasses, and do rules for smaller problems extrapolate to larger ones)?

Until such an investigation is undertaken, significant gaps will remain in our knowledge of the relative value and limitations of alternative information for solving combinatorial problems. Carefully mapped out "learning plans" of this sort should provide an important direction for future research.

4. INDUCED DECOMPOSITION

A great deal has been written about the relevance of decomposition to problem solving, ranging from the classical "decomposition principle" of linear programming [20, 63] to the "near decomposability" theorems of artificial intelligence [64–66]. Though not always viewed as such, relaxation strategies (including the familiar Lagrangean and surrogate constraint approaches) constitute instances of the divide-and-conquer philosophy inherent in the notion of decomposition. The same is true of restriction strategies, including the branching methods of branch and bound.

Decomposition ideas have gained added impetus as improved methods have been developed for certain types of problem structures, particularly those associated with networks and graphs (or matroids). In recent years attention has increasingly been given to how to identify these structures as embedded components within larger systems [67–72], bringing with it a parallel interest in improved procedures for integrating the solutions of subsystems with the solution of systems that encompass them [73–79].

A related (sporadically recurring) theme in discrete optimization is now coming into sharpened focus, and will quite probably soon rival the identification of embedded structures as a major avenue for enhancing IP problem solving capabilities. This theme, which may be called *induced decomposition*, is characterized by the goal of *creating* structure in discrete optimization problems rather than simply *finding* it. Early examples are the proposals for generating inequalities involving partial and nested sums of variables—providing exploitable information which, although not explicit, can readily be inferred from other problem constraints [80, 81]. Additional examples are the procedures for generating unit coefficient cuts as covering and matching inequalities [9, 12, 82]. (Such inequalities are often used, however, to augment the original system, and therefore constrain it more tightly, rather than as part of a decomposition strategy (see, for example, [10, 13]).

Other forms of induced decomposition have arisen with the techniques used to establish the equivalence of zero–one IP problems to zero–one generalized networks [83, 84]. These techniques create a generalized network structure by splitting the original variables into auxiliary variables which are linked by equal flow side constraints. (The side constraints are satisfied automatically on imposing zero–one conditions, which yields the equivalence of the original system and the resulting generalized network system.) Subsequent research has reinforced the interest in these structures by the development of special procedures for networks with "equal flow" constraints [74–77].

A potentially useful adjunct to the induced decomposition framework has come about quite recently with the formal characterization of the network creation methods in a broader context of *layering strategies* [85]. These strategies make use of the fact that the manner of creating auxiliary variables effectively partitions the problem into layers, and this in turn gives rise to a wealth of alternatives for decomposing the system into exploitable subsystems. For example, each different way of subdividing a set of linear inequalities into pairs creates a different decomposition into simple generalized network components, and the system incorporating any collection of such pairs (or larger component groupings) can be conveniently exploited by parallel processing within the layering framework. Lagrangean relaxation also has a particularly strong form in this setting [85, 86]. Instead of simply identifying embedded network structure, layering strategies afford the opportunity to choose which embedded structure to extract, and allow the creation of different structures by implicit duplication of constraints. Finally, the special form of the linking conditions produced by these strategies gives rise to a highly natural type of relaxation/restriction approach as a means for exploiting the induced decomposition.

As the importance of creating (or "engineering") embedded structure becomes more widely recognized, additional forms of induced decomposition may be expected to emerge. Both theoretically and empirically, the area is still in its infancy, and offers considerable promise as a framework for blending AI and OR perspectives to enhance the solution of complex problems.

5. TABU SEARCH

Tabu search may be viewed as a "meta-heuristic" superimposed on another heuristic. The approach undertakes to transcend local optimality by a strategy of forbidding (or, more broadly, penalizing) certain moves. The purpose of classing a move forbidden—i.e. "tabu"—is chiefly to prevent cycling. In view of the mechanism adopted for this purpose, the approach might alternatively be called "weak inhibition" search, for the moves it holds tabu are generally a small fraction of those available, and a move loses its tabu status to become once again accessible after a relatively short time. (In this respect the method may be contrasted to branch and bound, which likewise forbids certain moves to prevent cycling, but in a more rigid fashion—a form of "strong inhibition" search.)

Tabu search originated as a device for implementing the oscillating assignment strategy of [48], and has constituted a primary embodiment of that strategy in applications. The close practical alliance of these procedures motivates a perspective (adopted here) in which they are regarded as synonymous. However, the special features and rationale from which tabu search derives its name have not been clearly formulated in the literature, and the effectiveness of the method suggests the appropriateness of mending this gap. (Recent solution statistics on the method's performance are indicated subsequently.)

From an AI point of view, tabu search deviates to an extent from what might be expected of intelligent human behavior. Humans are often hypothesized to operate according to some sort of random (probabilistic) element which promotes a certain level of "inconsistency". The resulting tendency to deviate from a charted course, sometimes regretted as a source of error, can also prove a source of gain. Such a mechanism has been postulated to be useful, even fundamental, to human ingenuity.

Tabu search, on the other hand, operates without reference to randomization, as though to suggest random variation may be of value only if one is innately disposed to settle for an inferior solution, and is unequipped to find something better except through the providence of accident. (The success of tabu search is not taken to be a comment on the merit of such a perspective!)

More particularly, tabu search proceeds according to the supposition that there is no value in choosing a poor move, by accident or design, except for the purpose of avoiding a path already examined. Subject to this exception, the method seeks a best possible move at each step. (From a pragmatic standpoint, "best" may be defined relative to an abbreviated sample, or aspiration level search, as where a large number of candidate moves renders complete evaluation unattractive.)

By this orientation, the procedure initially heads directly to a local optimum. Local optimality, however, is not a condition that leads to disruption of the search process (as by immediately invoking a "restart" or "shakeup" response), since the ability to identify a best available move remains. To avoid retracing a path previously taken, the procedure records information about moves recently made, employing one or more *tabu lists*. The function of such lists is not to prevent a move from being repeated, but to prevent it from being reversed, and the prohibition against reversal is conditional rather than absolute (employing criteria subsequently to be described).

In its most straightforward form, the tabu list construction records the most recent m moves of the method, where m is a parameter. Moves are subdivided into types and differentiated by direction, and a separate tabu list is maintained for each of these type/direction classifications. For many problems, a single type of move is appropriate and the number of directions is just two. To illustrate, consider an integer programming problem in which a natural move is to assign a variable a value (or bound) adjacent to a current trial value. The directions for the move might then be classified as "up" and "down". Hence one tabu list would be created for the "up" moves and another for the "down" moves. If the variable x_4 is on the "up" tabu list for having moved from 3 to 4, for example, then sufficient memory is stored to prevent x_4 from being moved from 4 back to 3 (unless it satisfies the conditional criteria allowing it to escape this restriction). Note that x_4 may appear more than once on the list, as where a previous move from 2 to 3 may also be recorded. Likewise, x_4 may simultaneously

be on the "down" tabu list for having moved from 6 to 5. (That is, the method for generating trial solutions may possibly not imply that x_4 must first be "moved" to 6 in order to receive that value. This is evident in the common scheme where a move may consist of imposing a bound on a variable and then generating a trail solution by linear programming. Imposing a bound on x_1 may jump the current trial value of x_4 past a number of those values that would be reached by adjacent moves.)

In context of a plant location problem, to provide another illustration, a move may be directed *from* one location *to* another. In this case, each move has only one direction instead of two. (A move does not start at a "to" location and then move to a "from" location.) Consequently, a tabu list is established to prevent these single direction moves from being reversed. If a plant is on the list for having been moved from Location 2 to Location 3, then it is prohibited (conditionally) from moving from Location 3 to Location 2.

These two examples are taken from real world application of the method, the first in solving a nonlinear covering problem [48], and the second in solving an architectural design problem involving the location of component entities in group clusters [87]. Other commonly encountered classes of problems (including the one for which computational results are given subsequently) involve a move where one element, or set of elements, is exchanged for another. For example, the standard move of the 2-OPT traveling salesman heuristic deletes two nonadjacent edges of the tour and then adds the two uniquely determined edges that result in a different tour. In settings involving such exchanges, it is natural to record only part of the information about a move in order to save memory and time in processing the tabu lists. A partial record for the traveling salesman problem, for example, may identify just one of the two deleted edges, with the effect of preventing this edge from being added back to the tour until its residence on the tabu list expires. Similarly, in simpler settings, a variable may be prevented from moving to a particular value, or a plant from moving to a particular location, without regard for the origin of the move.

Tabu list processing

The tabu lists are managed by recording moves in the order in which they are made. Each time a new element is added to the "end" of a list, the oldest element on the list is dropped from the "beginning". (The list can initially be filled with dummy elements to avoid checking whether it has been filled.) Thus, tabu lists are effectively managed as circular lists, whose starting point progresses around the circle as the new first element replaces the last old one. In some discrete optimization settings, it is possible to identify different moves as equivalent, and it is important that the tabu status of a particular move extends to all of its equivalent counterparts. Such settings also often allow "null" moves that are not in the domain considered by improvement heuristics, but which must be explicitly proscribed in tabu search. For example, a zero–one knapsack problem may contain two variables that are indistinguishable except for indexing. A move that swaps one of these for the other in the current trial solution constitutes a null move. (Two moves that swap a different one of these variables with the same third variable provide an instance of equivalent moves.)

The *conditional* manner in which a move is prevented from being reversed is a key aspect of the procedure. For the purpose of discussion, suppose that the reverse move, the one rightly to be regarded tabu, is the one recorded*. At the simplest level, tabus status can be enforced by means of a penalty which is preemptively large for every move on a tabu list, so that none of these moves will be chosen so long as a nontabu alternative exists. The penalties are allowed to decay as a function of tabu list order, so that if no move is available (other than a tabu move) an older tabu move will be chosen in preference to a more recent one.

The preemptive penalty approach, however, overlooks the chief purpose of the tabu lists, which is to prevent cycling while affording flexibility. (There is more in this than meets the eye, for it would seem to imply that the smallest tabu list which prevents cycling would be best. Yet experience suggests a slightly larger list is often better, as subsequently noted.)

Two approaches faithful to this purpose, and which have produced solutions of better quality than the strictly preemptive scheme, are as follows. The first and simplest approach allows a move to

*Organizing the approach to prevent repetition rather than reversal of a move has not proved a competitive alternative.

override its tabu status provided the resulting trial solution, or a bound relating to such a solution, improves on the best value so far obtained.

The second approach imposes a less stringent requirement for overriding tabu status, but entails additional record keeping. To implement the approach, an aspiration list $A(z)$ is maintained for each value z of the objective function (over a relevant range). Translation and scaling, and mapping fractional values into adjacent integers, are used to insure that z takes integer values from 1 to U. (By this organization, $z = 0$ is assumed to provide a lower bound on the optimum (minimum) value of z, and all values of $z > U$ are treated as though $z = U$.)

$A(z)$ may be interpreted as the *aspiration level of the objective function value next to be reached when the current value is z*. Thus, initially, $A(z)$ is set to $z - 1$ for all z, indicating the aspiration of identifying a move that will improve z at least to the value $z - 1$. (Note, if fractional z values are rounded up, then $z - 1$ represents the "nearest neighbor" improvement for the true z value.) In general, the strategy of the aspiration list is to permit a move to override its tabu status if it succeeds in improving z to the value $A(z)$. Execution of this strategy rests on the issue of how to update $A(z)$.

Let z' and z'' denote values of z obtained on two successive iterations. The goal for future moves that begin at z' is to reach a value better than z'', hence the value $A(z')$ is updated to $z'' - 1$ if $z'' \leqslant A(z')$, which is equivalent to setting $A(z') = \text{Min}[A(z'), z'' - 1]$. Moreover, since the move from z' to z'' customarily implies the existence of a reverse move from z'' to z', an analogous update is employed in this case to yield $A(z'') = \text{Min}[A(z''), z' - 1]$. [$A(z'')$ may be updated in a different manner if the value of z represents an objective function bound rather than an attained value.]

Thus, by the foregoing updates, as long as a move improves the current value of z to $A(z)$ or better, the system achieves a transition that has not previously occurred. Consequently, allowing a move to override its tabu status under this condition is compatible with the goal of avoiding cycling.

A variation is to let the argument of the aspiration list be a parameter other than the objective value. Denoting this parameter by y, the value $A(y)$ may then represent the desired amount of *improvement* in the objective function when y is an observed property of the move. For example, if y represents the length of the shorter of the deleted edges in the 2-OPT traveling salesman heuristic, then initially $A(y) = 1$ for all y (which compels an improvement of at least 1). Thereafter $A(y)$ may be updated by the formula $A(y') = \text{Max}[A(y'), z' - z'' + 1]$ and $A(y'') = \text{Max}[A(y''), z'' - z' + 1]$, where y' is the length of the shorter edge deleted and y'' is the length of the shorter edge added (capable of being deleted in a reverse move), in the transition from z' to z''. Similar observations may of course be applied to other approaches such as the 3-OPT heuristic.

Tabu list sizes

It may be noted that the creation of more than one tabu list can be managed by means of a single list, though the method may operate in a different way if the elements of each list are processed by the same rule (whether in a combined list or separate lists). In particular, the past is forgotten at a constant rate in the case of a single list. In the case of two or more tabu lists, however, one list may be more active than another through a certain sequence of moves (e.g. where "up" moves occur more frequently than "down" moves).

The use of separate lists for different types and directions of moves appears to be advantageous on the basis of empirical finding. Two general conclusions concerning the use of separate tabu lists, which interestingly appear to be independent of the application, have emerged from experiment: (1) each tabu list should have the same value of m (i.e. the same size), though small variations seem relatively unimportant; (2) the best value for m is approx. 7 (though all values from 5 to 9, and up to 12 in the case of [35], appear to work well)*.

One is prompted to note the similarity between the empirically best values for tabu list size and the number of "chunks" that humans typically carry in short term memory. It is intriguing to speculate whether this similarity implies that empirical tests have rediscovered a principle already found by nature—i.e. that such a parameter value gives an effective way to handle combinatorial problems by means of a very simple memory mechanism. (The analogy is imperfect, of course, unless humans can be shown to use short term memory in the manner of a tabu list.)

*Tabu lists containing only a single element can sometimes yield solutions significantly better than a local optimum. (See the knapsack example in [48], which also illustrates the interplay of feasibility and optimality considerations.)

Experimentation indeed suggests there is some combinatorial significance to values of m in the range indicated. In the application cited at the conclusion of this section, values of m up to 4 were found to admit the cycling phenomenon in which a particular sequence of solutions would repeat indefinitely. (For $m = 4$, cycle lengths were observed typically to range from 14 to 30 moves.) Abruptly at $m = 5$ all evidences of the cycling phenomenon vanished. This behavior of course must depend to an extent on the problem setting and the handling of the tabu lists. Unless all moves equivalent to a given tabu move are simultaneously classed as tabu, for example, the outcome would not be expected.

Additional tabu parameters and list management rules

In some settings there is value to maintaining an additional form of tabu list to prevent a selected parameter from reversing its value until a certain level is reached. For example, in scheduling employees to duty rosters [35], there is often interest not only in the least cost schedule, but in the least cost schedule for some number of employees close to the optimum number. (The source of such interest is the fact that reasonable trade-offs can be used to influence hiring and lay-off policies under alternative scenarios.) Thus another tabu parameter in this setting periodically compels an increase or decrease in the number of employees scheduled, creating a search pattern that resembles a series of pendulum swings on both sides of the number currently associated with a local optimum. This form of tabu list control is useful even where trade-offs are not of concern: there may be only one acceptable number of employees, but controlled oscillations about that number can aid in finding improved configurations. Such a behavior, which in general induces oscillations around "ideal" values of various parameters (including the objective function value and measures of feasibility) has motivated the earlier use of the "oscillating assignment" label for this strategy.

In some settings where both feasibility and optimality considerations are relevant, a simple model which is a variant of another technique from [35] may be useful. The approach is motivated by the observation that although infeasibility can always be handled by a penalty function, there appears to be a qualitative difference between feasibility and optimality for some types of problems, and this suggests in turn that the form of the penalty function should change according to the current "degrees" of these two components. To illustrate, suppose an evaluation function $E(x)$ of a current (partial or complete) trial solution x is expressed in the form $E(x) = aF(x) + bO(x)$, where $F(x)$ denotes a feasibility measure and $O(x)$ represents an optimality measure. [$F(x)$ might further be subdivided into components to distinguish different types of feasibility, such as non-negativity, integer feasibility, etc.]. The purpose of segregating $F(x)$ and $O(x)$ in this representation is to allow the "qualitative differentiation" of feasibility and optimality to be implemented by adjusting a and b as a dynamic feature of the search process.

The nature of this adjustment can be illustrated by reference to the familiar "outside in" and "inside out" heuristic procedures. An "outside in" approach, as in [88], begins at a distance from the feasible region and changes the values of variables (as a rule, unidirectionally) until feasibility is attained. By contrast, an "inside out" approach, as in [89], starts from inside the feasible region (after an initialization step, if required) and changes values of variables until no improving changes remain except that would force an exit from this region.

Manipulating the a and b parameters, and using standard tabu lists to avoid cycling, provides a convenient mechanism for progressing beyond the normal domain of these two procedures—giving a flexible method to approach and then cross beyond the feasible boundaries from either side. The parameters may for example be used to govern the depth of penetration beyond (or into) the feasible region before instituting a reverse trajectory back toward the periphery. In the spirit of the AI perspective, this manipulation of weights offers an opportunity to employ an adaptive aspiration strategy, narrowing the parameter ranges about values that uncover good solutions (in the regions where this occurs), and then widening the ranges after a duration in which no improved solutions are located. Such a dynamic manipulation of a and b enables relatively simple functions $F(x)$ and $O(x)$ to behave as much more subtle functions, which are not readily expressible in closed mathematical form.

The differentiation of feasibility and optimality can also be applied to problems where feasibility and optimality have been absorbed into a single measure by the problem formulation, and hence supposedly offer no basis for separate consideration. In the goal programming model for the constrained clustering problem of [87], for example, it was found desirable to accentuate deviation

penalties in regions quite close to the goals (creating a concave relationship!), in spite of the fact that this distorted the influence of alternative moves on the true objective function. From a general perspective, the feasibility/optimality division may of course be subdivided further, as to differentiate integer feasibility from inequality conditions or to isolate the effects of different objective function measures. Alternatively, the division may be replaced by divisions representing a *procedural orientation*, as in choosing among constructive, destructive and swapping moves (or, more generally, various classes of state change and state preservation moves). Exploration of effective ways to manipulate the parameters in these settings might usefully be carried out in the framework of the "Learning Experiment" suggested in Section 3.

Finally, it should be remarked that it is possible, by more elaborate record keeping and list management rules, to rigorously assure that every possible solution will ultimately be examined, without cycling, while retaining the ability to choose any nonproscribed move (without reference to a branch and bound type of tree structure). Such a method was tested for the nonlinear covering problem of [48], where the moves consisted of increasing and decreasing values of integer variables within their bounds. The outcome disclosed that the more elaborate list management scheme was less effective than the simpler memory mechanism of maintaining the tabu lists as circular data structures. Such an outcome might be expected by an orientation that favors more flexible search forms. Yet there remain many settings to which tabu search has not yet been applied, and the potential relevance of such variants should not prematurely be discounted.

Computational outcomes for tabu search

Previous indications of the effectiveness of tabu search are given in [35, 48, 87], but these results do not isolate the effects of tabu search from other tandem strategies (such as decomposition and decision rule integration). By focusing on applications rather than methodologies, previous studies have also not disclosed the relationship between local optima obtained by a specified heuristic and the solutions obtained by tabu search when employing the same class of moves. We report such results here for a set of problems from a real world setting involving the optimization of channel loads in a computer processing environment. The form of these problems has been described in Section 3 in the guise of assigning weighted objects to boxes, where the goal is to minimize the total deviation of assigned weights from target weights. In the setting of channel load optimization, the problems include additional constraints limiting the number of objects (disk packs) that each box (channel) can receive.

The type of move selected to be the common denominator of the basic improvement heuristic and the tabu search procedure consisted simply of exchanging an object in one box with an object in another. By means of zero-weight "pseudo objects" this move definition was allowed to include reference to half-swaps that transferred an object from one box to another without a reverse transfer. The selection of moves was restricted to these half-swaps so long as a bound on the number of objects in a box was violated, requiring that the current choice ameliorate such a bound violation. Similarly, no moves that created a bound violation were permitted in the version of the heuristic employed.

To embed this characterization of moves in the tabu search framework, a "null move" was identified to be one that exchanged objects of equal weights. "Equivalent moves" were defined relative to the same underlying principle, which in a strict implementation would identify two moves as equivalent if they transferred the same respective weights between two specified boxes. (Equivalence would not be based on the difference of the weights since, while equal differences create the same respective changes in the weights of the two boxes, they may also create different compositions of the boxes.) These considerations were simplified, however, by storing information only for one of the boxes involved in an exchange, causing a tighter tabu restriction but avoiding reference to ordered pairs. Thus, equivalent moves were identified by the simple expedient of flagging all weights equal to the tabu weight in the selected box, and allowing only unflagged weights to be candidates for exchanges, except where the aspiration list prescribed otherwise.

Sixteen different test problems were supplied by the Division of Advanced Data Technology of the U.S. Bureau of Land Management as the basis for a preliminary study. Formulated as zero–one mixed integer programs, these problems contained 144 zero–one variables, 8 continuous variables and 40 constraints. (In spite of their modest size, they were not trivial, as will shortly be seen.) For the preliminary study, the basic heuristic and its tabu search elaboration were applied to nine starting

solutions: the three best, three worst and three "middle" solutions generated by the learning approach of Section 3 without consideration for bound violations. (The worst solutions were generally those used to seed the learning process, or obtained in early iterations of the process. "Duplicates" with the same sum of deviations from targets were weeded out before selecting these solutions as a starting point for subsequent search.) It should be noted that the quality of the starting solutions relative to the learning procedure, which disregarded bounds on the numbers of items to be assigned to a box, does not translate directly into a measure of quality for the more general problems. Summary statistics of the nine solution trials for each test problem appear in Table 1. Tabu list sizes were based on $m = 7$ and values in the table are reported to the nearest one-tenth. Table 1 shows the dramatic difference between local optima found by the iterative improvement heuristic and solutions obtained by tabu search. The overall average for the local optima exceeds that for tabu search by a factor of nearly 15 to 1. (The average of the ratios for the best solutions of each is approx. 6 to 1.) The *best* of the local optima proved superior to the *worst* of the tabu search solutions for only two problems (No. 3 and No. 5). It is also interesting to note the significantly greater stability of tabu search, whose solution ranges were much tighter than those for the local optima.

An additional result of interest from the preliminary study provides a comparison of tabu search to the commercial mixed IP code MPS. Because the quality of the final solution obtained by tabu search seemed little affected by the starting solution, the initialization of the method was simplified by giving it the best and the worst seed solutions for the learning method, eliminating the preliminary learning procedure entirely. (Since this procedure disregarded the bounds on the numbers of objects in the boxes, no correspondence between specific starting solutions and final solutions had been observed.) A test was then conducted on an additional problem selected by the Bureau of Land Management from the same class. Using a Honeywell DPS-8 computer, the tabu search procedure took 4 CPU seconds and obtained a solution with an objective function value of 1.0. The MPS code, by contrast, was taken off the machine after 600 CPU seconds, having obtained a solution with an objective function value of 21.0.

6. CONCLUSIONS AND IMPLICATIONS

Within the limited space of this article it has not been possible to cover all of the areas of discrete optimization that are likely to benefit from a blend of OR and AI perspectives. Nor has it been possible to give due credit to all the research that is likely to significantly influence the shape of things to come.

Notably lacking in this portrayal of important developments ahead has been a detailed look at the forms of models that will go hand-in-hand with determining improved strategies. (The "bootstrapping" mixed IP models mentioned in connection with the hindsight learning strategies, for example, can take a variety of forms, and the identification of the more useful ones will be a worthy

Table Objective function values

Problem	Range		Average	
	Local opt.	Tabu search	Local opt.	Tabu search
				2.3
				2.0
3				1.8
4				1.6
5				2.0
6				2.1
7				2.1
8				2.4
9				2.6
10				2.3
11				1.0
12				1.7
13				2.1
14				2.8
15				1.8
16				2.0
Overall averages:				2.0

contribution.) The variety of OR models [25, 90–95] are already appearing that are challenging the assumptions of classical statistical approaches, and such developments are central to the pattern recognition concerns that naturally attend learning strategies.

There is a circular chain of effect in AI/OR innovations, in which models and methods each influence the other, laying a foundation for mutually reinforcing advances. The phenomenon is evident in each of the areas covered in this paper, but perhaps nowhere more conspicuously than in the induced decomposition strategies, which must rely intimately on the interplay between models and methods to achieve their maximum potential.

While at present there appear to be many constituents of effective heuristic procedures, the time is perhaps approaching where, as in physics, it is appropriate to begin devising GUTs (grand unifying theories) to bind diverse elements together. In the combinatorial problem solving setting, the gains that derive from effective methods now emerging will undoubtedly stimulate the search for common denominators. As this occurs, the largely segregated perspectives of the past will increasingly give way to more unified views, opening dramatic and exciting possibilities for future developments.

REFERENCES

1. R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*. Wiley, New York (1972).
2. E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, New York (1976).
3. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithm and Complexity*. Prentice–Hall, Englewood Cliffs, N.J. (1982).
4. R. T. Rockafellar, *Convex Analysis*. Princeton University Press (1970).
5. H. M. Salkin, *Integer Programming*. Addison–Wesley, Reading, Mass. (1975).
6. H. D. Sherali and C. M. Shetty, *Optimization with Disjunctive Constraints*. Lecture Notes in Economics and Mathematical Systems, Vol. 81. Springer, Berlin (1980).
7. S. Zionts, *Linear and Integer Programming*. Prentice–Hall, Englewood Cliffs, N.J. (1974).
8. E. Balas, Disjunctive programming. In *Discrete Optimation* (Edited by P. L. Hammer, E. L. Johnson and B. H. Korte), pp. 3–52. North-Holland, Amsterdam (1979).
9. G. H. Bradley, P. L. Hammer and L. A. Wolsey, Coefficient reduction for inequalities in 0–1 variables. *Math. Progr.* **7,** 263–282 (1974).
10. H. Crowder, E. Johnson and M. Padberg, Solving large scale 0–1 linear programming problems. *Opns Res.* **31,** 803–934 (1983).
11. M. Guignard and K. Spielberg, Propagation, penalty improvement and use of logical inequalities. *Math. Opns Res.* **25,** 157–171 (1977).
12. M. W. Padberg, Covering, packing and knapsack problems. *Ann. Discr. Math.* **4,** 265–287 (1982).
13. T. J. van Roy and L. A. Wolsey, A software system for mixed integer programming by reformulating automatically using cuts. Talk at session TA7, TIMS XXVI meeting in Copenhagen (1984).
14. L. A. Wolsey, Facets and strong valid inequalities for integer programs. *Opns Res.* **24,** 367–372 (1976).
15. E. Balas, Disjunctive programming and a hierarch of relaxations for discrete optimization problems. Management Research Report No. MSRR-492, Carnegie–Mellon University (1983).
16. M. S. Bazaraa and C. M. Shetty, *Foundations of Optimization*. Springer, Berlin (1976).
17. A. M. Geoffrion, Lagrangean relaxation for integer programming. In *Mathematical Programming, Study 2*. North-Holland, Amsterdam (1974).
18. M. Held, P. Wolfe and H. D. Crowder, Validation of subgradient optimization. *Math. Progr.* **6,** 62–88 (1974).
19. M. H. Karwan and R. L. Rardin, Some relationships between Lagrangian and surrogate duality in integer programming. *Math. Progr.* **18,** 320–334 (1979).
20. L. Lasdon, *Optimization Theory for Large Systems*. Macmillan, New York (1970).
21. M. L. Fisher, R. Jaikumar and J. T. Lester III, A computerized vehicle routing application. Report No. 81-10-03, The Wharton School, University of Pennsylvania (1981).
22. B. Gavish and H. Pirkul, Efficient algorithms for solving multiconstraint zero–one knapsack problems to optimality. *Math. Progr.* **31,** 78–105 (1985).
23. A. M. Geoffrion and G. W. Graves, Multicommidity distribution system design by benders decomposition. *Mgmt Sci.* **20,** 822–844 (1974).
24. F. Glover, D. Klingman, N. Philips and G. T. Ross, Integrating modeling, algorithm design and computational implementation to solve a large scale nonlinear mixed integer problem. CBDA 104, The University of Texas, Austin, Tex. (1984). To be published in *Ann. Opns Res.*
25. J. M. Mulvey and H. P. Crowder, Cluster analysis: an application of Lagrangian relaxation. *Mgmt Sci.* **25,** 329 (1979).
26. E. Balas, Disjunctive programming: cutting-planes from logical conditions. In *Nonlinear Programming* (Edited by O. L. Mangasarian, R. R. Meyer and S. M. Robinson), Vol. 2, pp. 279–312. Academic Press, New York (1975).
27. R. G. Jeroslow, Cutting-plane theory: disjunctive methods. *Ann. Discr. Math.* **I,** 293–330 (1977).
28. R. G. Jeroslow, Representability in mixed integer programming, I: Characterization results. Georgia Tech., Atlanta, Ga (1984).
29. R. G. Jeroslow and J. K. Lowe, Modelling with integer variables. College of Management, Georgia Institute of Technology (1985). To be published in *Math. Progr. Stud.*
30. R. R. Meyer, Integer and mixed-integer programming models: general properties. *J. Opt. Theory Appl.* **16,** 191–206 (1975).
31. F. Glover, J. Hultz and D. Klingman, Improved computer-based planning techniques, part II. *Interfaces* **9,** 12–20 (1979).

32. F. Glover and F. Martinson, A netform system for resource planning in the U.S. Bureau of Land Management. *J. Opl Res. Soc.* **35,** 605–616 (1984).
33. H. J. Greenberg, A new approach to analyze information contained in a model. In *Validation and Assessment Issues of Energy Models* (Edited by S. I. Glass), pp. 517–524. NBS Pub. 569, Washington, D.C. (1979).
34. H. J. Greenberg and J. S. Maybee, *Computer-Assisted Analysis and Model Simplification*. Academic Press, New York (1981).
35. F. Glover and C. McMillan, The general employee scheduling problem: an integration of MS and AI. *Comput. Opns Res.* **13,** 563–573 (1986).
36. D. S. Johnson, Optimization by simulated annealing: a tutorial. AT&T Bell Laboratories. Presented at the *12th International Symposium on Mathematical Programming* (1985).
37. S. Kirkpatrick, X. Gelatt Jr and M. P. Vecchi, Optimization by simulated annealing. *Science* **220,** 671–680 (1983).
38. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines. *J. chem. Phys.* **21,** 1087 (1953).
39. B. L. Golden and C. C. Skiscim, Using simulated annealing to solve routing and location problems. University of Maryland, College of Business and Management, Working Paper Series MS/S 84-001 (1984).
40. F. Romeo and A. Sangiovanni-Vincentelli, Probabilistic hill climbing algorithms: Properties and applications. Department of EECS, University of California, Berkley, Calif. (1985).
41. W. Crowston, F. Glover, G. Thompson and J. Trwick, Probabilistic and parametric learning methods for the job shop scheduling problem. GSIA, Carnegie Institute of Technology (1964).
42. F. Glover, A strategy for traveling salesman problems. GSIA, Carnegie Institute of Technology (1964).
43. S. Lin, Computer solutions to the traveling salesman problem. *Bell Syst. Tech. J.* **44,** 2245–2269 (1965).
44. B. L. Golden and W. R. Stewart, The empirical analysis of TSP heuristics. In *The Traveling Salesman Problem* (Edited by E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan). North-Holland, Amsterdam (1985).
45. P. Krolak, W. Felts and G. Marble, A man-machine approach toward solving the traveling salesman problem. *Commun. ACM* **14,** 327–344 (1971).
46. E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan (Eds), *The Traveling Salesman Problem*. North-Holland, Amsterdam (1985).
47. G. L. Thompson, Approaches to solving large and symmetric traveling salesman problems. Carnegie–Mellon University (1985).
48. F. Glover, Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **8,** 156–166 (1977).
49. C. A. Trauth and R. E. Woolsey, Practical aspects of integer linear programming. Sandia Corporation Monograph, SC-R-66-925 (1966).
50. M. S. Bazaraa and A. N. Elshafei, On the use of fictitious bounds in tree search algorithms. *Mgmt Sci.* **23,** 904–908 (1977).
51. F. Glover, Parametric branch and bound. *Omega* **6,** 145–152 (1978).
52. F. Glover and L. Tangedahl, Dynamic Strategies for branch and bound. *Omega* **4,** 571–576 (1976).
53. M. M. Kostreva, Adaptive estimates improve binary programming. Research Publication GMR-3217, Mathematics Department, General Motors Research Laboratories, Waren, Mich. (1980).
54. R. E. Marsten, User's manual for ZOOM/XMP. Department of Management Information Systems, University of Arizona (1984).
55. R. E. Marsten, T. L. Morin and K. S. Nagarai, Using *a posteriori* bounds to improve the performance of branch-and-bound algorithms. Paper presented at the *12th International Symposium on Mathematical Programming*. MIT, Cambridge, Mass. (1985).
56. G. Mitra, Investigation of some branch-and-bound strategies for the solution of mixed-integer programs. *Math. Progr.* **4,** 155–170 (1973).
57. J. R. Anderson, *The Architecture of Cognition*. Harvard University Press, Cambridge, Mass. (1983).
58. A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence, Volume 1*. Department of Computer Science, Stanford University, HeurisTech Press, Stanford, Calif.; William Kaufmann, Inc., Los Altos, Calif. (1981).
59. N. J. Nilsson, *Principles of Artificial Intelligence*. SRI International, Tioga Publishing Company, Palo Alto, Calif. (1980).
60. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison–Wesley, Reading, Mass. (1984).
61. E. Rich, *Artificial Intelligence*. The University of Texas at Austin, McGraw–Hill, New York (1983).
62. P. H. Winston, *Artificial Intelligence*, 2nd edition. Artificial Intelligence Laboratory, MIT, Addison–Wesley, Reading, Mass. (1984).
63. G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J. (1963).
64. P. J. Courtois, *Decomposability, Queueing and Computer Applications*. Academic Press, New York (1977).
65. H. A. Simon, *The Science of the Artificial*. MIT Press, Cambridge, Mass. (1969).
66. H. A. Simon and A. Ando, Aggregation of variables in dynamic systems. *Econometrica* **29,** 111–138 (1961).
67. R. E. Bixby, Recent algorithms for two versions of graph realization and remarks on applications to linear programming. In *Progress in Combinatorial Optimization* (Edited by W. R. Pulleyblank), pp. 39–67 (1984).
68. R. E. Bixby and W. H. Cunningham, Converting linear programs to network problems. *Math. Opns Res.* **5,** 321–327 (1980).
69. R. E. Bixby and D. Wagner, An almost linear-time algorithm for graph realization. Technical Report 85-2, Department of Mathematical Sciences, Rice University (1985).
70. G. G. Brown, R. D. McBride and R. K. Wood, Extracting embedded generalized networks from linear programming problems. *Math. Progr.* **32,** 11–31 (1985).
71. L. Schrage, On hidden structures in linear programs. In *Computer Assisted Analysis and Model Simplification* (Edited by X. Greenberg and X. Maybee). Academic Press, New York (1981).
72. K. Truemper, How to detect hidden networks and totally-unimodular subsections of linear programs. TIMS/ORSA Joint National Meeting (1983).
73. A. Ali, E. Allen, R. Barr and J. Kennington, Reoptimization procedures for bounded variable primal simplex network algorithms. Technical Report 83-PR-2, Southern Methodist University (1984).
74. I. Ali, J. Kennington and B. Shetty, The equal flow problem. Technical Report 85-OR-1, Southern Methodist University (1985).
75. C. H. Chen and M. Engquist, A primal simplex approach to pure processing networks. Research Report CCS 496, Center for Cybernetic Studies, University of Texas (1985).

76. M. Engquist and C. H. Hen, Computational comparison of two solution procedures for allocation/processing networks. To be published in *Math. Progr. Study*.
77. R. McBride, Solving generalized processing network problems. Working Paper, School of Business, University of California, Los Angeles, Calif. (1982).
78. R. McBride, Solving embedded generalized network problems. *Eur. J. Opns Res.* **21**, 82–92 (1985).
79. R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Wiley, New York (1984).
80. F. Glover, A multiphase-dual algorithm for the zero–one integer programming problem. *Opns Res.* **13**, 879–919 (1965).
81. P. L. Hammer, E. L. Johnson, B. H. Korte and G. L. Nemhauser, *Studies in Integer Programming*. North-Holland, Amsterdam (1977).
82. P. L. Hammer, E. L. Johnson and V. N. Peled, Facets of regular 0–1 polytopes. *Math. Progr.* **8**, 179–206 (1975).
83. F. Glover and J. Mulvey, Equivalence of the 0–1 integer programming problem to discrete generalized and pure networks. *Opns Res.* **28**, 829–835 (1980).
84. V. Srinivasan and G. L. Thompson, Stopped simplex algorithm for the set partitioning problem with transportation-like subproblems. Presented at the *ORSA/TIMS Conference*, Boston, Mass. (1974).
85. F. Glover and D. Klingman, Layering strategies of creating exploitable structure in linear and integer programs. CBDA 199, The University of Texas, Austin, Tex. (1984).
86. M. Guignard-Spielberg, Lagrangean decomposition: an improvement over Lagragean and surrogate duals. Technical Report No. 62, Wharton School, University of Pennsylvania (1984).
87. F. Glover, C. McMillan and B. Novick, Interactive decision software and computer graphics for architectural and space planning. CAAI Report No. 85-3, University of Colorado, Boulder, Colo. (1985). To be published in *Ann. Opns Res.*
88. S. Senju and Y. Toyoda, An approach to linear programming with 0–1 variables. *Mgmt Sci.* **15**, B196–207 (1968).
89. G. A. Kochenberger, B. A. McCarl and F. P. Wyman, A heuristic for general integer programming. *Decis. Sci.* **5**, 36 (1974).
90. T. S. Arthanari and Y. Dodge, *Mathematical Programming in Statistics*. Wiley, New York (1981).
91. S. M. Bajgier and A. V. Hill, An experimental comparison of statistical and linear programming approaches to the discriminant problem. *Decis. Sci.* **13**, 604–618 (1982).
92. N. Freed and F. Glover, Evaluating alternative linear programming approaches to the discriminant problem. MSRS 85-5, University of Colorado. To be published in *Decis. Sci.*
93. J. M. Liittschwager and C. Wang, Integer programming solution of a classification problem. *Mgmt Sci.* **24**, 1515 (1978).
94. M. R. Rao, Cluster analysis and mathematical programming. *J. Am. statist. Ass.* **66**, 622 (1971).
95. H. P. Williams, *Model Building in Mathematical Programming*. Wiley, New York (1978).
96. N. Christofides and S. Eilon, Algorithm for large scale TSP's. *Opns Res. Q.* **23**, 511 (1972).