Binary Unconstrained Quadratic Optimization Problem*

Gary A. Kochenberger, Fred Glover and Haibo Wang

Contents

*This is an updated version of the previous paper given in [38].

G.A. Kochenberger (✉)
School of Business, University of Colorado, Denver, CO, USA

F. Glover
OptTek Systems, Inc, Boulder, CO, USA

H. Wang
College of Business Administration, Texas A&M International University, Laredo, TX, USA

Abstract

In recent years the unconstrained quadratic binary program (UQP) has emerged as a unified framework for modeling and solving a wide variety of combinatorial optimization problems. The unexpected versatility of the UQP model is opening doors to the solution of a diverse array of important and challenging applications. Developments in this evolving area are illustrated by describing its methodology with examples and by reporting substantial computational experience demonstrating the viability and robustness of latest methods for solving the UQP model, showing that they obtain solutions to wide-ranging instances of the model that rival or surpass the best solutions obtained by today's best special-purpose algorithms.

1 Introduction

The unconstrained quadratic binary program (UQP) has a lengthy history as an interesting and challenging combinatorial problem. Simple in its appearance, the model is given by

$$\text{UQP}: \; \textit{Opt} \; \mathrm{x}'\mathrm{Qx}$$

where x is an n-vector of binary variables and Q is an n-by-n symmetric matrix of constants. Published accounts of this model go back at least as far as the 1960s in the work of Hammer and Rudeanu [31] and have applications in such diverse areas as spin glasses [18, 30], machine scheduling [1], the prediction of epileptic seizures [35], solving satisfiability problems [12, 13, 31, 33], and determining maximum cliques [13, 55, 56]. The application potential of UQP is much greater than might be imagined, due to the reformulation possibilities afforded by the use of quadratic infeasibility penalties as an alternative to imposing constraints in an explicit manner. In fact, any linear or quadratic discrete (deterministic) problem with linear constraints in bounded integer variables can *in principle* be recast in the form of UQP via the use of such penalties.

As will be shown, this outcome has more than theoretical significance. A broad range of challenging problems in combinatorial optimization can not only be reexpressed as UQP problems but can be solved in a highly effective manner when expressed this way. The process of reformulating a given combinatorial problem into an instance of UQP is easy to carry out. The common modeling framework that results, coupled with recently reported advances in solution methods for UQP, serve to make the model a viable alternative to more traditional combinatorial optimization models as illustrated in the sections that follow.

1.1 Recasting into the Unified Framework

For certain types of constraints, equivalent quadratic penalty representations are known in advance making it easy to embody the constraints within the UQP objective function. For instance, let x_i and x_j be binary variables and consider the constraint

$$x_i + x_j \leq 1 \tag{1}$$

which precludes setting both variables to one simultaneously. A quadratic infeasibility penalty that imposes the same condition on x_i and x_j is

$$Px_ix_j \tag{2}$$

where P is a large positive scalar. This penalty function is positive when both variables are set to one (i.e., when (1) is violated), and otherwise the function is equal to zero. For a minimization problem then, adding the penalty function to the objective function is an alternative equivalent to imposing the constraint of (1) in the traditional manner.

In the context of transformations involving UQP, a penalty function is said to be a *valid infeasible penalty (VIP)* if it is zero for feasible solutions and otherwise positive. Including quadratic VIPs in the objective function for each constraint in the original model yields a transformed model in the form of UQP. VIPs for several commonly encountered constraints are given below (where x and y are binary variables and P is a large positive scalar):

Note that the penalty term in each case is zero if the associated constraint is satisfied, and otherwise the penalty is positive. These penalties, then, can be directly employed as an alternative to explicitly introducing the original constraints. For other more general constraints, however, VIPs are not known in advance and need to be "discovered." A simple procedure for finding an appropriate VIP for any linear constraint is given in the next section.

1.2 Accommodating General Linear Constraints

To recast a constrained problem in the form of UQP when the VIPs are not known in advance, consider as a starting point the general constrained problem

$$\begin{aligned} &\min\ x_0 = xQx \\ \text{subject to}& \\ &Ax = b,\quad x\ \textit{binary} \end{aligned} \tag{3}$$

This model accommodates both quadratic and linear objective functions since the linear case results when Q is a diagonal matrix (observing that $x_j^2 = x_j$ when x_j is a 0-1 variable). Under the assumption that A and b have integer components, problems with inequality constraints can also be put in this form by representing their bounded slack variables by a binary expansion. These constrained quadratic optimization models are converted into equivalent UQP models by adding a quadratic infeasibility penalty function to the objective function in place of explicitly imposing the constraints $Ax = b$.

Specifically, for a positive scalar P,

$$\begin{aligned} x_0 &= xQx + P\,(Ax - b)^t\,(Ax - b) \\ &= xQx + xDx + c \\ &= x\hat{Q}x + c \end{aligned} \tag{4}$$

where the matrix D and the additive constant c result directly from the matrix multiplication indicated. Dropping the additive constant, the equivalent unconstrained version of the constrained problem becomes

$$\text{UQP} : \min\ x\hat{Q}x,\ x\ \textit{binary} \tag{5}$$

From a theoretical standpoint, a suitable choice of the penalty scalar P can always be chosen so that the optimal solution to UQP is the optimal solution to the original constrained problem. Remarkably, as demonstrated later, it is often easy to find such a suitable value in practice as well.

The preceding general transformation that transforms (3) and (4) into (5) will be called *Transformation 1*. A fuller discussion of this transformation along with related material can be found in [13, 32, 34]. Transformation 1 provides the general procedure alluded to earlier that can in principle be employed to transform any problem in the form of (3) into an equivalent instance of UQP.

For problems where VIPs are known in advance, as by the penalty transformations given in Table 1, it is usually preferable to use the known VIP directly rather than applying Transformation 1. One special constraint in particular

$$x_j + x_k \le 1$$

where the VIP takes the simple form Px_jx_k appears in many important applications. Due to the broad applicability of this constraint, it is convenient to refer to this special case as *Transformation 2*.

This process of transforming a given problem into the unified framework of xQx is illustrated by the following three examples.

Example 1 Set Packing

Set packing problems have the form Max cx: Ax $\le$ e and x binary where A is a matrix of 0s and 1s and c and e are both vectors of 1s. These problems are important in the field of combinatorial optimization due to their application potential and their computational challenge. Consider the following example:

$$\begin{aligned} \max\ & x_0 = x_1 + x_2 + x_3 + x_4 \\ \text{st}\ & \\ & x_1 + x_3 + x_4 \le 1 \\ & x_1 + x_2 \le 1 \end{aligned}$$

Table 1 Known penalties

Classical constraint	Equivalent penalty (VIP)
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$

The VIPs in Table 1 make it possible to immediately recast this classical model into the UQP unified framework. Representing the positive scalar penalty P by 2M, the equivalent unconstrained problem is

$$\max\ x_0 = x_1 + x_2 + x_3 + x_4 - 2Mx_1x_3 - 2Mx_1x_4 - 2Mx_3x_4 - 2Mx_1x_2$$

which can be rewritten as

$$\max\ (x_1\ x_2\ x_3\ x_4) \begin{bmatrix} 1 & -M & -M & -M \\ -M & 1 & 0 & 0 \\ -M & 0 & 1 & -M \\ -M & 0 & -M & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

This model has the form max $x'Qx$ where Q, as shown above, is a square, symmetric matrix. The procedure illustrated here can be used with any set packing problem and has proven to be an effective approach for solving problems with thousands of variables and constraints (see [4]).

Example 2 Set Partitioning

The classical set partitioning problem has the form Min dx: Ax = e, x binary where again A is a matrix of 0s and 1s, e is a vector of 1s, and (in contrast to the vector c of set packing problems) d is a vector of integers, typically nonnegative. The set partitioning problem is found in applications that range from vehicle routing to crew scheduling [36, 51]. As an illustration, consider the following small example:

$$\min x_0 = 3x_1 + 2x_2 + x_3 + x_4 + 3x_5 + 2x_6$$

subject to

$$\begin{aligned} x_1 + x_3 + x_6 &= 1 \\ x_2 + x_3 + x_5 + x_6 &= 1 \\ x_3 + x_4 + x_5 &= 1 \\ x_1 + x_2 + x_4 + x_6 &= 1 \end{aligned}$$

and x binary. Applying Transformation 1 with P = 10 gives the equivalent UQP model:

$$\min x\hat{Q}x, \quad x \textit{ binary}$$

where the additive constant, c, is 40 and the symmetric $\widehat{Q}$ matrix is

$$\hat{Q} = \begin{bmatrix} -17 & 10 & 10 & 10 & 0 & 20 \\ 10 & -18 & 10 & 10 & 10 & 20 \\ 10 & 10 & -29 & 10 & 20 & 20 \\ 10 & 10 & 10 & -19 & 10 & 10 \\ 0 & 10 & 20 & 10 & -17 & 10 \\ 20 & 20 & 20 & 10 & 10 & -28 \end{bmatrix}$$

Solving this UQP formulation provides an optimal solution $x_1 = x_5 = 1$ (with all other variables equal to 0) to yield $x_0 = 6$. In the straightforward application of Transformation 1 to this example, the replacement of the original problem formulation by the UQP model does not require any new variables to be introduced. Set partitioning problems with thousands of variables have been successfully solved by reformulating them in this manner, as reported in [47].

In many applications, Transformations 1 and 2 can be used in concert to produce an equivalent UQP model, as demonstrated next.

Example 3 The K-Coloring Problem

Vertex coloring problems seek to assign colors to nodes of a graph in such a way that adjacent nodes receive different colors. The K-coloring problem attempts to find such a coloring using exactly K colors. A wide range of applications, ranging from frequency assignment problems to printed circuit board design problems, can be represented by the K-coloring model.

These problems can be modeled as satisfiability problems using the assignment variables as follows:

Let x_{ij} be 1 if node i is assigned color j, and 0 otherwise.

Since each node must be colored,

$$\sum_{j=1}^{K} x_{ij} = 1 \quad i = 1, \ldots, n \tag{6}$$

where n is the number of nodes in the graph. A feasible coloring, in which adjacent nodes are assigned different colors, is assured by imposing the constraints

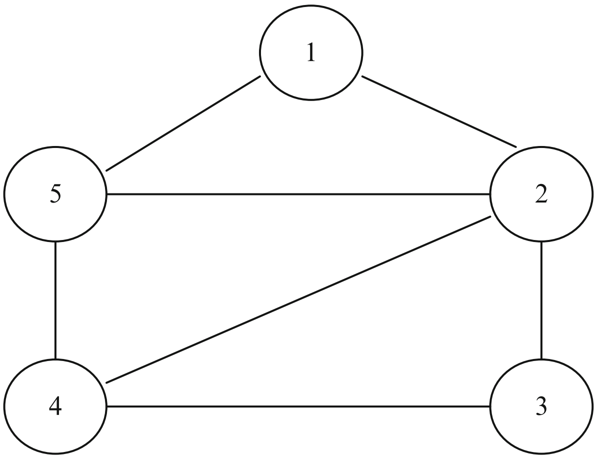
$$x_{ip} + x_{jp} \le 1 \quad p = 1, \ldots, K \tag{7}$$

for all adjacent nodes (i, j) in the graph.

This problem can be recast in the form of UQP by using Transformation 1 on the assignment constraints of (6) and Transformation 2 on the adjacency constraints of (7). No new variables are required. Since the resulting model has no explicit objective function, any positive value for the penalty P will do. The following example gives a concrete illustration of the reformulation process.

Find a feasible coloring of the following graph using three colors.

Thus, the goal is to find a solution to the system:



$$x_{i1} + x_{i2} + x_{i3} = 1 \quad i = 1,\ 5 \tag{8}$$

$$x_{ip} + x_{jp} \leq\ 1 \quad p = 1,\ 3 \tag{9}$$

$$\text{(for all adjacent nodes i and j)}$$

In this traditional form, the model has 15 variables and 26 constraints. To recast this problem in the form of UQP, it suffices to use Transformation 1 on the equations of (8) and Transformation 2 on the inequalities of (9). Arbitrarily choosing the penalty P to be 4, the equivalent problem in unified form is given by

$$\text{UQP}(\textit{Pen}) : \min\ x\hat{Q}x$$

where the $\hat{Q}$ matrix is

$$\hat{Q} = \begin{bmatrix}
-4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
4 & -4 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 \\
0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 4 & 0 \\
0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 4 \\
0 & 0 & 0 & 4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 & 4 \\
4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & -4 & 4 & 4 \\
0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 4 & -4 & 4 \\
0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & -4
\end{bmatrix}$$

Solving this unconstrained model, $x\hat{Q}x$, yields the feasible coloring:

$$x_{11},\ x_{22},\ x_{33},\ x_{41},\ x_{53},\ = 1 \ \text{ all other } \ x_{ij} = 0$$

This approach to coloring problems has proven to be very effective for a wide variety of coloring instances from the literature as disclosed in [41].

2 Solving UQP

Employing the UQP unified framework to solve combinatorial problems requires the availability of a solution method for xQx. The recent literature reports major advances in such methods involving modern metaheuristic methodologies. The reader is referred to references [6–9, 14, 16, 24–26, 37, 45, 49, 50, 53, 54] for a description of some of the methods that have provided valuable contributions to the area. The pursuit of further advances in solution methods for xQx remains an active research arena.

The computational work reported later in this chapter derives from previous studies of three methods: a basic tabu search method due to Glover, Kochenberger, and Alidaee [23–25], a tabu search method due to Lewis [47], and the more recent tabu search method of Glover, Jin-Kao, and Lu [29]. For convenience these methods will be referred to as methods 1, 2, and 3 respectively. A brief overview of each approach is given below. Complete details are provided in the aforementioned references.

Method 1 Overview: This tabu search metaheuristic for UQP is centered around the use of strategic oscillation, which constitutes one of the primary strategies of tabu search. The method alternates between constructive phases that progressively set variables to 1 (whose steps are called "add moves") and destructive phases that progressively set variables to 0 (whose steps are called "drop moves"). To control the underlying search process, the method uses a memory structure that is updated at *critical events*, identified by conditions that generate a subclass of locally optimal solutions. Solutions corresponding to critical events are called *critical solutions*.

A parameter SPAN is used to indicate the amplitude of oscillation about a critical event. To begin *span* is set equal to 1 and then is gradually increased until reaching some limiting value. For each value of *span*, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, *span* is gradually decreased, allowing again for a series of alternating constructive and destructive phases. When *span* reaches a value of 1, a *complete span cycle* has been completed and the next cycle is launched. The search process is typically allowed to run for a preset number of SPAN cycles.

Information stored at critical events is used to influence the search process by penalizing potentially attractive add moves (during a constructive phase) and inducing drop moves (during a destructive phase) associated with assignments of values to variables in recent critical solutions. Cumulative critical event information is used to introduce a subtle long-term bias into the search process by means of additional penalties and inducements similar to those discussed above. Other standard elements of tabu search such as short- and long-term memory structures are also included.

Method 2 Overview: This method is a modification of the previous method that implements a basic multi-start, tabu search procedure with path-relinking. A local search is performed using a 1-opt mechanism. When no improvements to the current solution can be found using 1-opt local search, including tabu aspiration possibilities, a path-relinking procedure is initiated between the current solution and an elite set of diverse solutions saved during the search process. After relinking, a random portion of the current solution is perturbed and testing continues. If after a number of iterations (specified by the restart limit) no improvements are found, then a larger perturbation is invoked based on long-term memory.

Method 3 Overview: The DDTS method repeatedly alternates between a simple version of tabu search (TS) and a diversification phase founded on a memory-based perturbation operator. Starting from an initial random solution, DDTS uses the TS procedure to reach a local optimum. Then, the perturbation operator is applied to displace the solution to a new region, whereupon a new round of TS is launched. To facilitate achieving effective diversification, the perturbation operator is guided by information from a special memory structure.

This tabu search procedure uses a neighborhood defined by the single 1-flip moves, which consists of changing (flipping) the value of a single variable x_j to its complement value $1 - x_j$. The implementation of this neighborhood uses a fast incremental evaluation technique to calculate the cost of candidate moves. The diversification strategy utilizes a memory-based perturbation operator composed of three parts: a flip frequency memory, an elite solution memory, and an elite value frequency memory. These memory structures are used jointly by the perturbation operator to enhance the diversification of the search process. This method has proven to be highly effective for solving large instances of UQP. Complete details of this method are given in Glover, Lu, and Hao [29].

3 Applications

To date several important classes of combinatorial problems have been successfully modeled and solved by employing the unified framework. Results with the unified framework applied to these problems have been uniformly attractive in terms of both solution quality and computation times. While the three solution methods described above are designed for the completely general form of UQP, without any specialization to take advantage of particular types of problems reformulated in this general representation, the outcomes typically prove competitive with or even superior to those of specialized methods designed for the specific problem structure at hand. The broad base of experience with UQP as a modeling and solution framework obtained by applying the three methods above includes a substantial range of problem classes including quadratic assignment problems, capital budgeting problems, multiple knapsack problems, task allocation problems (distributed computer systems), maximum diversity problems, p-median problems, asymmetric assignment problems, symmetric assignment problems, side constrained assignment problems, quadratic knapsack problems, constraint satisfaction problems (CSPs), set partitioning problems, fixed charge warehouse location problems, maximum clique problems, maximum independent set problems, maximum cut problems, graph coloring problems, graph partitioning problems, number partitioning problems, and-linear ordering problems.

Additional test problems representing a variety of other applications have also been reformulated and solved via UQP. The section below reports specific computational experience with some large-scale applications. Section 5 then suggests promising new applications areas for UQP.

4 Illustrative Computational Experience

The following summarizes results obtained on large-scale test problems of a variety of well-known problem classes from combinatorial optimization. Because methods 1, 2, and 3 were developed at different points in time, and each has been applied to classes of problems different from those treated by the other two, outcomes for these classes of problems are reported by describing the findings for the three methods in the sequence in which they have been applied.

4.1 The Task Allocation Problem

The task allocation problem, which consists of assigning tasks to processors, can be described as follows: Let $\mathrm{P} = \{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_\mathrm{m}\}$ be a set of distributed processors and $T = \{T_1, T_2, \ldots T_n\}$ be a set of tasks to be run on the processors. Let c_{ij} be the communications cost between tasks T_i and T_j and q_{tp} be the execution cost of task t on processor p. Then stipulating that x_{tp} equals 1 if task T_t is assigned to processor P_p and is otherwise equal to 0, the model becomes

$$\begin{aligned} &\min \sum_{t=1}^{n} \sum_{p=1}^{m} q_{tp}\, x_{tp} + \sum_{i<j}^{n} \sum_{p=1}^{m} c_{ij}\, x_{ip}(1 - x_{jp}) \\ &\text{st} \\ &\qquad \sum_{p=1}^{m} x_{tp} = 1 \quad \text{for } t = 1, \ldots, n \end{aligned}$$

which is of the form

$$\begin{aligned} &\min\ x'Qx \\ &\text{st} \\ &Ax = b \end{aligned}$$

Applying Transformation 1 of Sect. 1.2 yields a model in the form of UQP. In a study [46] comparing CPLEX 8.1 and method 1 on 16 large test problems ranging in size from 1,000 to 3,000 variables, CPLEX was unable to prove optimality on any of the problems (within a 48-h limit). Running method 1 for a total of 300 SPAN cycles took less than 12 min on the largest of the problems. Across all 16 problems, the best solutions produced by CPLEX had objective function values that were, on average, 52 % inferior to those produced by method 1, in spite of being allowed to run 240 times longer.

4.2 The Max 2-Sat Problem

It is well known (see for instance [13] and [33]) that the Max 2-Sat problem can be formulated as an unconstrained binary quadratic program of the form min $x'Qx$. Yet little computational experience treating the Max 2-Sat problem as an instance of UQP has appeared in the literature prior to the computational study of [42] which reports on the successful application of the tabu search Method 1 to a variety of old and new test problems from this class. In all cases, method 1 was run for 50 SPAN cycles. Applied to a public data set with 16 problems ranging from 50 variables and 100 clauses to 150 variables and 600 clauses, method 1 found best known solutions in less than 3 s while more than half of these problem instances could not be solved within a 12-h limit by the well-known Maxsat [10] procedure.

On another publicly available set of 20 problems with 200 variables and 1,000–2,000 clauses, method 1 again found best known solutions in less than 3 s. On a third data set of 17 problem instances ranging in size from 100 to 1,000 variables and 626 to 22,883 clauses, method 1 was tested against CPLEX 8.0 by applying the latter to the classical MIP formulation of Max 2-Sat. CPLEX was only able to solve and terminate naturally on the smallest of these problems, requiring 147 s in this case. Method 1 by contrast found the optimal solution for this problem in less than 1 s. On the other 16 problems, CPLEX was unable to terminate naturally within a 10-h time limit while the UQP approach of method 1 obtained solutions within 50 SPAN cycles in an average of less than 9 s. For these problems, the best solutions found in the 10-h time limit by CPLEX had objective function values on average 20 % inferior to those produced by the method 1 tabu search approach. Full details of these results are given in [42].

4.3 The Group Technology Problem

The group technology (GT) problem is concerned with clustering machines and parts together in a manner that facilitates economies in time and cost. From a graph theoretic point of view, nodes in a graph can be taken to represent machines and parts, which are connected by edges denoting the association of each pair of nodes in the network. The GT problem then becomes one of partitioning the nodes into cliques with similar characteristics. Thus, the GT problem can be modeled by the standard IP formulation for clique partitioning:

$$\max \sum_{(i,j)\in E} w_{ij}\, x_{ij}$$

$$\begin{aligned}&\text{st}\\ &\quad x_{ij} + x_{ir} - x_{jr} \le 1 \quad \forall \text{ all distinct } i, j.r \in V\\ &\quad x_{ij} \in \{0,1\} \text{ for all } \{i,j\} \in E\end{aligned}$$

The variable x_{ij} is equal to 1 if machine (or part) i and machine (or part) j are assigned in a cell and is equal to 0 otherwise. The coefficient w_{ij} is the weight of the edge (i, j) in the graph.

Since the variables are associated with edges in the graph, the model has many variables and constraints. In practice even modest-sized GT problems give rise to extraordinarily large IP models and often are beyond the capability of modern MIP solvers to handle. As an alternative to the standard IP model for clique partitioning, it is shown in [6] that the GT problem can be solved by employing the following quadratic model for clique partitioning:

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \sum_{k=1}^{K\max} x_{ik}\, x_{jk}$$

$$\begin{aligned}&\text{st}\\ &\quad \sum_{k=1}^{k\max} x_{ik} = 1 \quad \text{for } i = 1, n\end{aligned}$$

In this formulation n is the number of nodes, w_{ij} again denotes the weight of edge (i, j), Kmax is an upper bound on the number of groups to be formed, and $x_{ik} = 1$

if node i is assigned to clique k and otherwise equals zero. It is to be noted that variables are associated with nodes rather than edges in this model. Thus, in spite of being nonlinear, the model is much smaller than the standard IP model and is readily approached via the unified framework. Specifically, the problem has the form

$$\begin{aligned}&\max\ x'Qx\\&\text{st}\\&Ax = b\end{aligned}$$

and thus can be recast in the form of UQP using Transformation 1 of Sect. 1.2.

To test the effectiveness of the UQP approach for modeling GT, experiments were conducted on 36 standard test problems of modest size ranging from graphs with 46 nodes and 1,035 edges to 71 nodes and 2,485 edges. Each problem was solved with method 1 running for 100 SPAN cycles, with the largest problem taking slightly more than 1 min. To provide a benchmark for comparison, each problem was solved in its standard MIP formulation by CPLEX 6.5. CPLEX's time performance on these problems was very erratic. The average solution time was 1.2 days, and 4 problems required more than 4 days. Nonetheless, CPLEX terminated naturally on all 36 problems and thus confirmed the optimality of the solutions obtained by the UQP approach although requiring running times that were 1,500–6,000 times longer. Complete details are given in Wang et al. [61].

4.4 The Set Packing Problem

Example 1 of Sect. 1 of this chapter presented a small example of a set packing problem, illustrating how this class of problems can easily be reformulated as an unconstrained quadratic binary program. In general, this class of problems is given by

$$\begin{aligned}&\max\ \sum_{j=1}^{n} w_j\, x_j\\&\text{st}\\&\sum_{j=1}^{n} a_{ij}\, x_j \le 1 \quad \text{for } i = 1, \dots m\end{aligned}$$

where the a_{ij} are 0/1 coefficients, the w_j are weights, and the x_j variables are binary. Each constraint can alternatively be enforced by subtracting one or more quadratic penalties from the objective function, thus producing an equivalent problem in the form of an unconstrained quadratic binary program. Note that this recasting takes place without the introduction of new variables. Moreover, the size of the equivalent UQP model depends only on the number of original variables and is independent of the number of constraints in the original set packing problem.

Computational results are given in [4] for applying method 1 to a set of publicly available set of 16 test problems containing 1,000–2,000 variables and 2,000–10,000 constraints. Best known results for these problems reported in the literature are obtained by a leading heuristic (see [19]) specially designed for set packing problems. This special procedure was allowed to run for 5 h on each problem. For each problem,method 1 was run for 1,000 SPAN cycles, taking 20 min for the largest instance and 7 min on average. For the 16 test problems, method 1 quickly found the best known solutions for 13 of the 16 problems and produced solutions whose objective function values were more than 99 % of the best known values for the other 3 problems. To provide additional comparison, these problems were also solved using CPLEX 8.1 which consumed on average 38 h of computer run time before terminating due to reaching memory limitations. The reader is referred to [4] for complete details of these runs as well as additional computational experience.

4.5 The Set Partitioning Problem

As noted in Example 2 of Sect. 1, the set partitioning problem can be formulated as

$$\begin{aligned}
&\max \sum_{j=1}^{n} c_j x_j \\
&\text{st} \\
&\sum_{j=1}^{n} a_{ij} x_j = 1 \quad \text{for } i = 1, \ldots m
\end{aligned}$$

where the a_{ij} are 0/1 coefficients, the c_j are objective function coefficients, and the x_j variables are binary. Computational experience with a wide variety of solution methods has shown that even modest-sized instances of these problems are extraordinarily difficult to solve and become increasingly difficult as density grows. Applying Transformation 1, the set partitioning problem becomes a UQP problem without introducing new variables and whose size is independent of the number of constraints in the original problem.

An equivalent but simpler alternative to Transformation 1 is presented in [47] for re-expressing set partitioning problems as UQP models. Computational experience is reported with a set of 31 test problems ranging in size from 600 to 15,000 variables and 100 to 5,000 constraints. CPLEX 8.1, applied to the classical formulation given above, was used to provide a benchmark for comparison with the basic tabu search and path relinking approach of method 2. For each problem, CPLEX was run until optimality was proven or until reaching 12 h of processing time (or until the run was terminated by an "out of memory" error). For the first 21 problems, CPLEX terminated naturally with an optimal solution. Method 2 quickly found an optimal solution to these problems as well with a time performance that was on average 270 times faster than CPLEX. (This excludes the time it took CPLEX to prove the optimality of the solutions it found).

For the largest problem (15,000 variables and 5,000 constraints), CPLEX terminated with a memory fault and no solution while method 2 found a feasible solution in less than 20 min. For the other problems, CPLEX terminated due to the

12-h time limit, giving a "best solution found so far." On these problems, method 2 obtained solutions with objective functions approximately 2 % superior to those obtained by CPLEX while running more than 100 times faster.

4.6 The Linear Ordering Problem

The linear ordering problem (LOP) is a particularly hard problem defined by an n-by-n matrix of weights $C = \{c_{ij}\}$ where the goal is to find a permutation, p, of columns (and rows) that maximizes the sum of the weights in the upper triangular matrix. Such problems arise in a variety of settings (such as finding an acyclic tournament of maximum weight, or the aggregate ordering of paired observations) but are most often associated with the triangulation of input-output matrices in economics where the data in question often refer to sectors. The problem can be modeled utilizing the decision variable: $x_{ij} = 1$ if sector i goes before sector j in the permutation; and $x_{ij} = 0$ otherwise. Taking advantage of the fact that $x_{ij} + x_{ji} = 1$ for all i and j, a standard integer programming formulation for the problem is given by

$$\begin{aligned}
&\text{Max} \sum_{i<j} c_{ij}\, x_{ij} + \sum_{j<i} c_{ij}\left(1 - x_{ji}\right)\\
&\text{st}\\
&x_{ij} + x_{jk} - x_{ik} \le 1 \quad \forall\,(i,j,k):\ i<j<k\\
&x_{ij} + x_{jk} - x_{ik} \ge 0 \quad \forall\,(i,j,k):\ i<j<k\\
&x_{ij} \in \{0,1\}\ \ \forall\,(i,j):\ i<j
\end{aligned}$$

This model can be recast into the form of UQP by employing a specially crafted quadratic penalty, a penalty that is unique to the structure of the linear ordering problem. To see how this special penalty arises, note that for a particular set $i < j < k$, the *pair* of constraints shown above allows 6 of the 8 possible solutions, excluding only $x_{ij} = 1,\ x_{jk} = 1$, and $x_{ik} = 0$ and $x_{ij} = 0,\ x_{jk} = 0$, and $x_{ik} = 1$. It is easy to see that an exact quadratic penalty that precludes these same two solutions, while allowing the others, is given by

$$P\left\{x_{ik} + x_{ij}x_{jk} - x_{ij}x_{ik} - x_{jk}x_{ik}\right\}$$

Thus, without introducing additional variables, this special penalty can be used to easily transform the linear ordering problem into an equivalent UQP. For a problem with n sectors, both the IP formulation and the equivalent UQP model will have $n(n-1)/2$ variables. Again the size of the UQP model is independent of the number of constraints in the original IP model.

An application of this UQP approach is reported in [48] for a set of 11 test problems ranging in size from 190 variable and 2,280 constraints (a LOP instance with $n=20$) to 19,900 variables and 2,626,800 constraints (a LOP instance with $n=200$). As before, experience with CPLEX was reported on these problems to provide a benchmark of comparison. Both CPLEX and method 2 were given a time limit of 1 h. Best known solutions, obtained from a specially crafted Scatter Search method for LOP problems, are also available for purposes of benchmarking.

Due to the large size of the IP models, CPLEX was able to find and prove optimality for only the two smallest problems. Method 2 found these same optimal solutions within the 1-h time limit. For the four largest problems (more than 4,000 variables and 234,960 constraints), CPLEX was unable to find a feasible solution within the 1-h time limit while the UQP approach readily found feasible solutions. For the other problems, CPLEX reported solutions that were inferior to those produced by the tabu search/path-relinking approach of method 2. Across the entire set of test problems, solutions were produced using the UQP model whose objective function values were at least 95 % of the best known values obtained by any method in the literature specialized for solving LOP problems. See [48] for a complete discussion of these results.

4.7 The Max-Cut Problem

Given an undirected graph G(V,E) with edge weights w_{ij}, the Max-Cut (MC) problem seeks a partition $S_1 \subset V$ and $S_2 = V|S_1$ such that the weight of the cut, defined as the sum of the weights on the edges connecting the two sets, is maximized. MC is another classic problem in combinatorial optimization. This problem has a natural quadratic structure and with a simple change of variables, it can be readily put into the form of UQP. The common formulation from the literature is

$$\max \ \tfrac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j)$$

subject to

$$y_i \in \{-1, 1\} \ \ \forall\, i \in V$$

The change of variables $y_i = 2x_i - 1$ yields the unconstrained quadratic binary program

$$\max \sum_{i<j} w_{ij}(x_i + x_j - 2x_i x_j); \quad x_j \in \{0, 1\}$$

which is of the form $\max \ x'Qx$ This class of problems is described in detail in [28] together with reporting computational experience for 69 test problems from the literature ranging in size from 800 variables to 10,000 variables. Most articles in the literature addressing MC only consider problems with up to a few thousand variables.

This study employed the tabu search approach of method 3. Depending on the problem size, run time limits ranged from one half hour for small problems to 24 h for the 10,000 variable problems. The results obtained were highly attractive compared to previously published results in the literature, as evidenced by the fact that, over the entire test bed of 69 problems, method 3 matched best known solutions on 19 problems, found new best known solutions on 46 problems, and failed to find best known solutions on just 4 problems.

4.8 Comments on Computational Experience

The computational experience reported above is intended to demonstrate the viability and breadth of applicability of the unified framework. This framework has been successfully applied to many other problem classes as well. While the intention of the present chapter is to disclose the general applicability of the unified modeling and solution methodology, and not to provide a comprehensive comparison of this approach with the best-performing specialized methods for each class of problems at this time, it should nonetheless be emphasized that the results presented clearly establish that the reformulation approach not only works across a wide array of problem classes but works very well. The approach finds best known solutions for many problems, regardless of problem class, in modest computer times. As future studies take advantage of the opportunity to apply more advanced UQP algorithms such as method 3 across a wider range of applications, additional records will undoubtedly be set for finding best known solutions to test problems from a variety of sources.

4.9 Important Alternative Model for Assignment Problems

Many important classes of problems have assignment constraints where, generally, agents of some kind are assigned to tasks. For such problems, Transformation 1 can be used to enforce the assignment constraints via quadratic penalties as illustrated earlier in this chapter. For such problems, however, a slightly different manner of constructing the quadratic penalty matrix, Q, has proven to be attractive in certain cases provided that resulting quadratic optimization problem is carried out subject to a cardinality constraint rather than being unconstrained. Recall that Transformation 1 results in an additive constant and a modification of the elements on the main diagonal of the Q matrix. With the alternative method, neither the additive constant nor modified diagonal elements are employed. The idea is illustrated by the following example:

Suppose the goal is to obtain solutions that satisfy

$$\begin{aligned} x_{11} + x_{12} + x_{13} &= 1 \\ x_{21} + x_{22} + x_{23} &= 1 \\ x_{31} + x_{32} + x_{33} &= 1 \end{aligned}$$

Clearly exactly three variables will be equal to one and the other six variables will be zero. Finding solutions that satisfy the above assignment constraints is equivalent to solving the problem

$$\begin{aligned} &\min x_0 = x'Qx \\ &\text{st} \\ &\sum_{i=1}^{3} \sum_{j=1}^{3} x_{ij} = 3 \end{aligned}$$

where

$$x = (x_{11,}x_{12}, x_{13}, x_{21}, \dots x_{33})$$

and the Q matrix is given by

$$Q = \begin{bmatrix} 0 & P & P & 0 & 0 & 0 & 0 & 0 & 0 \\ P & 0 & P & 0 & 0 & 0 & 0 & 0 & 0 \\ P & P & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P & P & 0 & 0 & 0 \\ 0 & 0 & 0 & P & 0 & P & 0 & 0 & 0 \\ 0 & 0 & 0 & P & P & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & P & P \\ 0 & 0 & 0 & 0 & 0 & 0 & P & 0 & P \\ 0 & 0 & 0 & 0 & 0 & 0 & P & P & 0 \end{bmatrix}$$

where P is a positive scalar penalty. Note that the block diagram structure along the main diagonal makes this penalty matrix particularly easy to construct.

The cardinality constraint requires exactly three of the nine variables to be equal to 1. The penalty structure of Q, given the goal of forcing x_0 to zero, will not allow, say, x_{12} or x_{13} to be 1 in the event that x_{11} is equal to one. In this manner, all three assignment constraints are enforced via the penalties.

In applications involving assignment constraints, there are typically additional constraints (besides the assignment constraints), and these need to be "folded" into the Q matrix as well. This approach is particularly well suited for quadratic assignment problems where facilities are assigned to locations, clustering problems where data are assigned to clusters, coloring problems where colors are assigned to nodes, clique partitioning problems where nodes are assigned to cliques, and so forth.

5 Promising New Application Areas

This combined modeling/solution approach provides a unifying theme that can be applied in principle to all linearly constrained quadratic and linear programs in bounded integer variables, and the computational findings for a broad spectrum of problem classes raises the possibility that similarly successful results may be obtained for even wider ranges of problems. The generality of the approach of modeling and solving problems using the UQP formulation invites additional applications to be pursued via this unified framework. Promising applications that are part of current work in progress include:

1. *The variable (feature) selection problem*: This important problem in linear regression concerns choosing a set of independent variables that are strongly correlated with the independent variable and weakly correlated with one another. One model for this, given by Eksioglu et al. [20], takes the form of a

two-objective IP model that can be easily recast into the unified $x'Qx$ framework. Tests are currently underway applying this alternative approach to a variety of data sets and making comparisons with standard statistical approaches.

2. *Clustering*: Clustering is an important data mining tool with applications in many important areas from medicine to marketing. Taking a graph-theoretical perspective, clustering can be viewed as a clique partitioning problem, and thus the quadratic model presented in Sect. 4.3 can be used to cluster data. In this setting too, tests are in process to extend the encouraging early results from this approach reported in Kochenberger et al. [43] and in Wang et al. [62].
3. *Computational biology*: Several important problems arising in biology can be modeled and solved as combinatorial optimization problems. Of particular interest are the multiple sequence alignment problem, the lattice protein folding problem, the rotamer assignment problem and the contact map optimization problem. Each of these problems, as explained in Forrester and Greenberg [21], can be modeled as a constrained quadratic optimization problem in zero-one variables, thus permitting them to be transformed into the $x'Qx$ framework and solved by solution methods designed for the unified framework. Early testing of this approach is underway.
4. *General linear 0/1 programming*: The general 0/1 linear programming problem can be represented by

$$\begin{aligned}&\max\ \mathrm{cx}\\&\mathrm{st}\\&\mathrm{Ax} = b\\&\mathrm{x}\ \textit{binary}\end{aligned}$$

By using Transformation 1 it is possible to recast the problem in the form of

$$\begin{aligned}&\max\ x_0 = x^t Q x\\&\mathrm{st}\ \ x\ \textit{binary}\end{aligned}$$

For problems with inequality constraints, slack variables, via a binary expansion, can always be introduced to create the system of constraints $Ax = b$. This procedure is illustrated by the following example:

$$\begin{aligned}&\max\ \ 6x_x + 4x_2 + 8x_3 + 5x_4 + 5x_5\\&\mathrm{st}\\&\qquad 2x_1 + 2x_2 + 4x_3 + 3x_4 + 2x_5 \le 7\\&\qquad 1x_1 + 2x_2 + 2x_3 + 1x_4 + 2x_5 = 4\\&\qquad 3x_1 + 3x_2 + 2x_3 + 4x_4 + 4x_5 \ge 5\\&\qquad\ \ x \in \{0, 1\}\end{aligned}$$

Adding slack variables for the 1st and 3rd constraints

$$\begin{aligned} 0 \le s_1 \le 3 \quad &\Rightarrow s_1 = 1x_6 + 2x_7 \\ 0 \le s_3 \le 6 \quad &\Rightarrow s_3 = 1x_8 + 2x_9 + 4x_{10} \end{aligned}$$

gives the system $Ax = b$ with A given by

$$A = \begin{bmatrix} 2 & 2 & 4 & 3 & 2 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 2 & 4 & 4 & 0 & 0 & -1 & -2 & -4 \end{bmatrix}$$

For a scalar penalty $P = 10$, applying Transformation 1 gives the equivalent problem

$$\max\ x_0 = x'Qx$$

with an additive constant of -900 and a Q matrix

$$Q = \begin{bmatrix} 526 & -150 & -160 & -190 & -180 & -20 & -40 & 30 & 60 & 120 \\ -150 & 574 & -180 & -200 & -200 & -20 & -40 & 30 & 60 & 120 \\ -160 & -180 & 688 & -220 & -200 & -40 & -80 & 20 & 40 & 80 \\ -190 & -200 & -220 & 645 & -240 & -30 & -60 & 40 & 80 & 160 \\ -180 & -200 & -200 & -240 & 605 & -20 & -40 & 40 & 80 & 160 \\ -20 & -20 & -40 & -30 & -20 & 130 & -20 & 0 & 0 & 0 \\ -40 & -40 & -80 & -60 & -40 & -20 & 240 & 0 & 0 & 0 \\ 30 & 30 & 20 & 40 & 40 & 0 & 0 & -110 & -20 & -40 \\ 60 & 60 & 40 & 80 & 80 & 0 & 0 & -20 & -240 & -80 \\ 120 & 120 & 80 & 160 & 160 & 0 & 0 & -40 & -80 & -560 \end{bmatrix}$$

Solving $\max\ x_0 = x'Qx$ gives the nonzero values

$$x_1 = x_4 = x_5 = x_9 = x_{10} = 1$$

for which $x_o = 916$. Adjusting for the additive constant gives an objective function value of 16 which is optimal. Note that any linear problem in bounded integer variables, through a binary expansion, could be converted into $\max\ x_0 = x'Qx$ as illustrated here. Note also, though, that the elements of the Q matrix can, for some problems, get unacceptably large and may require suitable scaling to mitigate this problem.

5. *The Max 3-Sat Problem:* Section 4.2 discusses the Max 2-Sat problem, showing how it can be reformulated and solved as $\min\ x^tQx$. The penalty approach adopted for the Max 2-Sat problem can be expanded to address the Max 3-Sat problem. In this case, however, the penalty function that results is a cubic rather than a quadratic function. Nonetheless, by a simple transformation, the cubic function can be recast as a quadratic and thus the Max 3-Sat problem, like

the Max 2-Sat problem, can also be modeled and solved using the min x^tQx formulation.

In particular, the Max 3-Sat problem gives rise to four possible clause types. A linear constraint is associated with each which can be carried into the problem objective by a cubic penalty function which is equal to zero when the constraint is satisfied and otherwise equal to one. Finding solutions that maximize the number of clauses satisfied then corresponds to minimizing the penalty function that results by summing the individual penalties. The procedure is illustrated below.

The four clause types together with their associated constraint and penalty are:

1. *No negations*:
   Linear constraint: $x_i + x_j + x_k \geq 1$
   Cubic penalty: $(1 - x_ix_jx_k - x_i - x_j - x_k + x_ix_j + x_ix_k + x_jx_k)$
2. *One negation*:
   Linear constraint: $x_i + x_j + \bar{x}_k \geq 1$
   Cubic penalty: $(x_k - x_ix_k - x_jx_k + x_ix_jx_k)$
3. *Two negations*:
   Linear constraint: $x_i + \bar{x}_j + \bar{x}_k \geq 1$
   Cubic penalty: $(x_jx_k - x_ix_jx_k)$
4. *Three negations*:
   Linear constraint: $\bar{x}_i + \bar{x}_j + \bar{x}_k \geq 1$
   Cubic penalty: $(x_ix_jx_k)$

The conversion of the cubic penalty function to an equivalent quadratic function is carried out by the reduction procedure of Boros and Hammer [11] by replacing a product term xy by a new binary variable z and adding the penalty term $P(xy - 2xz - 2yz + 3z)$ to the objective function as illustrated in the following example with $n = 5$ variables and 12 clauses.

Clause #	Clause
1	$x_1 \vee x_2 \vee x_3$
2	$x_1 \vee \bar{x}_2 \vee x_3$
3	$\bar{x}_1 \vee x_2 \vee \bar{x}_3$
4	$x_2 \vee \bar{x}_3 \vee x_4$
5	$\bar{x}_2 \vee x_3 \vee x_4$
6	$\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$
7	$x_2 \vee x_4 \vee x_5$
8	$\bar{x}_2 \vee x_3 \vee x_5$
9	$x_2 \vee \bar{x}_3 \vee x_5$
10	$x_3 \vee x_4 \vee x_5$
11	$x_3 \vee \bar{x}_4 \vee \bar{x}_5$
12	$\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5$

Introducing the penalty functions for these clauses gives the following cubic penalty function to minimize:

$$\begin{aligned} x_0 = {} & 3 - x_1 + x_2 - 2x_4 - 2x_5 + 2x_1x_3 - 4x_2x_3 + 3x_4x_5 \\ & -x_1x_2x_3 + 3x_2x_3x_4 - x_2x_4x_5 - x_3x_4x_5 + 2x_2x_3x_5 \end{aligned}$$

Note that this function has five cubic terms. Introducing two new binary variables, $x_6 = x_2x_3$ and $x_7 = x_4x_5$ along with the associated quadratic penalties yield the equivalent quadratic penalty function:

$$\begin{aligned} x_0 = {} & 3 - x_1 + x_2 - 2x_4 - 2x_5 + 2x_1x_3 - 4x_2x_3 + 3x_4x_5 \\ & -x_1x_6 + 3x_4x_6 - x_2x_7 - x_3x_7 + 2x_5x_6 \\ & +P(x_2x_3 - 2x_2x_6 - 2x_3x_6 + 3x_6) \\ & +P(x_4x_5 - 2x_4x_7 - 2x_5x_7 + 3x_7) \end{aligned}$$

which is of the form

$$x_0 = 3 + (xQx)/2$$

Taking (arbitrarily) the penalty P to be 10, the 7-by-7 Q matrix is

$$Q = \begin{bmatrix} -2 & 0 & 2 & 0 & 0 & -1 & 0 \\ 0 & 2 & 6 & 0 & 0 & -20 & -1 \\ 2 & 6 & 0 & 0 & 0 & -20 & -1 \\ 0 & 0 & 0 & -4 & 13 & 3 & -20 \\ 0 & 0 & 0 & 13 & -4 & 2 & -20 \\ -1 & -20 & -20 & 3 & 2 & 60 & 0 \\ 0 & -1 & -1 & -20 & -20 & 0 & 60 \end{bmatrix}$$

Minimizing xQx yields $x_1 = x_2 = x_3 = x_6 = 1,\ x_4 = x_5 = x_7 = 0$ for which $xQx = -6$. Thus, $x_0 = 0$ implying that all 12 clauses are satisfied at this solution. Note that in carrying out the reduction from the cubic penalty function to the quadratic several choices for variable substitutions were available. In general it is desirable to make these choices in a manner that minimizes the number of new variables that are introduced.

The preceding discussion and illustrations disclose that the unified unconstrained approach offers a fresh and promising alternative method of attack for these important problems. As additional research is conducted to provide enhanced methods for solving the UQP model, the benefits of recasting diverse problems into this general framework will become even greater.

6 Conclusion

A variety of disparate combinatorial optimization problems can be treated by first reexpressing them within the common modeling framework of the unconstrained quadratic binary program. Once in this unified form, the problems can be solved effectively by recently developed solution approaches for UQP.

The empirical findings from extensive testing challenge the conventional wisdom that places high priority on preserving linearity and exploiting specific structure. Although the merits of such a priority are well-founded in many cases, experience with a large variety of problem classes suggests that an unflinching adoption of the conventional *linear* approach may preclude obtaining the best outcomes in a number of cases. In making use of the UQP model, any linearity that the original problem may have exhibited is destroyed. Moreover, any exploitable structure that may have existed originally is "folded into" the $\hat{Q}$ matrix, and a general UQP solution procedure takes no advantage of it. Nonetheless, the use of such procedures has been remarkably successful, yielding results that rival the effectiveness of the best specialized methods.

Acknowledgments The authors would like to acknowledge the contributions of their coworkers Drs. Bahram Alidaee, Jin-Kao Hao, Mark Lewis, Karen Lewis, Zhipeng Lu, and Cesar Rego whose work contributed to the results appearing in this chapter.

Cross-References

►Algorithms for the Satisfiability Problem
►Combinatorial Optimization in Data Mining
►Graph Searching and Related Problems
►On Coloring Problems
►Tabu Search

Recommended Reading

1. B. Alidaee, G. Kochenberger, A. Ahmadian, 0-1 quadratic programming approach for the optimal solution of two scheduling problems. Int. J. Syst. Sci. **25**, 401–408 (1994)
2. B. Alidaee, G. Kochenberger, F. Glover, C. Rego, A new modeling and solution approach for the number partitioning problem. J. Appl. Math. Decis. Sci. **9**(2), 113–121 (2005)
3. B. Alidaee, F. Glover, G. Kochenberger, H. Wang, Solving the maximum edge weight clique problem via unconstrained quadratic programming. Eur. J. Oper. Res. **181**, 592–587 (2007)
4. B. Alidaee, G. Kochenberger, K. Lewis, M. Lewis, H. Wang, A new approach for modeling and solving set packing problems via unconstrained quadratic binary programming. Eur. J. OR **186**(#2), 504–512 (2008)
5. B. Alidaee, G. Kochenberger, K. Lewis, M. Lewis, H. Wang, Computationally attractive non-linear models for combinatorial optimization. Int. J. Math. Oper. Res. **1**(1 and 2), 9–20 (2009)
6. T.M. Alkhamis, M. Hasan, M.A. Ahmed, Simulated annealing for the unconstrained binary quadratic pseudo-boolean function. Eur. J. Oper. Res. **108**, 641–652 (1998)
7. M. Amini, B. Alidaee, G. Kochenberger, A scatter search approach to unconstrained quadratic binary programs, in *New Methods Optimization*, ed. by D. Corne, M. Dorigo, F. Glover (McGraw-Hill, England, 1999), pp. 317–330
8. J.E. Beasley, Heuristic algorithms for the unconstrained binary quadratic programming problem. Working Paper, Imperial College, 1999
9. A. Billionnet, A. Sutter, Minimization of a quadratic pseudo-boolean function. Eur. J. OR **78**, 106–115 (1994)
10. B. Borchers, J. Furman, A two-phase exact algorithm for Max-Sat and weighted Max Sat. J. Comb. Optim. **2**, 299–306 (1999)
11. E. Boros, P. Hammer, The Max-cut problem and quadratic 0-1 optimization, polyhedral aspects, relaxations and bounds. Ann. OR **33**, 151–225 (1991)
12. E. Boros, A. Prekopa, Probabilistic bounds and algorithms for the maximum satisfiability problem. Ann. OR **21**, 109–126 (1989)

13. E. Boros, P. Hammer, Pseudo-boolean optimization. Discret. Appl. Math. **123**(1–3), 155–225 (2002)
14. E. Boros, P. Hammer, X. Sun, The DDT method for quadratic 0-1 minimization, RUTCOR Research Center, RRR 39–89, 1989
15. J.M. Bourjolly, P. Gill, G. Laporte, H. Mercure, A quadratic 0/1 optimization algorithm for the maximum clique and stable set problems. Working paper, University of Montreal, 1994
16. P. Chardaire, A. Sutter, A decomposition method for quadratic zero-one programming. Manage. Sci **41**(4), 704–712 (1994)
17. G. Cornuejolos, *Combinatorial Optimization: Packing and Covering*. CBMS-NSF (SIAM, Philadelphia, 2001)
18. C.M. De Simone, M. Diehl, P. Junger, Mutzel, G. Reinelt, G. Rinaldi, Exact ground State4s of Ising spin glasses: new experimental results with a branch and cut algorithm. J. Stat. Phys. **80**, 487–496 (1995)
19. X. Delorme, X. Gandibleau, J. Rodriques, GRASP for set packing. EJOR **153**, 564–580 (2004)
20. B. Eksioglu, R. Demirer, I. Capar, Subset selection in multiple linear regression: a new mathematical programming approach. Comput. Ind. Eng. **49**, 155–167 (2005)
21. R. Forrester, H. Greenberg, Quadratic binary programming models in computational biology. Algorithmic Oper. Res. **3**, 110–129 (2008)
22. F. Glover, G. Kochenberger, *Handbook of Metaheuristic* (Kluwer Academic, Boston/Dordrecht/London, 2003)
23. F. Glover, M. Laguna, *Tabu Search* (Kluwer Academic, Boston, 1997)
24. F. Glover, G. Kochenberger, B. Alidaee, Adaptive memory tabu search for binary quadratic programs. Manage. Sci. **44**(3), 336–345 (1998)
25. F. Glover, G. Kochenberger, B. Alidaee, M.M. Amini, Tabu with search critical event memory: an enhanced application for binary quadratic programs, in *MetaHeuristics: Advances and Trends in Local Search Paradigms for Optimization*, ed. by S. Voss, S. Martello, I. Osman, C. Roucairol (Kluwer Academic, Boston, 1999)
26. F. Glover, B. Alidaee, C. Rego, G. Kochenberger, One-pass heuristics for large-scale unconstrained binary quadratic programs. EJOR **137**, 272–287 (2002)
27. F. Glover, G. Kochenberger, B. Alidaee, M. Amini, Solving quadratic Knapsack problems by reformulation and tabu search: single constraint case, in *Combinatorial and Global Optimization*, ed. by P.M. Pardalos, A. Migdalas, R. Burkard (World Scientific, River Edge, 2002), pp. 111–121
28. F. Glover, J.K. Hao, G. Kochenberger, Z. Lu, H. Wang, Solving large scale max cut problems via tabu search. Working Paper, University of Colorado at Denver, 2010
29. F. Glover, Z. Lu, J.K. Hao, Diversification-driven tabu search for unconstrained binary quadratic programming. 4OR **8**(3), 239–253 (2010)
30. M. Grotschel, M. Junger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. 36(#3), 493–513 (1988)
31. P. Hammer, S. Rudeanu, *Boolean Methods in Operations Research* (Springer, New York, 1968)
32. P.B. Hansen, Methods of nonlinear 0–1 programming. Ann. Discret. Math. **5**, 53–70 (1979)
33. P. Hansen, B. Jaumard, Algorithms for the maximum satisfiability problem. Computing **44**, 279–303 (1990)
34. P. Hansen, B. Jaumard, V. Mathon, Constrained nonlinear 0-1 programming. INFORMS J. Comput. **5**(2), 97–119 (1993)
35. L.D. Iasemidis, D.S. Shiau, J.C. Sackellares, P. Pardalos, Transition to epileptic seizures: optimization, in *DIMACS Series in Discrete Math and Theoretical Computer Science*, American Mathematical Society Publishing, Providence, RI, vol. 55 (2000), pp. 55–73
36. A. Joseph, A concurrent processing framework for the set partitioning problem. Comput. Oper. Res. **29**, 1375–1391 (2002)

37. K. Katayama, M. Tani, H. Narihisa, Solving large binary quadratic programming problems by an effective genetic local search algorithm, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)* (Morgan Kaufmann, San Francisco, 2000)
38. G. Kochenberger, F. Glover, A unified framework for modeling and solving combinatorial optimization problems: a tutorial, in *Multiscale Optimization Methods and Applications*, ed. by W. Hager, S.-J. Huang, P. Pardalos, O. Prokopyev (Springer, New York/Boston, 2006)
39. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, A unified modeling and solution framework for combinatorial optimization problems. OR Spectr. **26**, 237–250 (2004)
40. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, Solving combinatorial optimization problems via reformulation and adaptive memory metaheuristics, in *Revolutionary Visions in Evolutionary Computation*, ed. by A. Menon, D. Goldberg (Kluwer, Boston, MA, 2004)
41. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, An unconstrained quadratic binary approach to the vertex coloring problem. Ann. OR **139**, 229–241 (2005)
42. G. Kochenberger, F. Glover, B, Alidaee, K. Lewis, Using the unconstrained quadratic program to model and solve max 2-Sat problems. Int. J. Oper. Res. **1**(1 and 2), 89–100 (2005)
43. G. Kochenberger, F. Glover, B. Alidaee, H. Wang, Clustering of microarray data via clique partitioning. J. Comb. Optim. **10**, 77–92 (2005)
44. G. Kochenberger, B. Alidaee, H. Wang, An effective modeling and solution approach for the generalized independent set problem. Optim. Lett. **1**, 111–117 (2007). Springer-Verlag
45. D.J. Laughunn, Quadratic binary programming. Oper. Res. **14**, 454–461 (1970)
46. M. Lewis, B. Alidaee, G. Kochenberger, Using xQx to model and solve the uncapacitated task allocation problem. OR Lett. **33**, 176–182 (2005)
47. M. Lewis, G. Kochenberger, B. Alidaee, A new modeling and solution approach for the set partitioning problem. Comput. OR **35**, 807–813 (2008)
48. M. Lewis, B. Alidaee, F. Glover, G. Kochenberger, A note on xQx as a modeling and solution framework for the linear ordering problem. Int. J. OR **5**(#2), 152–162 (2009)
49. A. Lodi, K. Allemand, T.M. Liebling, An evolutionary heuristic for quadratic 0-1 programming. Technical Report OR-97-12, D.E.I.S., University of Bologna, 1997
50. P. Merz, B. Freisleben, Genetic algorithms for binary quadratic programming, in *Proceedings of the 1999 International Genetic and Evolutionary Computation Conference (GECCO'99)* (Morgan Kaufmann, San Francisco, 1999), pp. 417–424
51. A. Mingozzi, M. Boschetti, S. Ricciardelli, L. Blanco, A set partitioning approach to the crew scheduling problem. Oper. Res. **47**(6) 873–888 (1999)
52. M. Padberg, On the facial structure of set packing polyhedra. Math. Program. **5**, 199–215 (1973)
53. G. Palubeckis, A heuristic-branch and bound algorithm for the unconstrained quadratic zero-one programming problem. Computing **54**, 284–301 (1995)
54. P. Pardalos, G.P. Rodgers, Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 131–144 (1990)
55. P. Pardalos, G.P. Rodgers, A branch and bound algorithm for maximum clique problem. Comput. OR **19**, 363–375 (1992)
56. P. Pardalos, J. Xue, The maximum clique problem. J. Glob. Optim. **4**, 301–328 (1994)
57. A. Pekec, M. Rothkopf, Combinatorial auction design. Manage. Sci. **49**(#11), 1485–1503 (2003)
58. M. Ronnqvist, Optimization in forestry. Math. Program. B **97**, 267–284 (2003)
59. L. Schrage, *Optimization Modeling with LINDO* (Duxbury, Pacific Grove, 1997)
60. R. Vemuganti, Applications of set covering, set packing and set partitioning models: a survey, in *Handbook of Combinatorial Optimization*, ed. by D. Zhu, P. Pardalos (Kluwer Academic, Boston, MA, 1998)
61. H. Wang, B. Alidaee, F. Glover, G. Kochenberger, Solving group technology problems via clique partitioning. Int. J. Flex. Manuf. **18**, 77–97 (2006)
62. H. Wang, T. Obremski, B. Alidaee, G. Kochenberger, Clique partitioning for clustering: a comparison with K-Means and latent class analysis. Commun. Stat. **37**(1), 1–13 (2008)