A Unified Framework for Modeling and Solving Combinatorial Optimization Problems: A Tutorial*

Gary A. Kochenberger[1] and Fred Glover[2]

[1] School of Business, University of Colorado at Denver, Denver, Colorado 80217, USA. Gary.Kochenberger@cudenver.edu
[2] School of Business, University of Colorado at Boulder, Boulder, Colorado 80304, USA. Fred.Glover@Colorado.edu

Summary. 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. This tutorial gives an introduction to this evolving area. The methodology is illustrated by several examples and substantial computational experience demonstrating the viability and robustness of the approach.

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

$$\text{UQP}: \text{ Opt } xQx$$

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 sixties (see for instance Hammer and Rudeanu [HR68]) with applications reported in such diverse areas as spin glasses [DDJMRR95, GJR88], machine scheduling [AKA94], the prediction of epileptic seizures [ISSP00], solving satisfiability problems [BH02, BP89, HR68, HJ90], and determining maximum cliques [BH02, PR92, PX94]. The application potential of UQP is much greater than might be imagined, due to the re-formulation 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 re-cast into the form of UQP via the use of

* Earlier versions of this material appear in references [KGAR04a, KGAR04b]

such penalties. This process of re-formulating a given combinatorial problem into an instance of UQP is easy to carry out, enabling UQP to serve as a common model form for a widely diverse set of combinatorial models. This common modeling framework, coupled with recently reported advances in solution methods for UQP, help to make the model a viable alternative to more traditional combinatorial optimization models as illustrated in the sections that follow.

1.1 Re-casting 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[(3)]

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

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 evidently 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 our transformations involving UQP, we say that a penalty function is 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):

Classical Constraint	Equivalent Penalty (VIP)
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$

(3) The degree-2 constraints of this form commonly appear in optimization problems pertaining to graphs as described in [BH02, PR92, PX94]. As we'll see later in this paper, however, their application extends far beyond classical graph problems.

The penalty term in each case is zero if the associated constraint is violated, and otherwise 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 section 1.3. Before moving on to this more general case, however, we give a complete illustration of the re-casting process by considering the set packing problem.

1.2 Set Packing

Set packing problems (SPP) are important in the field of combinatorial optimization due to their application potential and their computational challenge.

The standard formulation for SPP is:

$$\text{SPP: } \max \sum_{j=1}^{n} w_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \text{for } i = 1, ..., m$$

$$x \text{ binary}$$

where the a_{ij} are 0/1 coefficients and the w_j are positive weights. The number of constraints m is determined by the application, and generally may be very large. Many important applications of SPP have been reported in the literature, and an extensive survey of set packing and related models may be found in Vemuganti [Vem98]. The recent paper by Delorme, Gandibleax, and Rodriguez [DGR04] reports applications in railway infrastructure design, ship scheduling, resource constrained project scheduling, and the ground holding problem. Applications in combinatorial auctions and forestry are reported by Pekec and Rothkopf [PR03] and Ronnqvist [Ron03], respectively. Other applications, particularly as part of larger models, are found throughout the literature.

Since SPP is known to be NP-hard, exact methods generally cannot be relied upon to generate good solutions in a timely manner. In particular, the linear programming relaxation does not provide good bounds for these difficult problems. Nonetheless, considerable work has been devoted to improving exact methods for SPP with innovations in branch & cut methods based on polyhedral facets as described in Padberg [Pad73] and the extensive work of Cornuejolos [Cor95]. Despite these advances, however, SPP remains resistant to exact methods and, in general, it is necessary to employ heuristic methods to obtain solutions of reasonably decent quality within a reasonable amount of time. This is particularly true for problem instances with a large number of variables that are neither loosely nor tightly constrained.

Recasting SPP into the form of xQx:

The structure of the constraints in SPP enables quadratic VIPs to be easily constructed for each constraint simply by summing all products of constraint variables taken two at a time. To illustrate, consider the constraint

$$x_1 + x_2 + x_3 \leq 1$$

Such a constraint can be replaced by the quadratic penalty function

$$P\,(x_1x_2 + x_1x_3 + x_2x_3)$$

where P is a positive scalar. Clearly this quadratic penalty function is zero for feasible solutions and positive otherwise. Similarly, the general packing (or GUB) constraint

$$\sum_{j=1}^{n} x_j \leq 1$$

can be replaced by the penalty function

$$P(\sum_{i=1}^{n-1} x_i \sum_{j=i+1}^{n} x_j).$$

By subtracting such penalty functions from the objective function of a maximization problem, we have a model in the general, unified form of xQx. Note that this reformulation is accomplished without introducing new variables. This procedure is illustrated by the following two examples:

Example 1: Find binary variables that solve:

$$\text{SPP: } \max x_1 + x_2 + x_3 + x_4$$

s.t.

$$\begin{aligned} x_1 + x_3 + x_4 &\leq 1 \\ x_1 + x_2 &\leq 1 \end{aligned}$$

Representing the scalar penalty P by 2M, the equivalent unconstrained problem is:

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

which can be re-written as

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

where Q, as shown above, is a square, symmetric matrix. All the problems characteristics of SPP are embedded into the Q matrix.

Example 2: (Schrage [Sch97]):

$$\max \sum_{j=1}^{22} x_j$$

s.t.

$$\begin{aligned}
&x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 1\\
&x_1 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} \leq 1\\
&x_2 + x_8 + x_{15} + x_{16} + x_{17} + x_{18} \leq 1\\
&x_3 + x_9 + x_{15} + x_{19} + x_{20} + x_{21} \leq 1\\
&x_4 + x_{10} + x_{16} + x_{19} + x_{22} \leq 1\\
&x_5 + x_{11} + x_{17} + x_{20} + x_{21} + x_{22} \leq 1\\
&x_6 + x_{12} + x_{13} + x_{18} + x_{20} + x_{22} \leq 1\\
&x_7 + x_{12} + x_{14} + x_{18} + x_{21} + x_{22} \leq 1\\
&x_{13} + x_{14} + x_{21} \leq 1
\end{aligned}$$

The Q matrix for equivalent transformed model ($\max xQx$), with M arbitrarily chosen to be 8, is given by:

$$\begin{bmatrix}
1 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-8 & 1 & -8 & -8 & -8 & -8 & -8 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & -8 & -8 & -8 & 0 & 0 & 0 & 0 & 0\\
-8 & -8 & 1 & -8 & -8 & -8 & -8 & 0 & -8 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & -8 & -8 & 0\\
-8 & -8 & -8 & 1 & -8 & -8 & -8 & 0 & 0 & -8 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & -8 & 0 & 0 & -8\\
-8 & -8 & -8 & -8 & 1 & -8 & -8 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & -8 & -8 & -8\\
-8 & -8 & -8 & -8 & -8 & 1 & -8 & 0 & 0 & 0 & 0 & -8 & -8 & 0 & 0 & 0 & 0 & -8 & 0 & -8 & 0 & -8\\
-8 & -8 & -8 & -8 & -8 & -8 & 1 & 0 & 0 & 0 & 0 & -8 & 0 & -8 & 0 & 0 & 0 & -8 & 0 & 0 & -8 & -8\\
-8 & -8 & 0 & 0 & 0 & 0 & 0 & 1 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & 0 & 0 & 0 & 0\\
-8 & 0 & -8 & 0 & 0 & 0 & 0 & -8 & 1 & -8 & -8 & -8 & -8 & -8 & -8 & 0 & 0 & 0 & -8 & -8 & -8 & 0\\
-8 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & -8 & 1 & -8 & -8 & -8 & -8 & 0 & -8 & 0 & 0 & -8 & 0 & 0 & -8\\
-8 & 0 & 0 & 0 & -8 & 0 & 0 & -8 & -8 & -8 & 1 & -8 & -8 & -8 & 0 & 0 & -8 & 0 & 0 & -8 & -8 & -8\\
-8 & 0 & 0 & 0 & 0 & -8 & -8 & -8 & -8 & -8 & -8 & 1 & -8 & -8 & 0 & 0 & 0 & -8 & 0 & -8 & -8 & -8\\
-8 & 0 & 0 & 0 & 0 & -8 & 0 & -8 & -8 & -8 & -8 & -8 & 1 & -8 & 0 & 0 & 0 & -8 & 0 & -8 & -8 & -8\\
-8 & 0 & 0 & 0 & 0 & 0 & -8 & -8 & -8 & -8 & -8 & -8 & -8 & 1 & 0 & 0 & 0 & -8 & 0 & 0 & -8 & -8\\
0 & -8 & -8 & 0 & 0 & 0 & 0 & -8 & -8 & 0 & 0 & 0 & 0 & 0 & 1 & -8 & -8 & -8 & -8 & -8 & -8 & 0\\
0 & -8 & 0 & -8 & 0 & 0 & 0 & -8 & 0 & -8 & 0 & 0 & 0 & 0 & -8 & 1 & -8 & -8 & -8 & 0 & 0 & -8\\
0 & -8 & 0 & 0 & -8 & 0 & 0 & -8 & 0 & 0 & -8 & 0 & 0 & 0 & -8 & -8 & 1 & -8 & 0 & -8 & -8 & -8\\
0 & -8 & 0 & 0 & 0 & -8 & -8 & -8 & 0 & 0 & 0 & -8 & -8 & -8 & -8 & -8 & -8 & 1 & 0 & -8 & -8 & -8\\
0 & 0 & -8 & -8 & 0 & 0 & 0 & 0 & -8 & -8 & 0 & 0 & 0 & 0 & -8 & -8 & 0 & 0 & 1 & -8 & -8 & -8\\
0 & 0 & -8 & 0 & -8 & -8 & 0 & 0 & -8 & 0 & -8 & -8 & -8 & 0 & -8 & 0 & -8 & -8 & -8 & 1 & -8 & -8\\
0 & 0 & -8 & 0 & -8 & 0 & -8 & 0 & -8 & 0 & -8 & -8 & -8 & -8 & -8 & 0 & -8 & -8 & -8 & -8 & 1 & -8\\
0 & 0 & 0 & -8 & -8 & -8 & -8 & 0 & 0 & -8 & -8 & -8 & -8 & -8 & 0 & -8 & -8 & -8 & -8 & -8 & -8 & 1
\end{bmatrix}$$

Solving[4] this instance of xQx gives an optimal solution with an objective function value of 4 and $x_7 = x_{13} = x_{17} = x_{19} = 1$, all other variables equal to zero.

We conclude this section by summarizing some of the key points about the procedure illustrated above:

1. In the manner illustrated, any SPP problem can be re-cast into an equivalent instance of UQP.

[4] All instances of UQP solved in this tutorial were solved using the tabu search method described in [GKAA99, GKA98].

2. This reformulation is accomplished without the introduction of new variables.
3. It is always possible to choose the scalar penalty sufficiently large so that the solution to xQx is feasible for SPP. At optimality the two problems are equivalent in the sense that they have the same set of optimal solutions.
4. For "weighted" instances of SPP, the weights, w_j, show up on the main diagonal of Q.

We subsequently describe the outcome of using this and other types of problem reformulations as a means for solving a variety of optimization models.

1.3 Accommodating General Linear Constraints

The preceding section illustrated how to re-cast a constrained problem into the form of UQP when the VIPs were known in advance. In this section we indicate how to proceed in the more general case when VIPs are not known in advance. We take as our starting point the general constrained problem

$$\begin{array}{l} \min x_0 = xQx \\ \text{s.t. } Ax = b, \;\; x \text{ binary} \end{array} \tag{3}$$

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). Problems with inequality constraints can also be put into 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, we have

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

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 our constrained problem becomes

$$\text{UQP(PEN)}\colon \min x\hat{Q}x, \;\; x \text{ binary} \tag{5}$$

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

We refer to the preceding general transformation that takes us from (3) through (4) to (5) as transformation #1. This approach along with related material can be found in [BH02, Han79, HJM93]. This is the general procedure

that could in principle be employed to transform any problem in the form of (7) into an equivalent instance of UQP. As indicated earlier in section 1.1, VIPs are known in advance for certain simple constraints and when such constraints are encountered it is usually preferred to use the known VIP directly rather than applying transformation #1. One special constraint in particular

$$x_j + x_k \leq 1$$

appears in many important applications and as indicated in section 1.1 can be handled by a VIP of the form Px_jx_k. Due to the importance of this constraint and its frequency of occurrence in applications, we refer to this special case as transformation # 2. The use of these two transformations is illustrated in the next section by considering two classical problems in combinatorial optimization.

2 Further Illustrative Examples

Before highlighting some of the problem classes we have successfully solved using the foregoing transformation approaches, we give two small examples from classical NP-hard problem settings to provide additional concrete illustrations.

Example 1: Set Partitioning.

The classical set partitioning problem is found in applications that range from vehicle routing to crew scheduling [Jos02, MBRB99]. 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:

$$\text{UQP(PEN)} : \min x\hat{Q}x, \ x \text{ binary}$$

where the additive constant, c, is 40 and

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

Solving UQP(PEN) we obtain an optimal solution $x_1 = x_5 = 1$ (all other variables equal to 0) for which $x_0 = 6$. In the straightforward application of Transformation 1 to this example, the replacement of the original problem formulation by the UQP(PEN) model did not involve the introduction of new variables. In many applications, Transformation 1 and Transformation 2 can be used in concert to produce an equivalent UQP model, as demonstrated next.

Example 2*:* 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, we have

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

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

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

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

This problem can be re-cast into 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 model of (6) and (7) has no explicit objective function, any positive value for the penalty P will do. The following example gives a concrete illustration of the re-formulation process.

Consider the graph given in Figure 1 and assume we want find a feasible coloring of the nodes using 3 colors.

Our satisfiablity problem is that of finding a solution to:

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

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

(for all adjacent nodes i and j)

In this traditional form, the model has 15 variables and 26 constraints. To recast this problem into the form of UQP, we use Transformation 1 on the

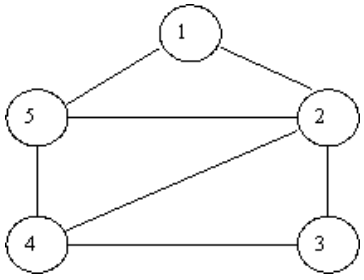

Fig. 1. Example of a graph for **the K-Coloring Problem**.

equations of (8) and Transformation 2 on the inequalities of (9). Arbitrarily choosing the penalty P to be 4, we get the equivalent problem:

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

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. Later in this paper we present some computational results for several standard K-coloring problems. An extensive presentation of the xQx approach to a variety of coloring problems, including a generalization of the K-coloring problem considered here, is given in Kochenberger, Glover, Alidaee and Rego [KGAR02].

3 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 [AHA98, AAK99, Bea99, BS94, BHS89, CS94, GARK02, GKAA99, GKA98, KTN00, Lau70, LAL97, MF99, Pau95, PR90] for a description of some of the more successful methods. The pursuit of further advances in solution methods for xQx remains an active research arena.

In the work reported here, we used a basic tabu search method due to Glover, Kochenberger, and Alidaee [GL97, GKAA99, GKA98]. A brief overview of the approach is given below. For complete details the reader is referred to the aforementioned reference.

Our TS method 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 we call "add moves") and destructive phases that progressively set variables to 0 (whose steps we call "drops moves"). To control the underlying search process, we use a memory structure that is updated at c*ritical 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. We begin with *span* equal to 1 and gradually increase it to 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 pre-set 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.

4 Applications

To date several important classes of combinatorial problems have been successfully modeled and solved by employing the unified framework. Our results

with the unified framework applied to these problems have been uniformly attractive in terms of both solution quality and computation times. While our solution method is designed for the completely general form of UQP, without any specialization to take advantage of particular types of problems reformulated in this general representation, our outcomes have typically proved competitive with or even superior to those of specialized methods designed for the specific problem structure at hand. Our broad base of experience with UQP as a modeling and solution framework 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
Linear Ordering Problems
Number Partitioning Problems.

Additional test problems representing a variety of other applications (which do not have "well-known" names) have also been reformulated and solved via UQP. In the section below we report specific computational experience with some of the problem classes listed above Additional applications are discussed by Boris and Hammer [BH91] and Lewis, Alidaee and Kochenberger [LAK04].

5 Illustrative Computational Experience

Sections 1 and 2 of this paper presented small examples intended to illustrate the mechanics of the transformation process. Here, we highlight our computational experience with several well-known problem classes. In each case, we specify the standard problem formulation, comment on the transformation(s)

used to recast the problem into the form of UQP, and summarize our computation experience.

It is not our objective here to provide a comprehensive comparison with the best known methods for the problem classes considered below. Rather, our purpose in this section is to provide additional validation of the potential merits of this unified approach. Nonetheless, the results shown below are very attractive and motivate such head-to-head comparisons in future studies.

5.1 Warehouse Location: (Single source, Uncapacitated)

Zero/One formulation:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$
$$\sum_{i=1}^{m} x_{ij} = 1 \;\; j = 1, \ldots, n$$
$$x_{ij} \leq y_i \quad \forall\, (i, j)$$
$$x, y \;\; \text{binary}$$

Recast as xQx:

- Complement the y variables (to enable the use of transformation # 2)
- Use both transformations
- No new variables required

Computational Experience:

- **Total number of Problems Solved: 4**

# variables	m	n	# TS cycles	Soln Time (sec)	Soln Optimal?
55	5	10	20	$<$ 1 sec	Yes
210	10	20	50	< 5	*
410	10	40	100	< 30	*
820	20	40	100	< 120	*

***** Optimal Solutions not known.

Remarks:

Transformation # 1 was used for the assignment constraints and transformation #2, once the "y" variables were complemented, was used for the variable upper bound constraints. No new variables were required. The problems were randomly generated with $c_{ij} = U(50, 100)$ and $f_i = U(100, 200)$. Each instance was recast as xQx using a penalty, P, equal to 200. Our tabu search heuristic was allowed to run for a fixed number of oscillation cycles as shown above, with the largest problem taking less than 2 minutes on a Pentium 400 PC. In each case feasible solutions were easily found. Moreover, the solution found for the first problem proved to be optimal. Optimal solutions to the other problems are not known.

5.2 Constraint Satisifiability problems (CSPs):

Zero/One formulation:

$$Ax = b \quad a_{ij} \in \{-1, 0, 1\}$$
$$b_i = 1 \text{ or } 2$$

Recast as xQx:

- Transformation #1
- No additional variables

Computational Experience:

- **Total number of Problems Solved: 26**

# variables	# rows	# problems	Soln Time	Soln Feasible
20	6	3	$<$ 1 sec	Yes
50	10	3	$<$ 3 sec	Yes
100	30	10	$<$ 15 sec	Yes
500	50	5	1 – 2 min	Yes
1000	100	5	4 – 5 min	Yes

Remarks:

Transformation # 1, with P taken to be 2, was used to develop the equivalent xQx model for each of these problems. No new variables were required. In all, a total of 26 random problem instances were solved by letting our tabu search heuristic run until the objective function was equal to zero, implying a feasible (and in this case, optimal) solution. Feasible solutions were quickly found in each case, with solutions for the largest instances found, on average, in roughly 4-5 minutes on a Pentium 200 PC. The smaller problems took only a few seconds.

5.3 Quadratic Knapsack Problems

Zero/One Formulation:

$$\begin{aligned} &\max \quad xQx \\ &\text{s.t. } Ax \leq b, \;\; x \text{ binary} \end{aligned}$$

Recast as xQx:

- Add slack variables
- Use transformation #1

Computational Experience:

- **Total number of Problems Solved: 53**

# Variables	# constraints	# problems	Soln times	Optimal Solns?
10,20 30	1	24	< 2 sec	23 proven opt
40,100,500	1	20	4, 9, 240 sec	*
20	2,4	8	< 4 sec	All opt
50	5	1	< 16 sec	*

* Optimal Solutions not Known

Remarks:

For this class of problems, a total of 53 random problems were solved. The problem instances ranged in size from 10 to 500 variables and 1 to 5 constraints. The largest of these problems are much larger (in terms of both variable and constraint count) than previously reported in the literature. Instances were constructed with $q_{ij} = U(-25, 25)$, $a_{ij} = U(1, 10)$, and b_i chosen to be a fraction of the sum of the $a_{ij}$for a given row. Slack variables (in binary expansion form) allowing for a maximum slack activity of 31 were added to each row to produce equality constraints and transformation # 1 was then used to produce the equivalent xQx representation. The variable counts given in the table above portray the original problems and do not include these slack variables.

The value of the penalty P used to achieve an equivalent xQx representation was heuristically incremented as needed in order to achieve feasible solutions for the larger instances solved. For the largest of the problems, n = 500, xQx was first generated with P = 150. Solving this model gave a solution that was infeasible with respect to the original problem. P was then raised to 1500 and a new xQx instance was formed whose solution was feasible. P = 1500 was then used in the transformation of each of the other (smaller) problems and in each case the solutions generated proved to be feasible. Moreover, the solutions obtained for the 23 of the 24 problems of size n = 30 or less proved to be optimal. Each problem was allowed to run for 100 TS cycles. The largest of the problems required 4 minutes on a Pentium 200 PC; all others were solved in less than 16 seconds with the smallest problems taking less than 2 seconds.

Many of the smaller problems (n = 30 or less) were solved again with a much smaller value of the penalty. P = 50 was large enough to produce optimal solutions to the n = 10 and n = 20 variable problems and P = 250 was large enough for the n = 30 problems. Computational times, as expected, remained very small, showing no apparent relationship to the penalty value.

5.4 Maximum Diversity

Zero/One Formulation:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}\, x_i\, x_j$$
$$\text{s.t.} \sum_{i=1}^{n} x_i = m$$

Recast as xQx:

- Transformation #1
- No new variables

Computational Experience:

- **Total number of Problems Solved: 25**

# vars (n)	M	# TS cycles	Soln Time	Solns Opt?
100	10, 15, 20, 25, 30	20	2 sec (each)	*
300	30, 45, 60, 75, 90	50	15 sec (each)	*
500	50, 75, 100, 125, 150	100	58 sec (each)	*
1000	100, 150, 200, 250, 300	100	194 sec (each)	*
2000	200, 300, 400, 500, 600	200	16 min (each)	*

* Optimal solutions not known

Remarks:

For this class of problems we solved a total of 25 random instances of sizes ranging from 100 to 2000 variables. For each size, five different values of "m" were considered as shown in the table above. For all problems, the q_{ij}values were chosen from U(-50,50). Transformation #1 was used with P = 2n. Our tabu search heuristic was run for a fixed number of cycles, terminating in each case with a feasible solution. Optimal solutions for these problems are not known. However, we have also solved much smaller problems for which optimal solutions are known and in each such case our approach was successful in finding the optimal solution. All runs were made on a Pentium 200 PC.

Prior to this paper, the largest instances reported in the literature were of size n = 100. Our results greatly expand the state of the art and illustrate a solution capability much beyond that reported by others.

5.5 Set Partitioning

Zero/One Formulation:

$$\min \sum_{j=1}^{n} c_j x_j$$
$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j = 1 \quad i = 1, ..., m$$
$$x \text{ binary}$$

Recast as xQx:

- Transformation #1
- No additional variables

Computational Experience:

- **Total number of Problems Solved: 18**

n	M	# TS cycles	# problems	Soln time	Soln Feas*
80	10	20	5	3 sec	Yes
100	10	20	5	9 sec	Yes
400	40	50	5	220 sec	Yes
800	80	100	3	5 min	Yes

* Optimal Solutions not known

Remarks:

For this class of problems, we solved a total of 18 random instances or sizes ranging from 80 variables and 10 constraints to 800 variables and 80 constraints. The problems varied in density from .1 to .3 and in each case the $c_j$values were chosen from U(10,100). Transformation #1 was used to convert to xQx with P = max $c_j$value for each problem. Our Tabu Search heuristic was run for fixed number of cycles as shown in the table above. While optimal solutions are not known, feasible solutions were quickly found for each problem with the largest of the problems being solved in less than 5 minutes on a Pentium 200 PC.

5.6 Vertex Coloring

Zero/One Formulation:

$$\begin{aligned}
&\min \sum_{j=1}^{nc} y_j \\
&\text{s.t. } \sum_{j=1}^{nc} x_{ij} = 1 \quad i = 1, \ldots, nv \\
&x_{ik} + x_{jk} \leq 1 \text{ for each edge } (i,j) \text{ and color } k \\
&x_{ik} \leq y_k \text{ for each vertex } i \text{ and each color } k \\
&x, y \text{ binary}
\end{aligned}$$

where:
nc = Maximum number of colors allowed
nv = number of vertices in graph

Recast as xQx:

- complement y variables
- Transformation # 1 for first constraints
- Transformation #2 for last two sets

- No additional variables

Computational Experience:

- **Total number of Problems Solved: 9**

ID	# nodes	# edges	nc	# xQx variables	Soln Time	xQx solution	Opt Solution
Jean_col	80	254	12	972	$<$ 2 min	10	10
David_col	87	406	12	1056	$<$ 2 min	11	11
Huck_col	74	301	14	1050	$<$ 2 min	11	11
Mycie13_col	11	20	8	96	$<$ 1 min	4	4
Mycie14_col	23	71	10	240	$<$ 1 min	5	5
Mycie15_col	47	236	10	480	$<$ 1 min	6	6
Mycie16_col	95	755	10	960	$<$ 2 min	7	7
Queen5_5_col	25	160	10	260	$<$ 1 min	5	5
Queen6_6_col	36	290	10	370	$<$ 2 min	8	7

Remarks:

In section 2 we presented a small example of a 3-coloring problem. Here we consider a generalization of the vertex coloring problem where we want to find a coloring with the minimum number of colors rather than finding one with a given number of colors.

For this class of problems, we solved 9 standard problems from the literature, which can be found at http://mat.gsia.cmu.edu/COLOR/instances.html.

The conversion to xQx was achieved by using transformation # 1 on the first set of constraints and, after complementing the "y" variables, using transformation # 2 on the last two sets of constraints. In each case, the penalty P was taken to be 20. Note that no new variables are required.

Optimal solutions were found in 8 of the 9 cases. We were off by one color for problem Queen6_6. The solution times for the largest problem was slightly less than 2 minutes on a Pentium 200 PC.

5.7 Maximum Clique (Max Independent Set)

Given a graph, G, and its complement graph:

$$G = (V, E),\ \bar{G} = (V, \bar{E})$$

Zero/One Formulation:

$$\begin{aligned} &\max \sum_{j=1}^{n} x_j \\ &\text{s.t.} \\ &x_i + x_j \leq 1 \ \ \forall\, (i,j) \in \bar{E} \end{aligned}$$

Recast as xQx:

- Transformation #2
- No additional variables

Computational Experience: (Max Clique)

- **Total number of Problems Solved: 33**

ID	# nodes	# instances	xQx solns	Soln Time	Solns Optimal?
P-hat 300	300	3	8, 25, 36	$<$ 2 sec	Yes
P-hat 500	500	3	9, 36, 50	$<$ 2 sec	Yes
P-hat 700	700	3	11, 44, 62	$<$ 4 sec	Yes
P-hat 1000	1000	3	10, 46, 68	$<$ 15 sec	Yes
P-hat 1500	1500	3	12, 65, 94	$<$ 6 min	Yes
C-fat 200	200	3	12, 24, 58	$<$ 1 sec	Yes
C-fat 500	500	3	14, 26, 64	$<$ 1 sec	Yes
Brock 200	200	4	21,12,15,17	$<$ 4 sec	Yes
Brock 400	400	4	27,29,31,33	7 min	Yes
Brock 800	800	4	23,24,25,26	32 min	Yes

Remarks:

For this class of problems we solved 33 standard test problems from the literature.

These problems can be found at ftp://dimacs.rutgers.edu/pub/challenge. The conversion to xQx was achieved by using transformation # 2, taking the penalty P to be 2 in each case. No new variables are required.

Optimal solutions were found for all 33 problems. With few exceptions, these optimal solutions were found in only a few seconds by our tabu search method on a Pentium 333 PC. As noted in the table above, Brock 400, Brock 800 and P-hat 1500-1 took somewhat longer, requiring 6, 7 and 32 minutes respectively.

5.8 Comments on Computational Experience

The computational experience reported above is intended to demonstrate the viability and utility of the reformulation approach. We have successfully applied this approach to many other problem classes as well. While our intention 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 known methods at this time, we nonetheless emphasize that results presented in section 5 clearly establish that the reformulation approach not only works across a wide array of problem classes but works very well.

The test bed we used in this section, 168 problems in all, is a combination of new, randomly generated problems and widely used problems from the literature. Optimal solutions are known for 100 of the 168 problems. For each problem class, the instances considered are representative of the problems considered by other researchers. In some cases, most notably for the maximum diversity and quadratic knapsack problems, we considered problem instances much larger than previously addressed in the literature. In all cases, our method was able to quickly find feasible solutions. For the problems where optimal solutions are known, our method matched the optimal solution in 99 of the 100 instances. This performance lends support to the expectation that the solutions obtained for the other problems, where optimal solutions are not known, are of very high quality as well. Solution times for our approach across the board are very small.

6 Special Transformations

Transformation # 1 can in principle be used to transform *any* linear constraint in bounded integer variables into a quadratic penalty term and in fact this transformation is the general workhorse of this recasting approach. However, its use with general inequalities requires the introduction of additional variables and thus alternative transformations not requiring additional variables should be employed where possible. As we have indicated in the examples and computational outcomes given above, it is often possible to use a mixture of transformations in the same problem, constructing penalties with an eye toward avoiding the introduction of new variables where circumstances permit.

The most common example of such an alternative is transformation #2, which accommodates a frequently encountered but very special class of inequalities. Section 1.1 listed a few additional special cases for which VIPs are known. This list is by no means exhaustive. Many additional special cases, either for single constraints or groups of constraints, are waiting to be discovered. We illustrate the possibility of discovering important special cases by considering the classical problem of *linear ordering.*

Linear Ordering:

The linear ordering problem is defined by an n-by-n matrix of weights $C = \{ c_{ij}\}$ where the problem is to find a permutation, p, of columns (and rows) such that the sum of the weights on the upper triangular matrix is maximized. 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 refers to sectors. This problem can be modeled utilizing the decision variable: $x_{ij} = 1$ if sector

i goes before sector j in the permutation; 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:

$$\begin{aligned}
&\max \sum_{i<j} c_{ij}x_{ij} + \sum_{j<i} c_{ij}\,(1 - x_{ji}) \\
&\text{s.t.} \\
&x_{ij} + x_{jk} - x_{ik} \leq 1 \quad \forall\,(i,j,k):\ i < j < k \\
&x_{ij} + x_{jk} - x_{ik} \geq 0 \quad \forall\,(i,j,k):\ i < j < k \\
&x_{ij} \in \{0,1\}\ \forall\,(i,j):\ i < j
\end{aligned}$$

After introducing slack variables, this model could be recast into the form of UQP by employing general transformation #1. However, the above constraints allow a special quadratic penalty not involving new variables that is greatly preferable to the penalty derived from transformation # 1.

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 $x_{ij} = 1,\ x_{jk} = 1,\ x_{ik} = 0$ and $x_{ij} = 0,\ x_{jk} = 0,\ x_{ik} = 1$. It is easy to see that an exact quadratic penalty that precludes these same two solutions, while allowing the others, is given by:

$$P\,\{x_{ik} + x_{ij}x_{jk} - x_{ij}x_{ik} - x_{jk}x_{ik}\}$$

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. This approach is illustrated below by a small example.

Example:

Consider the 4 sector example with an initial permutation *p = (1,2,3,4)* and matrix:

$$\begin{bmatrix} 0 & 12 & 5 & 3 \\ 4 & 0 & 2 & 6 \\ 8 & 3 & 0 & 9 \\ 11 & 4 & 2 & 0 \end{bmatrix}$$

The IP formulation becomes:

$$\begin{aligned}
&\max x_0 = 32 + 8x_{12} - 3x_{13} - 8x_{14} - 1x_{23} + 2x_{24} + 7x_{34} \\
&\text{s.t.} \\
&x_{12} + x_{23} - x_{13} \leq 1 \qquad x_{12} + x_{23} - x_{13} \geq 0 \\
&x_{12} + x_{24} - x_{14} \leq 1 \qquad x_{12} + x_{24} - x_{14} \geq 0 \\
&x_{13} + x_{34} - x_{14} \leq 1 \qquad x_{13} + x_{34} - x_{14} \geq 0 \\
&x_{23} + x_{34} - x_{24} \leq 1 \qquad x_{23} + x_{34} - x_{24} \geq 0
\end{aligned}$$

Representing P by 2M, the equivalent xQx model is given by the 6x6 matrix:

$$Q = \begin{bmatrix} 8 & M & M & -M & -M & 0 \\ M & -3-2M & M & M & 0 & -M \\ M & M & -8-4M & 0 & M & M \\ -M & M & 0 & -1 & M & -M \\ -M & 0 & M & M & -2-2M & M \\ 0 & -M & M & -M & M & 7 \end{bmatrix}$$

Choosing the penalty, M, to be 10 and solving the problem:

$$\max xQx$$

yields the value 15 with x_{12} *and* $x_{34} = 1$ for which the corresponding permutation is $p = (3, 4, 1, 2)$ and the (original) objective function value is 15 + 32 = 47.

7 Summary

In this tutorial we have demonstrated how a variety of disparate combinatorial problems can be solved by first re-casting them into the common modeling framework of the unconstrained quadratic binary program. Once in this unified form, the problems can be solved effectively by adaptive memory tabu search metaheuristics or other recently developed solution approaches for UQP.

Our findings 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, the UQP domain appears to offer a partial exception. In forming UQP(PEN), we destroy any linearity that the original problem may have exhibited. Moreover, any exploitable structure that may have existed originally is "folded" into the $\hat{Q}$ matrix, and the general solution procedure we apply takes no advantage of it. Nonetheless, our solution outcomes have been remarkably successful, yielding results that rival the effectiveness of the best specialized methods.

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 our 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.

As additional research is conducted to provide enhanced methods for solving the UQP model, the approach of recasting diverse problems into this general framework will become even more attractive. At present, we are solving problems reformulated as UQP that have more than 50,000 variables in the quadratic representation. On-going research will further expand our ability to solve instances of UQP, further establishing this approach as a unified framework with noteworthy practical and theoretical merit.

8 Acknowledgements

The authors would like to acknowledge the contributions of their co-workers Drs. Bahram Alidaee, Cesar Rego, and Haibo Wang whose work contributed to some of the results appearing in this paper.

References

[AKA94] Alidaee, B, G. Kochenberger, and A. Ahmadian, "0-1 Quadratic Programming Approach for the Optimal Solution of Two Scheduling Problems," *International Journal of Systems Science*, 25, 401-408, 1994.

[AHA98] Alkhamis, T. M., M. Hasan, and M. A. Ahmed, "Simulated Annealing for the Unconstrained Binary Quadratic Pseudo-Boolean Function," *European Journal of Operational Research,* 108, (1998), 641-652.

[AAK99] Amini, M., B. Alidaee, G. Kochenberger, "A Scatter Search Approach to Unconstrained Quadratic Binary Programs," *New Methods in Optimization,* Cone, Dorigo, and Glover, eds., McGraw-Hill, 317-330, 1999.

[Bea99] Beasley, J. E.,"Heuristic Algorithms for the Unconstrained Binary Quadratic Programming Problem, Working Paper, Imperial College, 1999.

[BS94] Billionet, A. and A. Sutter, "Minimization of a Quadratic Pseudo-Boolean Function." *European Journal of OR,* 78 pp. 106-115, (1994).

[BH02] Boros, E. and P. Hammer, " Pseudo-Boolean Optimization," *Discrete Applied Mathematics,* 123(1-3), 155-225 (2002)

[BH91] Boros, E., and P. Hammer, "The Max-Cut Problem and Quadratic 0-1 Optimization, Polyhedral Aspects, Relaxations and Bounds," *Annals of OR*, 33,151-225 (1991)

[BHS89] Boros, E, P. Hammer, and X, Sun, "The DDT Method for Quadratic 0-1 Minimization," RUTCOR Research Center, RRR 39-89, 1989.

[BP89] Boros, E. and A. Prekopa, "Probabilistic Bounds and Algorithms for the Maximum Satisfiability Problem," *Annals of OR,* 21 (1989), pp. 109-126.

[BGLM94] Bourjolly, J.M., P. Gill, G. Laporte, and H. Mercure, "A Quadratic 0/1 Optimization Algorithm for the Maximum Clique and Stable Set Problems," Working paper, University of Montreal, (1994).

[CS94] Chardaire, P, and A. Sutter, "A Decomposition Method for Quadratic Zero-One Programming," *Management Science*, 41:4, 704-712, 1994.

[Cor95] Cornuejolos, G., *Combinatorial Optimization: Packing and Covering,* CBMS-NSF, SIAM, (2001).

[DDJMRR95] De Simone, C. M. Diehl, M. Junger, P. Mutzel, G. Reinelt, and G. Rinaldi, 'Exact Ground State4s of Ising Spin Glasses: New Experimental Results with a Branch and Cut Algorithm," *Journal of Statistical Physics,* 80, 487-496 (1995)

[DGR04] Delorme, X., X. Gandibleau, and J. Rodriques, "GRASP for Set packing," *EJOR,* 153, (2004), pp. 564-580.

[GARK02] Glover, F., B. Alidaee, C. Rego, and G. Kochenberger, "One-Pass Heuristics for Large-Scale Unconstrained Binary Quadratic Programs," *EJOR 137, pp. 272-287*, 2002.

[GL97] Glover, F, and M. Laguna, "Tabu Search," Kluwer Academic Publishers, 1997.

[GKAA99] Glover, F., G. Kochenberger, B. Alidaee, and 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,* (Eds.) S. Voss, S. Martello, I. Osman, and C. Roucairol. Kluwer Academic Publisher, Boston, 1999.

[GKA98] Glover, F., G. Kochenberger., and B. Alidaee, "Adaptive Memory Tabu Search for Binary Quadratic Programs," Management Science, 44:3, 336-345, 1998.

[GJR88] Grotschel, M., M. Junger, and G. Reinelt, "An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design," *Operations Research,* Vol.36, #3, May-June, (1988), pp. 493-513.

[HR68] Hammer, P., and S. Rudeanu, *Boolean Methods in Operations Research,* Springer-Verlag, New York, 1968.

[Han79] Hansen, P.B., "Methods of Nonlinear 0-1 Programming," *Annals Discrete Math,* vol. 5, pp.53-70, 1979.

[HJ90] Hansen, P. and B. Jaumard, "Algorithms for the Maximum Satisfiability Problem," *Computing*, 44, 279-303 (1990).

[HJM93] Hansen, P, B. Jaumard., and V. Mathon, "Constrained Nonlinear 0-1 Programming," *INFORMS Journal on Computing*, 5:2, 97-119, 1993.

[ISSP00] Iasemidis, L. D., D. S. Shiau, J.C. Sackellares, and P. Pardalos, "Transition to Epileptic Seizures: Optimization," *DIMACS Series in Discrete Math and Theoretical Computer Science,* Vol. 55, (2000), pp. 55-73.

[Jos02] Joseph, A. "A Concurrent Processing Framework for the Set Partitioning Problem," Computers & Operations Research, 29, 1375-1391, 2002.

[KTN00] Katayama, K., M. Tani, and 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, 2000.

[KGAR04a] Kochenberger, G., F. Glover, B. Alidaee, and C. Rego, "A Unified Modeling and Solution Framework for Combinatorial Optimization Problems," *OR Spectrum,* 26, pp. 237-250, (2004).

[KGAR04b] Kochenberger, G., F. Glover, B. Alidaee, and C. Rego, "Solving Combinatorial Optimization Problems via Reformulation and Adaptive Memory Metaheuristics," *Revolutionary Visions In Evolutionary Computation, ed.* Anil Menon and David Goldberg, Kluwer Publisher, (to appear in 2004)

[KGAR02] Kochenberger, G., F. Glover, B. Alidaee, and C. Rego, " An Unconstrained Quadratic Binary Programming Approach to the Vertex Coloring Problem," Working Paper, University of Colorado at Denver, 2002.

[Lau70] Laughunn, D.J, "Quadratic Binary programming," *Operations Research*, 14, 454-461, 1970.

[LAK04] Lewis, M., B. Alidaee, and G. Kochenberger, "Using xQx to Model and Solve the Uncapacitated Task Allocation Problem," To Appear in *OR Letters* (2004).

[LAL97] Lodi, A., K. Allemand, and T. M. Liebling, "An Evolutionary Heuristic for Quadratic 0-1 Programming," Technical Report OR-97-12, D.E.I.S., University of Bologna, 1997.

[MF99] Merz, P. and B. Freisleben, "Genetic Algorithms for Binary Quadratic Programming," *Proceedings of the 1999 International Genetic and Evolutionary Computation Conference (GECCO'99),* pp. 417-424, Morgan Kaufmann, 1999.

[MBRB99] Mingozzi, A., M. Boschetti, S. Ricciardelli and L. Blanco, "A Set Partitioning Approach to the Crew Scheduling Problem," Operations Research, 47 (6) 873- 888, 1999.

[Pad73] Padberg, M., "On the Facial Structure of set packing Polyhedra," *Mathematical Programmming,*vol 5, (1973), pp.199-215.

[Pau95] Palubeckis, G. "A Heuristic-Branch and Bound Algorithm for the Unconstrained Quadratic Zero-One Programming Problem," *Computing,* pp. 284-301, (1995).

[PR90] Pardalos, P, and G.P. Rodgers, "Computational Aspects of a Branch and Bound Algorithm for Quadratic Zero-one Programming," *Computing*, 45, 131-144, 1990.

[PR92] Pardalos, P, and G.P. Rodgers, "A Branch and Bound Algorithm for Maximum Clique problem," *Computers & OR*, 19, 363-375, 1992.

[PX94] Pardalos, P, and J. Xue, "The Maximum Clique Problem," *The Journal of Global Optimization,* 4, 301-328, 1994.

[PR03] Pekec, A., and M. Rothkopf, "Combinatorial Auction Design," *Management Science,* Vol. 49, #11, Nov.2003, pp. 1485-1503.

[Ron03] Ronnqvist, M., "Optimization in Forestry," *Math. Programming*, *Series B,* 97, (2003). Pp. 267-284.

[Sch97] Schrage, L., *Optimization Modeling with LINDO,* Duxbury Press, (1997)

[Vem98] Vemuganti, R, "Applications of Set Covering, Set Packing and Set Partitioning Models: A Survey," *Handbook of Combinatorial Optimization*, eds. D. Zhu and P. Pardalos, Kluwer Academic Publishers, (1998).