Using the unconstrained quadratic program to model and solve Max 2-SAT problems

Gary Kochenberger*

School of Business, Campus Box 165,
University of Colorado, Denver, 80217-3364, CO, USA
E-mail: Gary.Kochenberger@cudenver.edu
*Corresponding author

Fred Glover

Leeds School of Business, UCB 419
University of Colorado, Boulder, 80309 0419, CO, USA
E-mail: Fred.Glover@colorado.edu

Bahram Alidaee

MIS/POM Department, Hearin Center for Enterprise Science,
School of Business Administration,
The University of Mississippi, University, 38677, MS, USA
E-mail: Balidaee@bus.olemiss.edu

Karen Lewis

231 Holman, School of Business Administration,
University of Mississippi University, 38677, MS, USA
E-mail: Klewis@bus.olemiss.edu

Abstract: Satisfiability (SAT) and Max-SAT problems have been the object of considerable research effort over the past few decades. They remain a very important research area today due to their computational challenge and application importance. In this paper we investigate the use of penalty functions to recast SAT problems into the modelling framework offered by the unconstrained quadratic binary program. Computational experience is presented, illustrating how promising this approach is for Max 2-Sat problems.

Keywords: satisfiability; metaheuristics; Tabu search.

Reference to this paper should be made as follows: Kochenberger, G., Glover, F., Alidaee, B. and Lewis, K. (2005) 'Using the unconstrained quadratic program to model and solve Max 2-SAT problems', *Int. J. Operational Research*, Vol. 1, Nos. 1/2, pp.89–100.

Biographical notes: Gary Kochenberger's academic specialty concerns the Building, Testing, and Implementing Algorithms For Solving Resource Allocation Problems that rise in both the private and public sectors. In recent years, his focus has been on problems of a combinatorial nature. He has published four books and numerous research papers in such journals as

Copyright © 2005 Inderscience Enterprises Ltd.

Management Science, Mathematical Programming, Journal of Optimization Theory and Applications, Operation Research, Computers and Operations research, Naval Research Logistics Quarterly, Decision Sciences, Journal of the Operational Research Society, European Journal of OR, Interfaces, Operations Research Letters, Omega, and the Journal of the Production and Operations Management Society.

Fred Glover is the Comcast Chaired Professor in Systems Science at the University of Colorado, Boulder. He has authored or coauthored more than 340 published papers and seven books in the fields of mathematical optimisation, computer science and artificial intelligence, with particular emphasis on practical applications in industry and government. Professor Glover is the recipient of the von Neumann Theory Prize, a member of the National Academy of Engineering, and has received numerous other awards and honorary fellowships. He serves on the advisory boards of several organisations and is cofounder of OptTek Systems, Inc.

Bahram Alidaee received his BS degree from the University of Tehran, Iran, his MBA degree from the University of North Texas and his PhD degree from the University of Texas at Arlington, 1988. He is currently an Associate Professor of Operations Management and the Interim Director for the Hearin Center for Enterprise Science at the School of Business Administration, the University of Mississippi. His research interests include Heuristic Programming, Complex Systems, Game Theory and Cost Allocations. He has published in journals such as *Management Science*, *Transportation Science*, *Production and Operations Management*, *European Journal of Operational Research*, *Journal of Operational Research Society*, *IEEE Transactions on Systems*, *Man and Cybernetics*, *Operations Research Letter*, *Information Processing Letters*, *Applied Mathematics Letters*, and other journals. He is member of INFORMS, DSI, POMS, and APICS.

Karen Lewis has a BS in Electrical Engineering from the University of Kansas. She worked as an engineer in software development for General Dynamics and Lockheed Martin for ten years before getting an MS and PhD in Operations Research from Southern Methodist University. Since then, she has worked as a Research Professor and Adjunct Professor at the University of Mississippi. Her areas of specialisation include Network Flows and Heuristics for Integer Programming Problems.

1 Introduction

The importance of Max-Sat problems is well established in the literature on combinatorial optimisation and NP-hard problems. Scores of papers exist describing both applications and a variety of solution approaches ranging from methods grounded in AI methodologies to those emerging out of mathematical programming origins. An excellent survey giving an overview of both applications and solution methods in given by Du et al. (1997). A rich source of more recent papers and general information regarding activities of the SAT research community can be found at http://www.satlive.org.

A common feature of contemporary methods for solving Max-Sat problems is that they are specially crafted to take advantage of the underlying problem structure. That is, they are designed for the single purpose of solving one or more classes of Max-Sat problems, employing tailored tactics intended to exploit the specific mathematical structure generated by the logical relationships of satisfiability. Notable examples of such special purpose methods are provided by the recently published works of Lardeux et al. (2004), Smyth et al. (2003) and Yagiura and Ibaraki (2001). The use of special purpose methods for particular problem classes is well entrenched in the combinatorial optimisation community at large and has been generally reinforced by the attractive performance that typically results from such approaches.

In recent years however, considerable experience has demonstrated that a common modelling framework, given by the unconstrained quadratic program (*UQP*), can be employed to successfully model and solve a wide variety of combinatorial optimisation problems. *UQP* is simply defined by:

$$UQP: opt\ x^t Qx$$

where x is a vector of binary decision variables, Q is a symmetric n-by-n matrix and *opt* is shorthand for *optimise*; i.e., for *minimise* or *maximise*. The success of this approach, as surveyed in the paper by Kochenberger et al. (2004), serves notice that the general purpose framework offered by UQP can indeed compete with, and sometimes surpass in performance, methods that are tailored to exploit specific problem structure. Our purpose in this paper is to illustrate how this approach can be effectively employed to model and solve certain Max-Sat problems.

The transformation from a given mathematical structure into the unified framework of UQP is accomplished by imposing quadratic infeasibility penalties as an alternative to the explicit imposition of the original problem constraints. This approach can, in principle, be applied to any problem having linear constraints, bounded integer variables, and a linear or quadratic objective function. In this regard, we refer to a penalty function, $g(x)$ as being a *valid infeasibility penalty (VIP)* if $g(x)$ is zero when x is feasible and positive otherwise. For models with linear constraints, it is always possible to find *quadratic VIPs* and associated penalties P (positive for minimisation and negative for maximisation) such that the original xQx function can be replaced by $xQx + Pg(x)$, yielding a new unconstrained quadratic objective in which $g(x) = 0$ at optimality, if and only if the original problem has a feasible solution. Thus, the transformed problem (that takes $g(x)$ into the objective function) can itself be expressed in the same form as *UQP*, and can be solved by methods designed for this latter problem.

Generally, VIPs and associated penalties P will not be known and will have to be discovered. Such discovery is straightforward as outlined in Hammer and Rudeanu (1968), Hansen (1979), Hansen and Jaumard (1990), Hansen, et al. (1993), and Kochenberger, et al. (2004). For certain problem classes, however, VIPs are known in advance, making the recasting into the form of UQP even easier. The Max 2-Sat problem, as we demonstrate in the section below, is of this nature, and additionally has the convenient property that the value of P can be selected to equal one. We point out that the modelling approach we employ here was first proposed, via a slightly different development, by Hammer and Rudeanu (1968) and more recently in Hansen and Jaumard (1990) and Boros and Hammer (2002). These authors, however, did not undertake to offer compelling evidence that their modelling device could be a computationally viable foundation for solving Max 2-Sat problems. Here we re-visit the approach with the goal

of providing the computational experience needed to establish it as a viable alternative to solving Max 2-Sat problems by modern, special purpose methods, and to highlight yet another example of the robustness afforded by the UQP unified modelling framework.

2 Transforming MAX 2-Sat into UQP

To motivate the replacement of constraints with penalties and to provide a basis for a comparison later in the paper with CPLEX, we first present a standard 0/1 linear programming formulation for the Max-Sat problem. We take as a given, a collection of m clauses in CNF form involving logical variables $x_1, x_2, \ldots, x_n$. The SAT problem is to determine whether or not there exists a truth assignment for the logical variables such that all clauses are simultaneously satisfied. Failing this, we want to find the truth assignment that satisfies the maximum number of clauses. If we introduce for each clause j an additional binary variable y_j that takes the value one if the clause is satisfied and zero otherwise, we get the general IP model for the SAT and Max-Sat problem:

$$SAT\_IP: \max y_0 = \sum_{j=1}^{m} y_j \tag{1}$$

subject to

$$\sum_{i \in P_j} x_i + \sum_{i \in N_j} \bar{x}_i \geq y_j \quad j = 1, m \tag{2}$$

where the index sets P_j and N_j represent positive and negative (complemented) literals that appear in clause j. When solving this model results in assigning values of one to all y variables, we have a solution to the *SAT* problem. Barring this, we have a solution to the Max-Sat problem. Thus the above model represents both the SAT and Max-Sat problem and the model specialises in the natural way for the 2-Sat case where each clause (constraint) contains exactly two literals.

Transforming *SAT_IP* for the Max 2-Sat case into *UQP* can be accomplished *without* introducing the y variables by means of the following observation. For 2-Sat problems, clauses can have 0, 1, or 2 negations. Each case gives rise to a standard linear inequality and in turn, an associated *valid infeasibility penalty,* $g(x)$ as shown below:

The three possibilities are:

- No negations:

  Classical constraint: $x_i + x_j \geq 1$

  $g(x)$: $(1 - x_i - x_j + x_i x_j)$

- One negation:

  Classical constraint: $x_i + \bar{x}_j \geq 1$

  $g(x)$: $(x_j - x_i x_j)$

- Two negations:

  Classical constraint: $\bar{x}_i + \bar{x}_j \geq 1$

  $g(x)$: $(x_i x_j)$.

By examining the truth table for x_i and x_j it is easy to verify that the quadratic penalty function listed for each case is in fact a VIP.

The approach then is to sum the penalties to produce the unconstrained quadratic program:

$$SAT\_UQP: \min x_0 = \sum_{i=1}^{m} g(x)_i = C + x^t Q x \tag{3}$$

where $g(x)_i$ is the quadratic penalty associated with clause i and C is the additive constant that results from summing the various penalty functions. Note that an attractive feature of this model is that the size of *SAT_UQP* is independent of the number of clauses in the problem. That is, a Max 2-Sat instance with 100 variables and 100 clauses yields the same sized instance of *SAT_UQP* as a Max 2-Sat instance with 100 variables and 5000 clauses. Note also that in an optimal solution to *SAT_UQP*, $x_0 = 0$ implies we have a solution to the 2-Sat problem while $x_0 > 0$ implies we have a solution to the Max 2-Sat problem in which x_0 clauses are not satisfied.

Example: (due to Hansen and Jaumard (1990))

Consider the Max 2-SAT example given below with 4 variables and 12 clauses. To the right of each clause we have placed the associated quadratic VIP.

Clause	*#Clause*	*Quadratic penalty*
1	$x_1 \vee x_2$	$(1 - x_1 - x_2 + x_1x_2)$
2	$x_1 \vee \overline{x}_2$	$(x_2 - x_1x_2)$
3	$\overline{x}_1 \vee x_2$	$(x_1 - x_1x_2)$
4	$\overline{x}_1 \vee \overline{x}_2$	(x_1x_2)
5	$\overline{x}_1 \vee x_3$	$(x_1 - x_1x_3)$
6	$\overline{x}_2 \vee \overline{x}_3$	(x_1x_3)
7	$x_2 \vee \overline{x}_3$	$(x_3 - x_2x_3)$
8	$x_2 \vee x_4$	$(1 - x_2 - x_4 + x_2x_4)$
9	$\overline{x}_2 \vee x_3$	$(x_2 - x_2x_3)$
10	$\overline{x}_2 \vee \overline{x}_3$	(x_2x_3)
11	$x_3 \vee x_4$	$(1 - x_3 - x_4 + x_3x_4)$
12	$\overline{x}_3 \vee \overline{x}_4$	(x_3x_4)

Summing the penalties yields:

$$SAT\_UPQ \min x_0 = 3 + x_1 - 2x_4 - x_2x_3 + x_2x_4 + 2x_3x_4$$

or,

$$\min x_0 = 3 + x^t Q_x$$

where the matrix Q is given by

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & 0.5 \\ 0 & -0.5 & 0 & 1 \\ 0 & 0.5 & 1 & -2 \end{bmatrix}$$

Note that the *SAT_UQP* representation of this Max 2-Sat instance is an unconstrained quadratic model in four variables while the equivalent *SAT_IP* representation of equation (1) and equation (2) results in a model with 16 variables and 12 constraints. Solving this unconstrained quadratic binary program yields the solution $x_0 = 1$ at $x_1 = x_2 = x_3 = 0$, $x_4 = 1$, meaning 11 of the 12 clauses are satisfied.

3 Computational experience

To test the attractiveness of the *SAT_UQP* approach to Max 2-Sat problems, we solved a variety of test problems from the literature along with some new instances that we randomly generated.

3.1 New random problems

In Table 1 and so on, we report the results which we obtained from our newly generated test problems. These problems range in size from instances with 100 variables and 626 clauses to 1000 variables and 10,878 clauses. One instance had 500 variables and 22,883 clauses. To provide a benchmark of comparison, we solved these problems both in form *SAT_UQP* and *SAT_IP* where there problems in the later representation were solved via CPLEX 8.0. In all cases presented in this paper, the *SAT_UQP* instances were solved by a straightforward Tabu search method as described in Glover et al. (1998) and Glover et al. (1999) and for which an overview is provided in the appendix to this paper. Each instance was allowed to run for an arbitrary limit of 50 SPAN cycles. (The notion of a SPAN cycle appears in several of the following sections and the reader not familiar with the Tabu search method we use may want to read the appendix before proceeding further)

The first three columns of Table 1 and so on give the problem ID along with the number of variables and clauses. The next two columns pertain to results obtained from *SAT_UQP* where '# Viol' is the number of violated or unsatisfied clauses and 'time' is run-time in seconds. The next four columns give the CPLEX results associated with the *SAT_IP* formulation where '# iters' refers to LP interations, '# nodes' refers to branch and bound nodes, and 'time' is run-time. All runs pertaining to *SAT_UQP* were carried out on a 1.7 GHz PC while the *SAT_IP* instances were run on a Sun Enterprise 3500 with two 400 MHz processors.

Table 1 New random problems

			SAT_UQP		*SAT_IP*			
ID	*n*	*# clauses*	*# Viol*	*Time*	*# Viol*	*# iters*	*# nodes*	*Time*
2Sat 100a	100	626	53	1 sec	53	601598	21226	147 sec
100b	100	795	98	1 sec	99	44 mil	1.7 mil	10 hrs
100c	100	953	129	1 sec	130	78 mil	2.7 mil	10 hrs
2Sat 200a	200	1891	232	3 sec	251	72 mil	3.2 mil	10 hrs
200b	200	2067	284	3 sec	315	70 mil	2.7 mil	10 hrs
200c	200	1699	219	3 sec	233	74 mil	3.5 mil	10 hrs
2Sat 300a	300	5585	905	6 sec	1054	14 mil	1.2 mil	10 hrs
300b	300	5590	915	6 sec	992	28 mil	0.9 mil	10 hrs
300c	300	4791	717	6 sec		(NA)		
2Sat 400a	400	5762	888	10 sec	1043	10 mil	1.0 mil	10 hrs
400b	400	6506	1012	10 sec	1248	9 mil	1.1 mil	10 hrs
400c	400	6507	991	10 sec	1106	23 mil	1.1 mil	10 hrs
2Sat 500a	500	5370	742	17 sec	964	10 mil	1.2 mil	10 hrs
500b	500	5841	806	17 sec	1023	10 mil	1.0 mil	10 hrs
500c	500	6304	907	17 sec	1104	11 mil	1.1 mil	10 hrs
500d	500	22883	3566	17 sec		(NA)		
2Sat 1000a	1000	10878	1505	61 sec		(NA)		

Source: Available from the authors.

CPLEX was allowed to run to completion or ten hours, which ever occurred first. As shown in Table 1, these problems, modelled via *SAT_IP* give rise to large and rather weak LP relaxations. As a result, CPLEX was only able to solve the first problem to completion. Due to size and other considerations, CPLEX results were not available for problems 300c, 500d, and 1000a. All others terminated via the ten hour limit with the results shown in the table. Note that our Tabu search heuristic, applied to *SAT_UQP*, found the optimal solution to the first problem and quickly found much better solutions than CPLEX for all other problems, the largest of which took 61 seconds to execute the arbitrary limit of 50 SPAN cycles. The times given in the table for *SAT_UQP* are for the entire 50 SPAN cycles. Many of the best solutions were found in early cycles in much shorter computation times.

3.2 Publicly available problems (Borchers and Furman (1999))

Table 2 reports the results we obtained from applying *SAT_UQP* to a set of problems provided by Borchers and Furman (1999). These problems were reported in their paper where they presented their Max-Sat algorithm called 'Maxsat'. The first three columns of Table 2 and so on give the size of the instances along with the best known solutions. Columns 4 and 5 give, respectively, the results and times for *SAT_UQP* and the final two columns give the results reported by Borchers and Furman obtained by their code Maxsat. As before, all instances of *SAT_UQP* were solved by our Tabu search method with each instance allowed to run for an arbitrary limit of 50 SPAN cycles. The results

reported for Maxsat were obtained on an IBM RS/6000-590. All times are in seconds unless noted otherwise in the table.

Table 2 Test problems

n	*m*	*Best known solution*	*Sat_UQP solution*	*Sat_UQP time*	*Maxsat*[3] *time*	*Maxsat solution*
50	100	4	4	<1	4	0.4
50	150	8	8	<1	8	1.5
50	200	16	16	<1	16	116.2
50	250	22	22	<1	22	652.4
50	300	32	32	<1	32	8,763
50	350	41	41	<1	NA	>12 hr
50	400	45	45	<1	NA	>12 hr
50	450	63	63	<1	NA	>12 hr
50	500	66	66	<1	NA	>12 hr
100	200	5	5	<2	5	3.2
100	300	15	15	<2	15	13,770
100	400	29	29	<2	NA	>12 hr
100	500	44	44	<2	NA	>12 hr
100	600	?	65	<2	NA	>12 hr
150	300	4	4	<3	4	4.1
150	450	22	22	<3	NA	>12 hr
150	600	38	38	<3	NA	>12 hr

Source: Borchers and Furman (1999).

Borchers and Furman reported results only for those instances where their algorithm terminated with a natural completion of the search process, stopping otherwise via a time limit of approximately 12 hours with no result reported. These test problems are relatively modest in size giving rise to small instances of *SAT_UQP*. As a result, our Tabu search method was able to quickly match the results reported by Borchers and Furman where they give results. For the other problems, we quickly found solutions which serve as best known but whose optimality have not been established.

In viewing the results of Table 2 and so on, it is important to note that Maxsat is an exact method while the Tabu search method we apply to *SAT_UQP* is a heuristic procedure. Thus, one should compare the results and computation times with appropriate care, taking into account the burden of proving optimality. Nonetheless it is fair to conclude that the results obtained from the *SAT_UQP* approach to these test problems are very attractive.

3.3 Publicly available problems (Smyth et al. (2003))

Table 3 and so on reports the results we obtained by applying *SAT_UQP* to a set of test problems provided by Smyth et al. (2003). These problems are part of an extensive repository of test problems, optimisers and other information pertaining to SAT that is

maintained at the University of British Columbia. The first three columns of the table give the problem ID along with the problem size. Column #4 gives the best known solutions (as provided by Smyth et al. (2003)). The next two columns give our results along with the time taken to achieve these results.

Table 3 Test problems

ID	# *Variables*	# *Clauses*	*Best known*	*SAT_UQP*	*Time* (*seconds*)
RND200.1000.01	200	1000	85	85	<3
RND200.1000.02	200	1000	91	91	<3
RND200.1000.03	200	1000	90	90	<3
RND200.1000.04	200	1000	84	84	<3
RND200.1000.05	200	1000	85	85	<3
RND200.1000.06	200	1000	88	88	<3
RND200.1000.07	200	1000	85	85	<3
RND200.1000.08	200	1000	79	79	<3
RND200.1000.09	200	1000	88	88	<3
RND200.1000.010	200	1000	85	85	<3
RND200.2000.01	200	2000	271	271	<3
RND200.2000.02	200	2000	247	247	<3
RND200.2000.03	200	2000	253	253	<3
RND200.2000.04	200	2000	260	260	<3
RND200.2000.05	200	2000	255	255	<3
RND200.2000.06	200	2000	268	268	<3
RND200.2000.07	200	2000	263	263	<3
RND200.2000.08	200	2000	252	252	<3
RND200.2000.09	200	2000	253	253	<3
RND200.2000.01	200	2000	266	266	<3

Source: Smyth et al. (2003).

As in the case of the problems referenced earlier, we ran each instance of *SAT_UQP* for the problems of Table 3 and so on for a limit of 50 SPAN cycles. For each problem we were able to match the best known result by performing a run of less than three seconds on a 1.7 GHz PC. The times reported are for the entire 50 cycles although the best known solutions were found in nine or fewer cycles for each of the 20 problems. Consequently, the best known solutions were actually found in less than 1 second in each case. Times for individual problem instances are not available from the repository so that exact comparisons are not possible at this time. We should point out that the UBC repository has hundreds of test problems. We report here on the first ten problems from the two largest classes of Max 2-Sat instances that are made available. We also tested many of the smaller problems and for each case tried we quickly reproduced the best known solution.

3.4 Extensions to the 3-SAT case

The same general approach taken here in formulating *SAT_UQP* for the Max 2-Sat problem can be followed to produce a penalty function approach for the 3-Sat and Max 3-Sat case. The difference is that for 3-Sat there are four possible clause types (rather that three), each with a classical constraint and a corresponding VIP. However, each VIP is a cubic function rather than a quadratic as in the case of the 2-Sat problem. Nonetheless these cubic functions can be reduced to quadratic functions by a reduction due to Boros and Hammer (2002), allowing 3-Sat problems to be handled, in principle, via the *SAT_UQP* model as well. This approach, however, involves the introduction of additional variables. We are currently investigating whether the method may nevertheless be computationally attractive, even if not to the same degree as its counterpart for the 2-Sat case.

4 Summary and conclusions

In this paper we have demonstrated that the Max 2-Sat problem can be effectively modelled and solved by first recasting it as an unconstrained quadratic binary program (*UQP*). Performance comparisons with CPLEX 8.0 and with results reported in the literature for other methods confirm the attractiveness of the *UQP* approach. Across a sizeable set of test problems, this approach quickly reproduced best known solutions and in a number of Instances, established new best known solutions.

The results presented here were obtained using a generic Tabu search method for solving *SAT_UQP* with no specialisation for the structure of the problems attempted. This is in marked contrast with most other methods in the literature that are specifically crafted for Max Sat problems. Many of the recently reported specialised methods for solving Max-Sat problems are very effective and efficient. Our intent here is not to take them on head to head. Rather, our main goal is to establish that a general purpose heuristic intended for the generic instance of UQP can produce high quality solutions to Max 2-Sat problems very quickly. How well this approach stacks up to the specialised methods will perhaps be resolved in future studies. We are satisfied at this point that in this study we have presented sufficient computational evidence to establish that the approach we are taking not only works but that it works quite well.

We also point out that, while our illustrations are for the unweighted Max 2-Sat problem, it is straightforward to extend our approach to address the weighted version, in which each clause has a figure of merit (weight) indicating the relative desirability of satisfying that clause. Once cast in the form of *UQP*, the problem can be efficiently solved by modern metaheuristic solution methods for *UQP*.

Perhaps the most important aspect of the present study is that its results are consistent with the competitive results obtained by the UQP approach to other combinatorial problems as well (see, for instance Kochenberger et al. (2004, 2005)), and Lewis et al. (2004)). Our performance for Max 2-Sat problems serves to further emphasise the viable, robust nature of UQP as a fruitful modelling and solution framework for certain combinatorial optimisation problems. As new and improved optimisers become available for solving UQP, this unified framework will become an increasingly attractive approach for rapidly finding high-quality solutions to such problems.

Acknowledgement

We would like to thank the research teams of Borchers and Furman and that of Smyth, Hoos, and Stutzle for making their test problems available for this study.

References

Borchers, B. and Furman, J. (1999) 'A two-phase exact algorithm for Max-SAT and weighted Max SAT', *J. of Combinatorial Optimization*, Vol. 2, pp.299–306.

Boros, E. and Hammer, P. (2002) 'Pseudo-boolean optimization', *Discrete Applied Mathematics*, Vol. 123, Nos. 1–3, pp.155–225.

Du, D., Gu, J. and Pardalos, P. (1997) *Satisfiability Problem: Theory and Application,* DIMACS Series in Discrete Mathematics, American Mathematical Society, Vol. 35.

Glover, F., Kochenberger, G. and Alidaee, B. (1998) 'Adaptive memory Tabu search for binary quadratic programs', *Management Science*, Vol. 44, No. 3, pp.336–345.

Glover, F., Kochenberger, G., Alidaee, B. and Amini, M.M. (1999) 'Tabu search critical event memory: an enhanced application for binary quadratic programs', in Voss, S., Martello, S., Osman, I. and Roucairol. C. (Eds.): *Meta Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publisher, Boston.

Hammer, P. and Rudeanu, S. (1968) *Boolean Methods in Operations Research*, Springer-Verlag, New York.

Hansen, P. and Jaumard, B. (1990) 'Algorithms for the maximum satisfiability problem', *Computing*, Vol. 44, pp.279–303.

Hansen, P., Jaumard, B. and Mathon, V. (1993) 'Constrained nonlinear 0-1 programming', *INFORMS Journal on Computing*, Vol. 5, No. 2, pp.97–119.

Hansen, P.B. (1979) 'Methods of nonlinear 0-1 programming', *Annals Discrete Math*, Vol. 5, pp.53–70.

Kochenberger, G., Glover, F., Alidaee, B. and Rego, C. (2005) 'An unconstrained binary programming approach to the vertex coloring problem', *Annals of Operations Research*. (forthcoming).

Kochenberger, G.A., Glover, F., Alidaee, B. and Rego, C. (2004) 'A unified modeling and solution framework for combinatorial optimization problems', *OR Spectrum*, Vol. 26, pp.237–250.

Lardeux, F., Saubion, S. and Hao, J.K. (2004) 'Designing efficient recombinating operators for SAT problems', *Lecture Notes in Computer Science*, Springer, Vol. 2936, pp.103–114.

Lewis, M., Alidaee, B. and Kochenberger, G. (2004) 'Using xQx to model and solve the uncapacitated task allocation problem', *Operations Research Letters*, on-line at Science Direct.

Smyth, K., Hoos, H. and Stutzle, T. (2003) 'Iterated robust Tabu search for Max-Sat', *Proc. of the 16th Canadian Conference on AI*, Springer-Verlag, pp.129–144.

Yagiura, M. and Ibaraki, T. (2001) 'Efficient 2 and 3 flip neighborhood search algorithms for the Max SAT: experimental evaluation', *J. of Heuristics*, Vol. 7, pp.423–442.

Appendix: Overview of our Tabu search method for solving UQP

Our TS method for UQP is centred on the use of strategic oscillation, which constitutes one of the primary strategies of Tabu search. The variant of strategic oscillation we employ may be sketched in overview as follows.

The method alternates between constructive phases that progressively set variables to one (whose steps we call 'add moves') and destructive phases that progressively set variables to zero (whose steps we call 'drops moves'). To control the underlying search process, we use 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. We begin with *span* equal to one 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 one, a *complete span cycle* has been completed and the next cycle is launched.

Information stored at critical events is used to influence the search process by penalising 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. A complete description of the framework for the method is given in Glover et al. (1999).