

*Working Paper Series*

HEARIN CENTER  
FOR  
ENTERPRISE SCIENCE

HCES-09-03

Modeling and Solving the Task Allocation Problem as an Unconstrained  
Quadratic Binary Program

By

Mark Lewis  
Bahram Alidaee  
Gary Kochenberger



*The University of Mississippi*

Director, Keith Womer  
School of Business Administration  
The University of Mississippi  
Post Office Box 1848  
University, MS 38677-1848  
(662) 915-5820  
<http://hces.bus.olemiss.edu>

---

# Modeling and Solving the Task Allocation Problem as an Unconstrained Quadratic Binary Program

Mark Lewis<sup>a,†</sup>, Bahram Alidaee<sup>b</sup> and Gary Kochenberger<sup>c</sup>

*a* Hearin Center for Enterprise Science, School of Business, University of Mississippi, University, MS 38677, USA. [mlewis@bus.olemiss.edu](mailto:mlewis@bus.olemiss.edu)

*b* Hearin Center for Enterprise Science, School of Business, University of Mississippi, University, MS 38677, USA. [balidaee@bus.olemiss.edu](mailto:balidaee@bus.olemiss.edu)

*c* School of Business, University of Colorado, Denver, CO 80217, USA  
[Gary.kochenberger@cudenver.edu](mailto:Gary.kochenberger@cudenver.edu)

Latest Revision: December, 2003.

---

**Abstract** — Task allocation problems consist of assigning a set of tasks to one of several processors such that the associated inter-task communication and processing costs are minimized. Modeled as a nonlinear integer program, these problems are computationally challenging. This paper presents a new modeling and solution approach capable of efficiently addressing large instances of the task allocation problem. Computational experience and a comparison to the state-of-the-art commercial code (CPLEX) illustrates the attractiveness of our approach.

---

**Keywords:** Task allocation, integer programming, metaheuristics.

---

<sup>†</sup> This research was supported in part by Office of Naval Research grant N000140310621.

## 1. Introduction

The future of information technology is clearly directed towards taking advantage of the extraordinary increases in network connectivity and bandwidth. The term super computer system is now synonymous with massively parallel processing. The trend in database systems is towards distributed databases, and the internet provides many opportunities for client/server applications to exploit distributed server processing of client requests. As information systems strive to find efficient methods of utilizing the ready availability of larger and more powerful distributed and parallel systems, it is evident that a critical leverage point in the effective use and operation of these systems is the efficient assignment of tasks to processors. It is also clear that network bandwidth increases will create tightly coupled, yet distributed systems, where a high number of inter-task communications between processors can occur. To this end, we present a new approach for solving larger, denser instances than previously reported for an important problem in distributed systems, the task allocation problem.

The fundamental task allocation problem consists of assigning tasks, at minimal total cost, to processors. There are costs associated with specific task-processor assignments as well as costs for communicating between any tasks assigned to different processors. The *uncapacitated task allocation problem* (UTAP) regards the execution and communication costs as most relevant and does not place any constraints on processor limitations or load balancing. Variations of the *capacitated task allocation problem* are presented in Hamam [14]. Although this paper restricts its attention to UTAP, the constrained version will be considered in future research.

UTAP is modeled as a quadratic integer program, similar to the quadratic graph coloring formulation found in Carlson and Nemhauser [5]. Although the classic task allocation problem model presented by Stone [24] used max-flow/min-cut to polynomially solve two-processor problems, the extension to larger numbers of processors becomes NP-hard [2, 21]. An exact branch-and-bound algorithm is presented in Magirou [23], which utilized an effective lower bound quickly computed by dynamic programming on a task graph. As the level of inter-task communication increased, for problems having up to 15 tasks, 6 processors, the time required increased significantly. Billionnet [2] also presented an exact approach that was used to solve (or get very close to optimal) problems with up to 50 tasks and 20 processors, however the number of quadratic terms (inter-task communications) was a very small percentage (<1%) of the total possible.

Because of its complexity, various heuristic approaches have been proposed, including simulated annealing [14], genetic algorithms [13], and hybrids [13, 7]. A general overview of UTAP and various heuristic approaches to its solution is found in Chu [8]. A greedy heuristic, based on execution costs presented in Lo [21] is used to find solutions to problems which include task interference costs. Their approach was tested on problems having up to 35 tasks being assigned to 6 processors. In Kopidakis [20], the problem is transformed to a maximization of clustered task communication and the original graph is transformed to include normalized edge (assignment penalty) weights. Their MaxEdge clustering approach outperformed Lo [21] on problems having up to 50 tasks and 20 processors with up to 80% inter-task communication. In Chen [7] a hybrid tabu search / noising method approach is used on problems having up to 20 tasks, 6 processors with good run time and solution quality.

In this paper we utilize tabu search within an xQx formulation to solve a range of sparse and dense problems having up to 3,000 binary variables and over 122,000

quadratic terms (100 tasks, 30 processors), much larger than previously seen. We significantly improve upon the solutions found for these problems by the industry-standard solver, CPLEX, using its mixed integer quadratic capabilities.

## 2. Mathematical Model

In arriving at the UTAP model, the following assumptions are made. The processors are *heterogeneous*, that is, different processors have different processing capabilities and thus have different processing costs for the same task. Multiple tasks executed by the same processor have negligible communication costs. The *transmission cost* of communicating between processors is a constant for all processors and is not included in the minimization model. The communication cost between tasks is *independent of the processor* executing the task, see Dutta et al [9] for a quadratic partitioning formulation where communication between processors is taken into account. There are no task precedence relationship constraints, i.e. communication between tasks  $T_i$  and  $T_j$  does not depend on scheduling one task to be complete before the other can start, as in Hsu et al [17]. There are no interference costs for competing tasks that are assigned to the same processor as in Lo [21].

In the fashion of Billionnet, Costa and Sutter [2], let  $P = \{P_1, P_2, \dots, P_m\}$  be the set of distributed processors and  $T = \{T_1, T_2, \dots, T_n\}$  be the set of tasks to be run on the processors. Let  $U = \{(i, j) : T_i \text{ and } T_j \text{ require communication during processing}\}$ . Let  $c_{ij}$  be the communication cost between tasks  $T_i$  and  $T_j$ , where  $(i, j) \in U$ . Let  $q_{tp}$  be the execution cost of task  $t$  on processor  $p$  and  $x_{tp}$  be the binary decision variable that is equal to 1 if task  $T_i$  is assigned to processor  $P_p$ , and is otherwise 0.

The model below is due to Billionnet et al [2] and represents a reformulation of the traditional UTAP model so that the objective function rewards assigning to the same processor those tasks requiring communications.

$$\text{Letting } l_{tp} = q_{tp} + \sum_{\substack{i, j \in U, \\ i < j}} c_{ij}$$

the UTAP model can be written as:

$$\begin{aligned} \text{Min } & \sum_{t=1}^n \sum_{p=1}^m l_{tp} x_{tp} - \sum_{\substack{i, j \in U, \\ i < j}} \sum_{p=1}^m c_{ij} x_{ip} x_{jp} \\ \text{s.t. } & \sum_{p=1}^m x_{tp} = 1 \quad \text{for } t = 1, \dots, n \\ & x_{tp} \in \{0, 1\} \quad \text{for } t = 1, \dots, n \text{ and } p = 1, \dots, m \end{aligned} \quad (1)$$

### 2.1 Solving UTAP

Several methods for solving UTAP, as mentioned earlier in section 1, have been proposed in the literature. The computational burden of UTAP restricts the use of exact solution methods to small and medium sized problems. Larger instances require heuristic methods to identify high quality solutions within reasonable amounts of computational time. Our approach to solving UTAP first re-casts the problem into the form of an unconstrained quadratic binary program (UQP) which, in turn, is solved by a

tabu search method developed for solving the general UQP model. This approach to solving combinatorial problems by formulating and solving them as UQP instances has proven to be very effective for a broad array of problem classes. As we illustrate later in this paper, the approach is very attractive for UTAP as well. The reformulation of UTAP as an instance of UQP is accomplished by the use of quadratic infeasibility penalties. The general process is described below:

**Transformation to  $xQx$ :**

Many combinatorial optimization problems can be modeled as constrained optimization problems of the form

$$\min x_0 = xQx$$

subject to

$$Ax = b, \quad x \text{ binary}$$

The foregoing 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). These constrained quadratic optimization models are then converted into equivalent UQP models by adding a quadratic infeasibility penalty function to the objective function, as an alternative to explicitly imposing the constraints  $Ax = b$ , as follows:

We choose a positive scalar  $P$ , to yield

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

where the matrix  $D$  and the additive constant  $c$  result directly from the matrix multiplication indicated. We can drop the additive constant, whereupon the equivalent unconstrained version of our constrained problem becomes

$$UQP(PEN) : \min x\hat{Q}x, \quad x \text{ binary} \tag{2}$$

For  $P$  sufficiently large, the optimal solution to UQP(PEN) is the optimal solution to the original constrained problem. For the problems considered here choosing an appropriate value of  $P$  did not pose a problem. In fact the choice of a suitable penalty,  $P$ , has not been a problem in other problem settings either. Note that the re-casting of the original into  $xQx$  is accomplished without the introduction of additional variables. This (general) notion of reformulating a constrained problem as an equivalent unconstrained quadratic problem is an old but greatly under-utilized idea put forth in the late 60s by Hammer and Rudeanu [15] and more recently by Hansen et al [16] and Boros and Hammer [4]. The paper by Kochenberger et al [18] discusses the application of this approach to a wide variety of diverse problem classes.

**2.2 Solving UQP**

Several methods for solving UQP have appeared in the literature in recent years. (see [1, 3, 6, 10, 22]). The approach used in this research for minimizing  $xQx$  is centered around 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 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 *critical events*, and their corresponding *critical solutions*. In this problem setting, we define a critical event to have occurred when exactly  $n$  assignment variables are equal to 1.

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.

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. A complete description of the framework for the method is given in Glover et al [10, 11].

### 3. Computational Experience

A problem generator was written to generate difficult UTAP problems. The inputs to the generator were the number of tasks,  $n$ , the number of processors,  $m$ , the upper bound on processor execution costs ( $B_{proc}$ ), the percent of inter-task communications and a random number seed. The generator creates both an lp-formatted quadratic problem and the associated  $Q$  matrix for UQP. The cost  $q_{tp}$  for assigning a task  $t$  to a processor  $p$  is randomly generated, then the inter-task costs  $c_{ij}$  are randomly generated according to the calculated upper bound,  $B_{com}$  (see below).

Experimentation with upper bound limits on the parameters  $B_{proc}$  and  $B_{com}$  indicate that the interaction between the processor’s task execution costs and the inter-task communication cost is critical in creating a difficult problem. When total execution cost is much larger than the communication costs (or vice-versa), the problems generated were easily solved. Generating problems with comparable sums for execution and communication costs was used by Billionnet [2] and the relation between communication and execution costs and the number of inter-task communications is discussed in Kopidakis et al [20].

In this investigation, the upper bound on inter-task communication cost,  $B_{com}$  is based on the relationship between total processor costs, number of possible inter-task communications, and percent communication.

$$B_{com} = k * \left[ \sum_t \sum_p q_{tp} / (n * (n - 1)) \right] / \text{percent\_inter-tasking} \quad (3)$$

where  $k$  is a constant used to linearly adjust the upper bound, for our problems  $k = 0.5$  created difficult problems. Note that in (3)  $B_{com}$  is directly proportional to  $B_{proc}$  and

inversely proportional to the possible number of inter-tasks and the percent inter-tasking, i.e. as the amount of inter-task communication goes up,  $B_{com}$  goes down to make the total communication costs of the same order as total execution costs. The problem generator first creates all  $q_{tp}$  between 1 and  $B_{proc}$ , then calculates  $B_{com}$ . The characteristics of the problem set used for comparison testing is illustrated in Table 1. All problems have  $B_{proc} = 100$ . The problem ID indicates the number of processors and number of tasks, e.g. 10\_100 indicates 100 tasks executed by 10 processors.

Table 1. Problem Characteristics

Problem ID	% Inter-tasking	# binary vars	# of quadratic terms
10_100a	25%	1000	12810
10_100b	50%	1000	24840
10_100c	75%	1000	37150
10_100d	5%	1000	2560
15_100a	25%	1500	19065
15_100b	50%	1500	37140
15_100c	75%	1500	55875
15_100d	5%	1500	3930
25_100a	25%	2500	30925
25_100b	50%	2500	61225
25_100c	75%	2500	93475
25_100d	5%	2500	6675
30_100a	25%	3000	36420
30_100b	50%	3000	73710
30_100c	75%	3000	112200
30_100d	5%	3000	7860

Each test problem was then solved by CPLEX and by our tabu search heuristic after reformulating as UQP. For all our testing, the scalar penalty,  $P$ , was taken to be 9999. This proved to be sufficiently large for the problems reported on here.

#### 4. Results of Testing

CPLEX version 8.1 extends its branch-and-cut algorithm so that quadratic terms involving binary variables are allowed in the objective function and it also performs presolve reductions on quadratic problems. All problems solved by CPLEX were run on a Dell PowerEdge Server having four 2 GHz processors and 4 GB of RAM. Parallel processing was not utilized. The solutions reported for our tabu search heuristic on the xQx formulation were obtained on a 1.6 GHz Pentium 4 PC with 256 MB of RAM. CPLEX, which ran for an average time of 11.7 hours (stopped by either memory limitations or a time limit of 48 hours) was unable to prove optimality for any of the test problems. The best solutions found by CPLEX were generally obtained in less than five minutes with no solution improvement realized in the lengthy branch-and-cut process that ensued. For each problem instance, our tabu search heuristic was run for an arbitrary limit of 300 SPAN cycles, at which time the best solution found was reported. For the largest problems a run of 300 SPAN cycles took slightly under 12 minutes. Run times for the smallest problems were less than 2 minutes. CPLEX can also restart an optimization search from a previous solution and it was hoped that this capability would be able to prove optimality for the tabu search solutions; unfortunately, the starting solutions provided by tabu search were neither improved upon nor shown to be optimal by CPLEX within our memory and time limits.

Table 2 presents the results from solving sixteen UTAP problems with both tabu search and CPLEX. The tabu search metaheuristic shows significant improvements in the quality of the solution, especially for the larger problems assigning 100 tasks to 25 or 30 processors. Tabu search solves these problems with an average 49% improvement over the solutions found by CPLEX. There was not such a dramatic difference on the problems with less than 2500 variables. The average optimality gap for all problems was 52%, indicating that a good lower bound was difficult to find. In an attempt to prove optimality, CPLEX was started with the significantly improved tabu search solutions, but after 12 hours the solutions were not improved, optimality was not proven and the optimality gap was only reduced from an average of 63% to 47%. Based on results reported in the literature, we expect that other recently reported methods would have computational difficulties with our test problems due to their large number of variables and dense inter-tasking characteristics. For instance, Billionnet [2] reports good results for medium sized problems with little inter-tasking but suggest that the complexity of solving the relaxations increases rapidly with increases in inter-tasking. Although Kopidakis [20] reported that the three heuristics they compared were not affected by inter-tasking density, their maximum problem size was 1,000 binary variables.

Table 2. Comparison of /Tabu Search to CPLEX MIQP Optimizer

Problem ID	% Inter-tasking	Tabu search	CPLEX	CPLEX - tabu search	1 - CPLEX / tabu search
10_100a	25%	39993	39453	-540	1.4%
10_100b	50%	39234	39020	-214	0.5%
10_100c	75%	40137	40188	51	-0.1%
10_100d	5%	35674	35372	-413	0.8%
15_100a	25%	47686	50334	2648	-5.6%
15_100b	50%	48893	51384	2491	-5.1%
15_100c	75%	49817	51424	1607	-3.2%
15_100d	5%	42419	42874	455	-1.1%
25_100a	25%	43821	72109	28288	-65%
25_100b	50%	43871	65721	65721	-50%
25_100c	75%	46288	72670	26382	-57%
25_100d	5%	43700	62402	18702	-43%
30_100a	25%	44302	67421	23119	-52%
30_100b	50%	44362	69626	25264	-57%
30_100c	75%	46530	53580	7050	-15%
30_100d	5%	43866	68924	25058	-57%

## 5. Summary and Conclusions

In this paper we have presented UQP as an alternative model for the classical representation found in the literature for the task allocation problem (UTAP). This slight re-formulation is motivated by the successes we have found in applying this approach to other problem settings. Once in the form of UQP, we solve a given UTAP instance by employing our tabu search methodology designed for the UQP model. Computational experience was presented showing that this approach is not only competitive with other methods found in the literature and the state-of-the-art commercial product, CPLEX, but that it is clearly preferred when the problem size grows beyond a certain limit.

Additionally, while increasing graph density poses severe problems for other approaches presented in the literature, the UQP representation we employ and our tabu search solution method are relatively unaffected by the number of inter-task communications.

Note that while we restrict our computations in this paper to problems of size up to 3000 binary variables, we have solved instances of xQx representing other classes of problems with more than 13000 binary variables. Thus, the xQx representation and the solution methods we have developed for xQx have the potential for solving much larger UTAP instances.

Our experience to date underscores the viability of the approach put forth here and motivates continuing work focusing on addressing larger instances of UTAP as well as the capacitated version of the problem. Results of these extensions will be reported in future papers.

## References

- M. Amini, B. Alidaee, G. Kochenberger, A Scatter Search Approach to Unconstrained Quadratic Binary Programs, *New Methods in Optimization*, Cone, Dorigo, and Glover, McGraw-Hill, 1999.
- A. Billionnet, M. C. Costa, A. Sutter, An efficient algorithm for a task allocation problem, *Journal of the Association for Computing Machinery*, 39 (3) (1992) 502-518.
- A. Billionnet, A. Sutter, Minimization of a quadratic pseudo-boolean function." *European Journal of OR*, 78 (1994) 106-115.
- E. Boros, P. Hammer, Pseudo-boolean optimization, *Discrete Applied Mathematics*, 123(1-3), 155-225 (2002)
- R. Carlson, G. Nemhauser, Scheduling to minimize interaction cost, *Oper. Res.*, 14 (1966), 52-58.
- P. Chardaire, A. Sutter, A Decomposition Method for Quadratic Zero-One Programming, *Management Science*, 41:4, 704-712, 1994.
- W. Chen, C. Lin, A hybrid heuristic to solve a task allocation problem, *Computers & Oper. Res.* 27 (2000) 287-303.
- W. Chu, L. Holloway, M. Lan, K. Efe, Task allocation in distributed data processing, *IEEE Computer*, 13 (11) (1980) 57-69.
- A. Dutta, G. Kochler, A. Whinston, Optimal allocation in a distributed processing environment, *Management Science*, Duxbury, 1982.
- F. Glover, G. A. Kochenberger, B. Alidaee, Adaptive memory tabu search for binary quadratic programs, *Management Science*, 44 (3) (1998) 336-345
- F. Glover, G. Kochenberger, B. Alidaee, and M.M. Amini, Tabu with Search Critical Event Memory: An Enhanced Application for Binary Quadratic Programs, In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, (Eds.) S. Voss, S. Martello, I. Osman, and C. Roucairol. Kluwer Academic Publisher, Boston, 1999.
- F. Glover, B. Alidaee, C. Rego, and G. Kochenberger, One-Pass Heuristics for Large-Scale Unconstrained Binary Quadratic Programs, *EJOR* 137, (2002), 272-287.
- A. B. Hadj-Aloune, J. Bian, K. Murty, A hybrid genetic/optimization algorithm for a task allocation problem, *Journal of Scheduling*, 2 (1999) 189-201.
- Y. Hamam, K. Hindi, Assignment of program modules to processors: A simulated annealing approach, *European Journal of Oper. Res.*, 122 (2000) 509-513.
- P. Hammer, S. Rudeanu, *Boolean methods in operations research and related areas*, Springer, Berlin, Heidelberg, New York, 1968.
- P. Hansen, Jaumard, B, and Mathon, V. Constrained nonlinear 0-1 programming, *INFORMS Journal on Computing*, 5:2, (1993), 97-119.

- T. Hsu, J. Lee, D. Lopez, W. Royce, Task allocation on a network of processors, IEEE Trans. On Computers, 49 (12) (2000) 1339-1353.
- G. Kochenberger, F. Glover, B. Alidaee, and C. Rego, An Unconstrained Quadratic Binary Programming Approach to Modeling and Solving Combinatorial Optimization Problems, University of Colorado Working Paper, 2002.
- G. Kochenberger, Glover, F, Alidaee, B and C. Rego, C. A unified modeling and solution framework for combinatorial optimization problems, OR-Spectrum, 2003, (to appear)
- Y. Kopidakis, M. Lamari, V. Zissimopoulos, Journal of parallel and distributed computing, 42 (1997) 21-29.
- Lo, Heuristic algorithms for task assignment in distributed systems, IEEE Trans. On Comp., 37 (11) (1988) 1384-1397.
- A. Lodi, K. Allemand, T.M. Liebling An Evolutionary Heuristic for Quadratic 0-1 Programming, European Journal of Operational Research 119, 662-670, 1999.
- F. Magirou, J. Z. Milis, An algorithm for the multiprocessor assignment problem, Oper. Res. Let., 8 (1989) 351-356.
- H. Stone, Multiprocessor scheduling with the aid of network flow algorithms, IEEE Trans. On Softw. Eng., SE-3 (1977) 85-93.