Neighborhood Combination for Unconstrained Binary Quadratic Programming

Zhipeng Lü* Fred Glover† Jin-Kao Hao‡

*†‡LERIA, Université d'Angers
2 boulevard Lavoisier, 49045 Angers, France
lu@info.univ-angers.fr hao@info.univ-angers.fr

†1OptTek Systems, Inc.
1919 Seventh Street Boulder, CO 80302, USA
glover@opttek.com

Abstract

Using the Unconstrained Binary Quadratic Programming (UBQP) problem as a case study, we present an experimental analysis of neighborhood combinations for local search based metaheuristic algorithms. In this work, we use one-flip and two-flip moves and investigate combined neighborhoods with these two moves within two metaheuristic algorithms. The goal of the analysis is to help understand why and how some neighborhoods can be favorably combined to increase their search power.

1 Introduction

Neighborhood search or local search is known to be a highly effective metaheuristic framework for solving a large number of constraint satisfaction and optimization problems. One of the most important features of local search is the definition of its neighborhood. The behavior of local search depends strongly on the characteristics of the neighborhood relation. Using the Unconstrained Binary Quadratic Programming (UBQP) problem as a case study, we present in this work an experimental analysis of neighborhoods.

The unconstrained binary quadratic programming problem may be written as:

UBQP: Maximize $x_o = xQx'$

x binary

where Q is an $n \times n$ matrix of constants and x is an n-vector of binary (zero-one) variables.

The formulation UBQP is notable for its ability to represent a wide range of important problems, including those from social psychology ([10]), financial analysis ([14, 18]), computer aided design

([13]), traffic management ([4, 20]), machine scheduling ([1]), cellular radio channel allocation ([3]) and molecular conformation ([19]). Moreover, many combinatorial optimization problems pertaining to graphs are known to be susceptible to formulation by the UBQP problem. A review of additional applications and formulations can be found in [12].

In order to study the search capability of different neighborhood combinations, we are interested in two well-known moves, namely one-flip and two-flip moves, and investigate the performance of two metaheuristic algorithms on different neighborhood combinations using these two basic moves. Computational results show that certain combinations are superior to others. Using three criteria for neighborhood evaluation, we perform further analysis to explain why and how some neighborhoods can be combined to enhance the search.

The remaining part of this paper is organized as follows. In Section 2, the two neighborhood moves and their fast evaluations are described. Sections 3 and 4 are dedicated to several neighborhood combinations and local search based algorithms respectively. In Section 5, we present our computational results on different algorithm-neighborhood combinations. Finally in Section 6, we discuss three criteria used for neighborhood evaluation and draw conclusions.

2 Neighborhood Moves and Fast Evaluation

2.1 One-flip move

The one-flip move complements (flips) a chosen binary variable by subtracting its current value from 1. One-flip is widely used in local search algorithms for binary problems such as UBQP, multi-dimensional knapsack, covering and satisfiability problems.

Let $N = \{1, \ldots, n\}$ denote the index set for components of the x vector. We preprocess the matrix Q to put it in lower triangular form by redefining (if necessary) $q_{ij} = q_{ij} + q_{ji}$ for $i > j$, which is implicitly accompanied by setting $q_{ji} = 0$ (though these 0 entries above the main diagonal are not stored or accessed). Let Δx_i be the move value of flipping the variable x_i, and let $q_{(i,j)}$ be a shorthand for denoting q_{ij} if $i > j$ and q_{ji} if $j > i$. Then each move value can be calculated in linear time using the formula:

$$\Delta x_i = (1 - 2x_i)(q_{ii} + \sum_{j \in N, j \neq i, x_j = 1} q_{(i,j)}) \tag{1}$$

For large problem instances, it is imperative to be able to rapidly determine the effect of a move on the objective function x_o. For this purpose, we employ a fast incremental evaluation technique first introduced by Glover et al [7] and enhanced by Glover and Hao [5] to exploit an improved representation and to take advantage of sparse data - a characteristic of many real world problems. The procedure maintains a data structure that stores the move value (change in x_o) for each possible move, and employs a streamlined calculation for updating this data structure after each iteration.

Moreover, it is not necessary to recalculate all the move values after a move. Instead, one needs just to update a subset of move values affected by the move. More precisely, it is possible to update the move values upon flipping a variable x_i by performing the following abbreviated calculation:

1. $\Delta x_i = -\Delta x_i$
2. For each $j \in N$, $j \neq i$,
   if $x_j = x_i$, then $\Delta x_j = \Delta x_j + q_{(i,j)}$
   if $x_j = 1 - x_i$, then $\Delta x_j = \Delta x_j - q_{(i,j)}$

where x_i represents x_i's value before being flipped.

2.2 Two-flip move

In the case of a two-flip neighborhood, we are interested in the change in x_o that results by flipping 2 variables, x_k and x_j, and will refer to this change by δ_{kj}. It is convenient to think of the two-flip process as a combination of two single one-flip moves, and we can derive δ_{kj} using the one-flip move values Δx_k and Δx_j as follows (supposing $k > j$):

$$\delta_{kj} = \Delta x_k + \Delta x_j + \theta_{kj}\ Q_{kj} \tag{2}$$

where $\theta_{kj} = 1$ if $x_k = x_j$ and $\theta_{kj} = -1$ otherwise.

After a two-flip move is performed, we execute an efficient update of the two-flip delta array δ that is affected by this move. Accompanying this, we introduce additional data structures to speed up the process of identifying the best two-flip move for the next iteration. See [6] for more details. In the following, we respectively denote the neighborhoods with one-flip and two-flip moves as N_1 and N_2.

3 Neighborhood Combinations

In order to increase the search capability of single neighborhoods, it has become a popular practice to combine two or more different neighborhoods, especially when those neighborhoods have quite different characteristics. There are several ways to combine different neighborhoods [11]. In this paper we focus on two of them: neighborhood union and token-ring search [17].

There are two forms of neighborhood union: strong neighborhood union and selective neighborhood union. For strong neighborhood union, denoted by $N_1 \sqcup N_2$, the algorithm picks each move (according to the algorithm's selection criteria) from all the N_1 and N_2 moves. For selective neighborhood union, denoted by $N_1 \cup N_2$, the search algorithm selects one of the two neighborhoods to be used at each iteration, choosing the neighborhood N_1 with a predefined probability p and choosing N_2 with probability 1-p. An algorithm using only N_1 or N_2 is of course a special case of an algorithm using $N_1 \cup N_2$ where p is set to be 1 and 0 respectively.

In token-ring search, the neighborhoods are alternated, applying the currently selected neighborhood without interruption, starting from the local optimum of the previous neighborhood, until no improvement is possible. More precisely, the search procedure uses one neighborhood until a *best* local optimum is determined, subject to time or iteration limits imposed on the search (For metaheuristic searches, this may not be the first local optimum encountered). Then the method switches to the other neighborhood, starting from this local optimum, and continues the search in the same fashion. The search comes back to the first neighborhood at the end of the second neighborhood exploration, repeating this process until no improvement is possible. The token-ring search of two neighborhoods can be denoted as $N_1 \rightarrow N_2$ (starting from N_1) or $N_2 \rightarrow N_1$ (starting from N_2) [17].

4 Metaheuristic Algorithms

For the purpose of studying the different neighborhoods and their combinations, we implement two metaheuristic algorithms, Tabu Search (TS) [8] and Iterated Local Search (ILS) [15].

Within TS, a tabu list is introduced to forbid revisiting a solution previously visited. In our implementation, each time a variable x_i is flipped, this variable enters into the tabu list (an n-vector) and cannot be flipped for the next tt iterations (tt is the "tabu tenure"). For the current study, we set $tt = C + rand(10)$ where C is a given constant and rand(10) takes a random value from 1 to 10.

Our TS procedure uses a token ring search (denoted $N_1 \rightarrow N_2$ for our two neighborhood case), by starting the TS procedure with neighborhood N_1. When the search ends with its best local optimum, we restart TS from this solution, but using the other neighborhood N_2. Starting again from N_1, using the local optimum found by N_2, this process is repeated until no improvement is possible, at which point we say that a TS phase is achieved. The application of TS to a single neighborhood stops when the best solution cannot be improved within a given number θ of moves and we call this number the *improvement cutoff* of TS. In this paper, we set $\theta = 10,000$ for all experiments.

Our ILS algorithm takes the standard steepest descent (SD) algorithm as its local search procedure and employs the so-called Critical Element-Guided Perturbation (CEGP) strategy to jump out of the local optima trap [16]. This perturbation operator is composed of three steps: 1) Scoring: assign a score to each variable; 2) Selection: choose a certain number of highly-scored variables (critical elements); 3) Perturbing: randomly perturb the solution using the chosen critical elements.

Similar to the TS procedure, when ILS uses the token-ring search of two neighborhoods ($N_1 \rightarrow N_2$), the SD algorithm alternates between N_1 and N_2 by starting with N_1. Interested readers are referred to [9] for more details about the CEGP-based ILS algorithm for UBQP.

5 Computational Comparison

In this Section, we show computational results of the aforementioned ILS and TS algorithms using the following neighborhoods: N_1 (one-flip), N_2 (two-flip), $N_1 \sqcup N_2$ (strong union), $N_1 \cup N_2$ (selective union) with $p = 0.5$ and $p = 0.8$ and $N_1 \rightarrow N_2$ (token-ring). Experiments are carried out on the set of the 10 largest instances with 2500 variables from ORLIB [2]. To make the comparison as fair as possible, all the experiments use the same stopping conditions, i.e. the CPU timeout is set to be 150 seconds on our computer with 3.4GHz and 2GB Memory. Given the stochastic nature of our TS and ILS algorithms, each problem instance is independently solved 25 times.

Table 1 shows the computational statistics of the ILS algorithm (N_{12} is a shorthand for denoting $N_1 \cup N_2$). Columns 2 and 3 respectively give the density (dens) and the best known objective

Table 1: Results of the ILS algorithm on the 10 Beasley instances with size n=2500 from ORLIB

instance	dens	f_{best}	solution gaps to f_{best} for ILS algorithm ($f_{best} - f$)					
			N_1	N_2	$N_1 \sqcup N_2$	$N_{12(p=0.5)}$	$N_{12(p=0.8)}$	$N_1 \rightarrow N_2$
b2500.1	0.1	1515944	5115	8465	8326	8561	8639	**4041**
b2500.2	0.1	1471392	4984	5866	6482	6765	6356	**3432**
b2500.3	0.1	1414192	3994	6737	7591	8310	7906	**3439**
b2500.4	0.1	1507701	**2073**	5094	6204	6441	5196	2315
b2500.5	0.1	1491816	3903	5635	6358	7106	6598	**2496**
b2500.6	0.1	1469162	3955	5174	5847	6408	4330	**2800**
b2500.7	0.1	1479040	**2229**	7258	7561	7042	8202	4038
b2500.8	0.1	1484199	2305	4255	5264	**1416**	5309	1965
b2500.9	0.1	1482413	3940	4728	4687	6074	6864	**2316**
b2500.10	0.1	1483355	4707	3812	6827	9028	7723	**3587**
average			3720.5	5702.4	6514.7	6715.1	6712.3	**3042.9**

Table 2: Results of the TS algorithm on the 10 Beasley instances with size n=2500 from ORLIB

instance	dens	f_{best}	gaps to f_{best} for TS algorithm ($f_{best} - f$)					
			N_1	N_2	$N_1 \sqcup N_2$	$N_{12(p=0.5)}$	$N_{12(p=0.8)}$	$N_1 \rightarrow N_2$
b2500.1	0.1	1515944	**0**	440	1354	2338	1123	**0**
b2500.2	0.1	1471392	14	934	854	824	1445	**12**
b2500.3	0.1	1414192	**0**	1444	1845	1704	1646	**0**
b2500.4	0.1	1507701	**0**	341	235	150	248	**0**
b2500.5	0.1	1491816	**0**	891	594	236	1497	**0**
b2500.6	0.1	1469162	**0**	1976	1369	1145	1131	**0**
b2500.7	0.1	1479040	**0**	1370	1284	1784	706	**0**
b2500.8	0.1	1484199	4	497	467	650	568	**0**
b2500.9	0.1	1482413	**0**	421	503	430	638	**0**
b2500.10	0.1	1483355	**0**	1023	789	560	836	**0**
average			1.8	933.7	929.4	972.1.1	983.8	**1.2**

values (f_{best}). Columns 4 to 8 give the solution gap to the best solutions for each neighborhood and neighborhood combination. For each instance, the solution gap in Table 1 is represented as $f_{best} - f$, where f is the average objective value obtained by 25 independents runs. The overall results, averaged over 10 instances, are presented in the last row.

From table 1, we observe that neighborhood N_1 outperforms N_2 in terms of solution quality for these test problems. When comparing the four neighborhood combinations $N_1 \sqcup N_2$, $N_1 \cup N_2$ with $p = 0.5$ and $p = 0.8$ as well as $N_1 \rightarrow N_2$ with each other, $N_1 \rightarrow N_2$ is superior to the three neighborhood unions and it is also slightly better than N_1. Contrary to our expectation, the neighborhood unions $N_1 \cup N_2$ with $p = 0.5$ and $p = 0.8$ get the worst results among all these neighborhoods and neighborhood combinations. For each pairwise of these neighborhoods, we performed a 95% confidence t-test to compare their solution quality, leading to the following ranking of the neighborhoods: $N_1 \rightarrow N_2 > N_1 > N_2 \approx N_1 \sqcup N_2 \approx N_{12(p=0.5)} \approx N_{12(p=0.8)}$.

Similarly, the computational results of our TS algorithm on the two neighborhoods and their union and token-ring combinations are given in table 2. Once again, we performed a 95% confidence t-test to compare different neighborhoods and observed that $N_1 \rightarrow N_2$ and N_1 are superior to others in terms of solution quality. These results coincide well with the results obtained by the ILS algorithm and confirm the ranking given above: $N_1 \rightarrow N_2 > N_1 > N_2 \approx N_1 \sqcup N_2 \approx N_{12(p=0.5)} \approx N_{12(p=0.8)}$.

6 Discussions

The foregoing computational results show that better outcomes are achieved with the token-ring combination of one-flip and two-flip moves. Moreover, we observe that the simple one-flip based neighborhood performs quite well. Some important questions remain.

1. These results are based on *random* instances. It would be interesting to know whether these results would be confirmed on other types of problems. To this end, a sequel to this study will carry out additional experiments using more diverse instances transformed from other problems [12].
2. It would be useful to identify the conditions under which a particular neighborhood or a neighborhood combination is preferable.
3. More importantly, it would be valuable to understand what causes a particular neighborhood or neighborhood combination to be effective under given circumstances. For this purpose, we will investigate the approach proposed in [17] which identifies several criteria to characterize the search capacity of a neighborhood such as *percentage of improving neighbors*, *improvement strength* and *search steps*.
4. It would be worthwhile to investigate other combinations of N_1 and N_2. For example, we may select an N_1 improving move that gives the best N_2 move, leading to a "conditional" combination.

We anticipate that answers to these issues will provide information that will be valuable for the design of improved algorithms, and yield a foundation for similar analysis in related contexts.

Acknowledgement

The work is partially supported by a "Chaire d'excellence" from "Pays de la Loire" Region (France) and regional MILES (2007-2009) and RaDaPop projects (2008-2011). We are grateful for comments by the referees that have improved the exposition of the paper.

References

[1] B. Alidaee, G. Kochenberger and A. Ahmadian. 0-1 quadratic programming approach for the optimal solution of two scheduling problems. International Journal of Systems Science, 25 (1994), 401-408.

[2] J.E. Beasley. Obtaining test problems via internet. Journal of Global Optimization, 8 (1996) 429-433.

[3] P. Chardaire and A. Sutter. A decomposition method for quadratic zero-one programming. Management Science, 41(4) (1994) 704-712.

[4] G. Gallo, P. Hammer and B. Simeone. Quadratic knapsack problems. Mathematical Programming, 12 (1980)132-149.

[5] F. Glover and J.K. Hao. Efficient evaluations for solving large 0-1 unconstrained quadratic optimization problems. International Journal of Metaheuristics (2009) (To appear).

[6] F. Glover and J.K. Hao. Fast 2-flip move evaluations for binary unconstrained quadratic optimization problems. Working paper, LERIA, Université d'Angers (2009).

[7] F. Glover, G.A. Kochenberger and B. Alidaee. Adaptive memory tabu search for binary quadratic programs. Management Science, 44 (1998) 336-345.

[8] F. Glover and M. Laguna. Tabu Search. Kluwer Academic, Boston, 1997.

[9] F. Glover, Z. Lü and J. K. Hao. Diversification-driven Tabu Search for unconstrained binary quadratic problems. Research Report, LERIA, Université d'Angers (2009).

[10] F. Harary. On the notion of balanced of a signed graph. Michigan Mathematical Journal, 2 (1953) 143-146.

[11] L. D. Gaspero, A. Schaerf, Neighborhood portfolio approach for local search applied to timetabling problems. Journal of Mathematical Modeling and Algorithms 5(1) (2006) 65–89.

[12] G.A. Kochenberger, F. Glover, B. Alidaee and C. Rego. A unified modeling and solution framework for combinatorial optimization problems. OR Spectrum, 26 (2004) 237-250.

[13] J. Krarup and A. Pruzan. Computer aided layout design. Mathematical Programming Study, 9 (1978) 75-94.

[14] D.J. Laughunn. Quadratic binary programming. Operations Research, 14 (1970) 454-461.

[15] H. R. Lourenco, O. Martin, T. Stützle, Iterated local search. Handbook of Meta-heuristics, Springer-Verlag, Berlin Heidelberg, 2003.

[16] Z. Lü and J.K. Hao. A critical element-guided perturbation strategy for Iterated Local Search. Cotta and P. Cowling (Eds.): EvoCOP 2009, Lecture Notes in Computer Science 5482 (2009) 1-12.

[17] Z. Lü, J. K. Hao and F. Glover. Neighborhood analysis: a case study on curriculum-based course timetabling. Journal of Heuristics (2009) (To appear).

[18] R.D. McBride and J.S. Yormark. An implicit enumeration algorithm for quadratic integer programming. Management Science, 26 (1980) 282-296.

[19] A.T. Phillips and J.B. Rosen. A quadratic assignment formulation of the molecular conformation problem. Journal of Global Optimization, 4 (1994) 229-241.

[20] C. Witsgall. Mathematical methods of site selection for electronic system (EMS). NBS Internal Report. (1975).