Solving Large Binary Quadratic Programming Problems by Effective Genetic Local Search Algorithm

Kengo Katayama **Masafumi Tani** **Hiroyuki Narihisa**
Department of Information and Computer Engineering
Okayama University of Science
1 - 1 Ridai-cho, Okayama, 700-0005 Japan
E-Mail: {katayama, tani, narihisa}@ice.ous.ac.jp

Abstract

A genetic local search (GLS) algorithm, which is a combination technique of genetic algorithm and local search, for the unconstrained binary quadratic programming problem (BQP) is presented. An effective local search algorithm, which is a variant of the *k-opt* local search for the BQP by Merz *et al.*, is described, and the performance of the GLS with the variant local search heuristic is demonstrated on several large-scale problem instances. Our computational results indicate that the GLS is able to frequently find the best-known solution with a relatively short running time and obviously our average solution values obtained are better than previous powerful heuristic approaches especially for the large problem instances of 2,500 variables.

1 INTRODUCTION

It is well known that search performances of genetic algorithms can be enhanced by incorporating local search heuristics. Such the incorporation is often called as *Genetic Local Search* (GLS). In many cases, the GLS is capable of finding relatively good solutions for difficult optimization problems. However, the performance of the GLS may depend on choices of genetic operators and local search of which the GLS algorithm is composed. In this paper, the GLS is interpreted as follows: a main search engine of the GLS is the local search heuristic, and the genetic operators such as selection, crossover, and mutation work as auxiliary search engines for the local search. In addition, the genetic operators may contribute to explore new regions of search space of a given problem and many individuals of a population in the GLS can be biased so as to lead to high-quality solutions corresponding to a collection of its promising search points during evolutions in the GLS.

In this paper, we consider the unconstrained binary quadratic programming problem (BQP) which is an NP-hard problem and has a large number of important applications. To solve the BQP, several exact methods have been developed [3, 6, 10, 22]. However due to the computational complexity of the problem, at the present time it is only capable of solving the small size instances. To obtain near-optimal solutions, several heuristic approaches such as tabu search [5, 9], simulated annealing [2, 5, 13], genetic (local search) algorithms [18, 20], and iterated local search algorithms [14] have been proposed for the BQP.

For example, Merz and Freisleben [20] have proposed a genetic local search algorithm for the BQP in the GECCO-99 and impressive results for the test problems of the BQP contained in the OR-Library [4] were shown. Moreover, they gave better values of best-known solutions for several large instances than that found by tabu search and simulated annealing investigated by Beasley [5]. Furthermore, Katayama and Narihisa [13] have proposed a high-performance simulated annealing for the large instances ranging from 500 to 2,500 variables. This simulated annealing obtained new best-known solutions, which were even better than that reported by Merz et al. [20], for several large instances and it was verified to be powerful and faster than the heuristic algorithms presented in [20] and [5].

These heuristic algorithms mentioned above belong to the meta-heuristics and are based on *1-opt* local search. The *1-opt* local search is the simplest one, which iteratively searches the solutions that can be reached by flipping a single element in the current solution. In this paper, we present a new genetic local search (GLS) algorithm for the BQP. Our GLS algorithm is composed of simple genetic operators and an effective local search heuristic which is more powerful than the *1-opt* local search. The effective local search is a variant of the *k-opt* local search, which has been proposed by Merz and Freisleben [21]. By using this variant *k-opt* local search heuristic, we implement the effective GLS algorithm which is able to obtain high-quality solutions

even for large-scale instances of the BQP. Experimental results demonstrated that the GLS frequently could find the best-known or near best-known solutions for the instances ranging from 1,000 to 2,500 variables in the OR-Library. Moreover, we showed that its running times were reasonable.

The paper is organized as follows. In Section **2**, we give the definition of the BQP and review the previous research on the recent heuristics for the BQP. Section **3** describes our genetic local search algorithm, including our effective *k-opt* local search algorithm. In Section **4**, we evaluate our GLS for the large instances and show its performance by comparing with the previous powerful heuristic approaches, and Section **5** contains concluding remarks.

2 THE BQP AND PREVIOUS RESEARCH

The objective of the BQP is to find, given a symmetric rational $n \times n$ matrix $Q = (q_{ij})$, a binary vector of length n that maximizes the following quantity:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j, \; x_i \in \{0, 1\} \; \forall i = 1, \ldots, n. \quad (1)$$

The BQP belongs to the class NP-hard and has a large number of important applications, e.g., machine scheduling [1], traffic message management problem [8], CAD problem [16], capital budgeting and financial analysis problem [19], molecular conformation problem [25]. Furthermore, it has been known that the BQP is equivalent to many classical combinatorial optimization problems such as maximum cut, maximum clique, maximum vertex packing, minimum vertex cover, maximum independent set, and maximum weight independent set problems [3, 11, 23, 24].

To solve the BQP, several exact methods such as branch & bound or branch & cut have been developed by many researchers, e.g., Pardalos and Rodgers [22], Barahona, Jünger, and Reinelt [3], Billionnet and Sutter [6], Helmberg and Rendl [10]. However due to the computational complexity of the problem, at the present time it is only capable of solving the small size instances. For larger instances, such the methods would become prohibitively expensive to apply, whereas high-performance heuristics might find high-quality solutions with short times. The recent survey concerning the exact methods can be found in [5]

On the other hand, several heuristic approaches have been proposed to the BQP. These approaches are generally based on improving the current single solution (or multiple solutions) by a greedy search and other concepts for the neighborhood of the solution. In the following, we briefly review recent powerful heuristic algorithms that could find optimal/best-known or very good approximate solutions for the BQP.

Glover, Kochenberger and Alidaee [9] have proposed a tabu search heuristic for instances of up to 500 variables. Their tabu search consists of a strategic oscillation scheme which alternates between constructive and destructive phases.

Lodi, Allemand, and Liebling [18] have proposed a heuristic based on the genetic algorithm for the same problem set studied by Glover et al. Their heuristic is combined with the local search algorithm that is based on the constructive and destructive phases as in Glover et al. In a crossover operator, they performed an operator based on the uniform crossover utilizing the MinRange algorithm which is based on the property by Pardalos and Rodgers [22].

Alkhamis, Hasan, and Ahmed have proposed a simulated annealing [2]. Unfortunately, only small problem instances with up to 100 variables were investigated.

In [5], Beasley has provided larger BQP test problems with up to 2,500 variables as new test problems of the OR-Library [4]. Beasley showed the best-known solution for each of the provided instances by performing other tabu search and simulated annealing.

Merz and Freisleben have proposed a genetic local search algorithm [20], and tested on the instances studied in [5]. In their genetic local search, a simple local search (called *1-opt*, see below) and a variant of uniform crossover, HUX [7], were employed. For several large instances, they gave new best-known solutions and showed that their algorithm outperformed two alternative heuristics reported by Beasley. Furthermore, Merz et al. have developed a greedy heuristic and two local search heuristics called *1-opt* and *k-opt* [21]. They showed that in particular the *k-opt* local search was capable of finding high-quality solutions even for the large-scale problem instances, and they also implied that these heuristics are well suited as components for the meta-heuristics.

Katayama and Narihisa [13] have proposed a new simulated annealing-based heuristic with reannealing process and tested on the large instances ranging from 500 to 2,500 variables contained in the OR-Library. Although the simulated annealing was based on the simple *1-opt* neighborhood structure, better average solution results for the large instances were shown by comparing with the other heuristics: the genetic local search by Merz et al. [20] and the heuristics by Beasley [5]. Moreover, it was considerably faster than the others, and new best-known solutions for several large instances were reported.

3 GENETIC LOCAL SEARCH ALGORITHM FOR THE BQP

The genetic local search algorithm is composed of the genetic operators and the local search. Generally, since the local search is applied to new offspring created by

crossover or mutation operators, all individuals (solutions) in the population represent local optima. For a set of the local optimal solutions, a selection operator is applied to select promising individuals to make the next collection of parents. In this section, we describe details of search parts of which the GLS algorithm for the BQP is composed, including our variant *k-opt* local search algorithm.

When applied GLS to a specific problem, it is important to determine the fitness function and the representation. For the BQP, Eq. (1) can be used as the fitness function. On the other hand, 0-1 binary representation is an obvious choice for the BQP since it represents the underlying 0-1 integer variables. In our representation, we used a n-bit binary string, where n is the number of variables in the BQP, and a value of 0 or 1 at the j-th bit implies that $x_j = 0$ or 1 in the individual used in our GLS, respectively. All combinations of 0 or 1 appeared in all n elements of the individual are feasible.

3.1 LOCAL SEARCH

Local search is a generally applicable approach that can be used to find approximate solutions to hard optimization problems. The basic idea is to start from an initial solution x, e.g., a randomly generated solution, and to search a better solution in its *neighborhood* $N(x)$ until no better solution is found in $N(x)$. Thus, the resulting solution can not be improved by reference to $N(x)$ and is a local optimal solution in the neighborhood N.

The performance of such the local search substantially depends on: 1) the definition of the neighborhood $N(x)$ and 2) the move strategy, i.e., how to search better '*neighbor*' solutions from $N(x)$ in the current solution.

Neighborhood

For the BQP, due to the binary representation a structure of the simplest neighborhood is the *1-opt* neighborhood $N_1(x)$ that can be performed by flipping a single element in the solution x, i.e., the hamming distance d_H between the current solution x and its neighbor solution $x' \in N_1(x)$ is equal to 1. Therefore, a size of candidates of neighbor solution obtained by $N_1(x)$ from the current solution x is only n. Merz and Freisleben have proposed a powerful local search heuristic that efficiently searches a small fraction of the *k-opt* neighborhood such that $1 \leq d_H \leq k$ (a number of d_H is variable) [21]. The variable-depth *k-opt* local search algorithm for the BQP is based on ideas of Kernighan and Lin who proposed effective heuristics for the graph partitioning problem [15] and the traveling salesman problem [17], which are well-known combinatorial optimization problems.

Move Strategy

The move strategy in the local search is defined by a scheme of how to search the neighbor solutions that can be reached by reference to the neighborhood in the current solution. Generally, the move strategy used in the local search may be divided into two types: 1) best-improvement move strategy and 2) first-improvement move strategy. The *best-improvement* move strategy selects the improved neighbor solution x' with the best cost in the entire candidate set of $N(x)$ and the solution x' becomes the next solution. On the other hand, the *first-improvement* move strategy scans neighbor solutions in $N(x)$ according to a pre-specified (random) order, and if the improved neighbor solution x' is found during the scan, the solution x' is immediately accepted as the next solution.

Both strategies can be adopted as a component of algorithms for the BQP. For example, Merz et al. have adopted the best-improvement for the *1-opt* local search algorithm in their GLS [20]. For our simulated annealing approach to the BQP, we have adopted the first-improvement in the *1-opt* neighborhood. The detailed description and fundamental results of the best-improvement *1-opt* and the first-improvement *1-opt* can be found in [13].

Variant *k-opt* Local Search for the BQP

A variant of the *k-opt* local search algorithm proposed by Merz and Freisleben [21] is described. Our variant *k-opt* local search for the BQP is based on an ingenious combination of both the *k-opt* local search with the best-improvement by Merz et al. (`MFk-opt`) and the *1-opt* local search with the first-improvement presented by Katayama and Narihisa (`KN1-opt`) [13]. Roughly speaking, the search by the *1-opt* neighborhood of `KN1-opt` contributes to oscillations that are performed so as to move good points likely to lead to better solutions. Furthermore, although the resulting solution found by the *k-opt* local search with the best-improvement depends on the starting solution, `KN1-opt` works so as to mitigate such the search of `MFk-opt`. This variant local search is performed until no better solution can be found.

Fig. 1 shows the variant local search procedure written by a similar code description in [21]. Given a solution x and all gains g, provided that x should not be local optimum and each (g_i) of g has a gain value for neighbor solutions that can be reached by flipping i-th bit in the given solution x. The details for calculating and fast updating of all gains at Step1.2.2.2 and Step1.2.7 for the BQP's solution can be found in [21]. In the inner repeat-loop of Step1.2, a search process of both our *1-opt* local search and the *k-opt* local search is performed. The process from Step1.2.3 to Step1.2.8 is the same as the search of the *k-opt* local search by Merz et al. and the process for `KN1-opt` is located from

procedure Variant-k-Opt-Local-Search (x, g)
1 Do the following until $G_{max} \leq 0$. (Initially, a large integer value is set to G_{max})
1.1 Set $x_p = x$, $G_{max} = 0$, $G = 0$, $C = \{1, \ldots, n\}$.
1.2 Do the following until $C = \phi$.
1.2.1 Generate a random permutation $RP[\ ]$ ranging from 1 to n.
1.2.2 Perform the following n times (j is incremented from 1 to n)
1.2.2.1 $k = RP[\ j\ ]$.
1.2.2.2 If $G + g_k > G_{max}$ then
- Set $G = G + g_k$ and $G_{max} = G$.
- Set $x_k = 1 - x_k$.
- Set $x_b = x$.
- Update gains g_i for all i in {1,...,n}.
- Set $C = C \backslash \{k\}$.
1.2.3 Find j with $g_j = \max_i g_i$.
1.2.4 Set $G = G + g_j$.
1.2.5 If $G > G_{max}$, then set $G_{max} = G$, $x_b = x$.
1.2.6 Set $x_j = 1 - x_j$.
1.2.7 Update gains g_i for all i.
1.2.8 Set $C = C \backslash \{j\}$.
1.3 If $G_{max} > 0$, then set $x = x_b$, else set $x = x_p$.
2 Return x.

Figure 1: A variant *k-opt* local search procedure.

Step1.2.1 to Step1.2.2.2. In this study, the `KN1-opt` process is performed in only the first iteration of loop of Step1.2 that comes after Step1.1 rather than in all iterations of the loop, in order to search gainful neighbors. Therefore, the final stage of the variant search is equivalent to the `MFk-opt`. Although a candidate set C is used to assure that each bit is flipped exactly once in the `MFk-opt` process, we do not care it in the search process of `KN1-opt`. However, for forthcoming `MFk-opt`, all bits that contributed to improve the solution are removed from C at Step1.2.2.2.

To reduce the running time, Merz et al. suggested to modify the termination condition of the inner repeat-loop. We also modify it (Step1.2) so that the iterative search in the loop is terminated if there were no new x_b solution for more than m iterations, provided that $m \ll n$. In a preliminary testing, we found a good value of m in a trade-off between solutions obtained and running times. In our all experiments in this paper, we set to $m = 50$. Turning now to a fundamental performance, the variant local search based on this ingenious combination of both *1-opt* neighborhood with the first-improvement `KN1-opt` and *k-opt* neighborhood with the best-improvement `MFk-opt` can obtain slightly better local optimal solutions on average without a large amount of running time than the original *k-opt* local search of Merz et al. Furthermore, since this variant search is performed with random choices of better neighbors found by `KN1-opt`, the resulting local optima strongly do not depend on the starting solutions in many cases.

3.2 GENETIC OPERATORS

Designs of the genetic operators are simple in our GLS approach to the BQP. However, crossover and mutation are designed so as to create appropriate offspring for the variant *k-opt* local search described above, since the local search starts from these offspring solutions except for a creating process of initial solutions. The flow of our GLS is illustrated in Fig. 2. Initial PS individuals ($I_1^0, \ldots, I_{PS}^0$) of a population P^0 are randomly generated and are locally optimized by the variant *k-opt* local search heuristic to obtain a new P^1, which is an initial set of local optima. After that, a main loop of the GLS is repeated until a generation counter GC is reached a predefined termination generation GN or for a time-limit. Each of the genetic operators and others implemented in our GLS is described in the following.

Crossover

In the BQP, classical genetic crossovers can be applied without any modification. For our GLS approach to the BQP, the uniform crossover (UX) [26] is employed because several researchers who investigated genetic local search approaches to the BQP have chosen the UX or its variant as a crossover. From a collection of promising individuals in the population, the couples of $PS/2$ are randomly chosen, but each individual is permitted to be a mate exactly once in the population. In our crossover, one offspring is created from the couple of two parents as shown in Fig. 3, and thus a number of all offspring after the crossovers of $PS/2$ times is

procedure Genetic-Local-Search
0 Determine a population size PS and a generation number GN or a time-limit for the termination. Initialize a generation counter $GC = 1$.
1 Generate initial individuals $I_1^0, \ldots, I_{PS}^0$ of population P^0 randomly.
2 Each individual $I^0 \in P^0$ is locally optimized by the local search for obtaining a new P^1.
3 Do the following until GC is reached to GN or for the time-limit.
  3.1 Increment $GC = GC + 1$.
  3.2 Do the following until a number $PS/2$ of all couples is reached.
    3.2.1 Select a couple I_a, $I_b \in P^{GC-1}$ randomly.
    3.2.2 Perform Crossover(I_a, I_b), obtaining I_c.
    3.2.3 Perform Mutation to the offspring I_c, if needed.
    3.2.4 Run the local search on the offspring I_c, obtaining new I_c.
    3.2.5 I_c is inserted to an offspring population P_c^{GC}.
  3.3 (Each fitness of all individuals $\in P^{GC-1}$ and P_c^{GC} is calculated.) Select better PS individuals to be the population P^{GC} from P^{GC-1} and P_c^{GC} so that individuals which have the same fitness can not be survived.
  3.4 Perform diversification strategy to the population P^{GC}, if needed.
4 Return the best individual $\in P$.

Figure 2: The flow of genetic local search.

$parent1$	1	0	0	1	1	0	1	0
$parent2$	0	0	1	1	0	1	1	0
	$*$	0	$*$	1	$*$	$*$	1	0
$offspring$	0	0	1	1	1	0	1	0

Comment : For the offspring, a value of 0 or 1 at each position '*' is chosen with probability 0.5.

Figure 3: An example of our uniform crossover.

equal to $PS/2$. In the crossover process, the resulting offspring are expected to be suitable solutions. The suitable solutions imply that they move to hopeful regions of search space of the BQP from the old states and are not local optima. Generally, it is difficult to judge whether the offspring is suitable without a measure. As the measure, we check a hamming distance between their two parents (see Mutation for more details). If this judgment by the hamming distance is satisfied, the offspring is improved by the effective local search without any modification such as the mutation as described below.

Mutation

When the resulting offspring created by the crossover is not suitable, the mutation operator is applied to the offspring as an auxiliary operator of the crossover. In our GLS, we check the hamming distance d_H of parents that were used in each of the crossover processes as described above. If $d_H(I_a, I_b)$, i.e., the hamming distance between two parents, was below a number of $n/10$, where n denotes a number of variables of a given BQP instance, the mutation operator flips a value of the offspring location randomly chosen from all locations where have the same value at the same location in both parents until the hamming distance between the offspring and the parent I_a (or I_b) is reached the number of $n/10$. In our preliminary testing, it appears that the number $n/10$ was suitable in our GLS.

Selection

All solutions after the local search which starts with offspring created by the crossover and mutation operators represent local optima. Before applying a selection operator in each generation, we have local optima of $PS + PS/2$ which are parent and newly created offspring solutions. Due to the fixed population size, PS individuals with better fitness are selected from all the candidate local optima. However, duplicate individuals are not contained in the new population and new individual, i.e., offspring, which is one of the duplicates, is survived to the next generation. The similar selection in the GLS approach can be found in [12].

Diversification/Restart Strategy

Obviously, during the search, our selection operator leads to a convergence in the population. Thus, we perform a diversification/restart strategy to move other points of the search space if no new best individual in the population was found for more than 30 generations or the average distance of the population has dropped below a threshold 30. (In our observations, this strategy was mainly performed with the former requirement.) In response to these requirements, the individuals except for the best one in the population are mutated by flipping randomly chosen $n/2$ bits for each individual. After that, each of the

mutated individuals is improved by the local search heuristic to obtain a new set of local optima and the search is started again with newly diverse individuals. This strategy is borrowed from [7] and [20]. Although our strategy is considerably disruptive, our *k-opt* local search heuristic can easily compensate for search points significantly moved and in the long run, these, i.e., processes of the diversification and the concentration, cooperate in moving new regions where are not visited in the search space and may contribute to improve a total performance of the GLS.

4 COMPUTATIONAL RESULTS

This section reports the results of computational experiments which were performed for our effective GLS algorithm described above. In order to show the performance of our GLS, a comparison is carried out with the recently reported results of the genetic local search by Merz and Freisleben [20], the tabu search and simulated annealing by Beasley [5], and the simulated annealing by Katayama and Narihisa [13] for the large-scale test problems ranging from 1,000 to 2,500 variables contained in the OR-Library [4].

Our GLS algorithm implemented in *C* are executed on a Sun Ultra 5/10 (UltraSPARC-IIi 440MHz) under the OS Solaris 7. The algorithm code is compiled with `gcc` compiler using optimization flag `-O2`. The population size PS is 20. The crossover rate is set to 1.0 and the mutation rate is not set because the mutation operator is adaptively performed according to states of offspring created by the crossover. See sub-section **3.2**.

The large-scale test problems are 20 instances. The breakdown of these instances is as follows. Each of two sets that were firstly studied in [5], `beas1000` ($n = 1,000$) and `beas2500` ($n = 2,500$), consists of ten instances with a matrix density $dens(Q)$ of 0.1, where $dens(Q)$ is defined as the number of non-zero entries divided by the number of total entries in the matrix. At the present time, the instance of size $n = 2,500$ is the largest BQP test problem in the OR-Library.

Table 1 summarizes the results of our GLS incorporating the variant *k-opt* local search and of the other algorithms, which have been tested on the large-scale instances. For our GLS, 30 runs were performed and the running times to reach the best-known solutions were recorded. If the best-known solution could not be found for each run, the run was performed until the time-limit (60(s) for each instance of `beas1000` and 360(s) for `beas2500`) was reached. In this table, for our GLS, the best found solution value "best", the average final solution "avg.", the number of times in which the best-known solution could be found "(b/30)", the average running time "t1" in seconds in the case which the GLS could find the best-known solution, and the time-limit "t2" in seconds except for the case which the GLS could find the best-known solution, were provided. In the other results, for SA-KN [13], we provided the reported results of the average final solution in 30 runs, "(b/30)" by SA-KN, and the average running times in seconds which were required by SA-KN on Sun Ultra 5/10 (UltraSPARC-IIi 440MHz). For GLS-MF [20], the average final solution in 30 runs was provided for each instance. Their GLS was performed until 600(s) for each instance of `beas1000` or 1,200(s) for each of `beas2500` was reached on Pentium II PC (300MHz). For TS-B and SA-B, in [5], Beasley provided the best result for each instance. These algorithms (TS-B and SA-B) for each instance of `beas1000` consumed about 4,500(s) and 6,800(s), respectively, and for each of `beas2500` about 14 hours and 17 hours were required on Silicon Graphics (R4000 CPU with 100MHz), respectively.

As observed in Table 1, our GLS obtained good average solution values in comparison to the other existing heuristic algorithms: GLS-MF and two alternative approaches of TS-B and SA-B, especially for the largest problem set tested, `beas2500`, because our GLS algorithm could frequently find the best-known solution which agreed with the value of the best solution found by the simulated annealing algorithm (SA-KN) for each instance investigated in [13]. Particularly, in *all* 30 runs, the best-known solutions were found for all the instances of the set of `beas1000` and `beas2500-1`, `-4`, `-5`, `-6` instances.

For the general framework of GLS, our approach to the BQP is similar to the other existing genetic approaches by Merz et al. and Lodi et al. [18] in that all individuals in the population after the local search represent local optima and the similar genetic operators (e.g., uniform crossover) are performed. However, one of the most different points between our GLS and the others may be that our GLS algorithm is performed so as to achieve an effective use and properties of our local search algorithm for finding better local optima. To show a superior point of our GLS, we should give the average number of generations which the GLS spent to find the best-known solution. For example, in instances of 1,000 variables, the GLS spent the following average generation numbers: 13, 10, 5, 11, 25, 11, 21, 30, 15, and 15 generations for each instance (from `beas1000-1` to `beas1000-10`) of `beas1000`. In comparison with GLS-MF (their PS was 40, see [20]), the generation numbers spent by our GLS are considerably fewer. From this fact, we believe that our variant *k-opt* local search used in the GLS contributed much to find high-quality solutions with fewer generations (i.e., it implies the smaller number of cost evaluations) or short running times.

In terms of the running times, it seems that the simulated annealing SA-KN is fast compared to the other algorithms (because we have already observed in [13] that SA-KN on S-4/5 with 110MHz was still faster than the others) if we do not push the fact in respect

Table 1: Comparison of our genetic local search algorithm (GLS) and four algorithms: simulated annealing (SA-KN) by Katayama and Narihisa, genetic local search (GLS-MF) by Merz and Freisleben, tabu search (TS-B) and other simulated annealing (SA-B) by Beasley, for `beas1000` and `beas2500` instances.

BQP	GLS				SA-KN		GLS-MF		TS-B	SA-B
instance	best	avg.(b/30)	t1(s)	t2(s)	avg.(b/30)	t(s)	avg.	t(s)	best	best
beas1000-1	371438	371438.0 (30)	5	—	371342.1 (10)	1.5	371304.1	600	371438	371134
beas1000-2	354932	354932.0 (30)	4	—	354836.5 (16)	1.4	354862.3	600	354932	354637
beas1000-3	371236	371236.0 (30)	3	—	371193.5 (22)	1.5	371233.8	600	371073	371226
beas1000-4	370675	370675.0 (30)	4	—	370605.4 (13)	1.5	370506.0	600	370560	370265
beas1000-5	352760	352760.0 (30)	10	—	352685.0 (3)	1.5	352687.6	600	352736	352297
beas1000-6	359629	359629.0 (30)	5	—	359480.0 (10)	1.5	359487.8	600	359452	359313
beas1000-7	371193	371193.0 (30)	8	—	371046.3 (11)	1.5	371084.9	600	370999	370815
beas1000-8	351994	351994.0 (30)	11	—	351844.9 (2)	1.5	351844.6	600	351836	351001
beas1000-9	349337	349337.0 (30)	7	—	349160.9 (3)	1.4	349253.3	600	348732	348309
beas1000-10	351415	351415.0 (30)	6	—	351214.1 (10)	1.5	351125.6	600	351408	351415
beas2500-1	1515944	1515944.0 (30)	32	—	1515828.9 (9)	15.1	1514804.6	1200	1514971	1515011
beas2500-2	1471392	1471195.1 (13)	215	360	1470600.9 (1)	15.2	1469721.0	1200	1468694	1468850
beas2500-3	1414192	1414111.9 (21)	117	360	1413657.1 (8)	15.1	1412943.0	1200	1410721	1413083
beas2500-4	1507701	1507701.0 (30)	22	—	1507630.3 (21)	14.6	1507674.2	1200	1506242	1506943
beas2500-5	1491816	1491816.0 (30)	51	—	1491692.8 (6)	15.2	1491623.4	1200	1491796	1491465
beas2500-6	1469162	1469162.0 (30)	52	—	1468810.3 (5)	15.3	1467918.2	1200	1467700	1468427
beas2500-7	1479040	1479038.8 (29)	117	360	1478397.4 (2)	15.4	1477101.7	1200	1476059	1478654
beas2500-8	1484199	1484197.1 (25)	86	360	1483907.9 (6)	15.0	1483226.9	1200	1484199	1482953
beas2500-9	1482413	1482411.3 (27)	176	360	1482192.0 (1)	15.1	1481622.9	1200	1482306	1481834
beas2500-10	1483355	1483172.8 (16)	178	360	1482522.4 (1)	15.4	1481899.2	1200	1482354	1482166

to the average results of the solutions found by SA-KN. However, our GLS seems to be capable of finding better solutions on average with reasonable running times (although it is not clear whether the SA-KN can obtain comparative results on average with the longer running times spent by our GLS). If required the best-known solution frequently, our GLS approach appears to be one of the most promising approaches to the BQP. From a practical point of view, the performance of our GLS algorithm outperformed the previously reported heuristic algorithms in terms of average solution results.

In the OR-Library, there are many test problem instances of the BQP that are smaller than `beas1000` and `beas2500`. However, these small instances relatively become easier to solve by our GLS approach. For example, for `glov500-1` of $n = 500$ and $dens(Q) = 0.1$ the GLS could found the best solution of $f(x) = 61,194$ in all 30 runs and the average running time was less than a second on the computational circumstances mentioned above. Even for a test instance of $n = 500$ and $dens(Q) = 0.75$, namely `glov500-4`, which appeared to be hard for the genetic local search algorithm by Merz et al. [20], our GLS obtained the best solution of $f(x) = 172,771$ in all 30 runs with the average time of 24 seconds. For `glov500-5` with $dens(Q) = 1.0$, the best solution of $f(x) = 190,507$ could be found by our GLS in all 30 runs with the average time of 11 seconds. In this connection it seems that the density of the matrix, $dens(Q)$, has a significant effect on the performance or running times of the GLS as well as the previous GLS of Merz et al. To claim the effectiveness of our GLS approach in the strict sense, additional testing may be required on larger instances with higher densities such as $des(Q) > 0.1$.

5 CONCLUSION

This paper presented an effective genetic local search (GLS) algorithm for solving the large-scale instances of the unconstrained binary quadratic programming problem (BQP). It was shown that our GLS incorporating the variant *k-opt* local search heuristic was able to frequently obtain the best-known solution with reasonable running times. Although the GLS consumed more running times than the simulated annealing presented in [13], we demonstrated that the GLS was capable of finding better solutions on average than the simulated annealing or the other existing heuristic algorithms especially for the large problems.

Moreover, it was showed that the GLS approach could be considerably improved by incorporating the effective local search into the GLS framework. However, our GLS preferably performed the mutation in comparison to the previous GLS approaches to the BQP. This may raise the question of whether GLS algorithms that often perform disruptive operations are always better than 'conservative' GLS algorithms for the BQP that seems to have a structured landscape.

Acknowledgements

The authors would like to thank Peter Merz of University of Siegen for valuable discussions and comments.

References

[1] B. Alidaee, G.A. Kochenberger, and A. Ahmadian, "0-1 quadratic programming approach for the optimal solution of two scheduling problems," International Journal of Systems Science, vol.25, no.2, pp.401–408, 1994.

[2] T.A Alkhamis, M. Hasan, and M.A. Ahmed, "A simulated annealing for the unconstrained quadratic pseudo-Boolean function," European Journal of Operational Research, vol.108, pp.641–652, 1998.

[3] F. Barahona, M. Jünger, and G. Reinelt, "Experiments in quadratic 0-1 programming," Mathematical Programming, vol.44, pp.127–137, 1989.

[4] J.E. Beasley, "OR-Library: Distributing test problems by electronic mail," Journal of the Operational Research Society, vol.41, no.11, pp.1069–1072, 1990.

[5] J.E. Beasley, "Heuristic algorithms for the unconstrained binary quadratic programming problem," Technical Report, Management School, Imperial College, UK, 1998.

[6] A. Billionnet and A. Sutter, "Minimization of a quadratic pseudo-Boolean function," European Journal of Operational Research, vol.78, pp.106–115, 1994.

[7] L. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," Foundations of Genetic Algorithms, (G.J.E. Rawlings, ed.), pp.265–283, 1991.

[8] G. Gallo, P.L. Hammer, and B. Simeone, "Quadratic knapsack programs," Mathematical Programming, vol.12, pp.132–149, 1980.

[9] F. Glover, G.A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," Management Science, vol.44, no.3, pp.336–345, 1998.

[10] C. Helmberg and F. Rendl, "Solving quadratic (0,1)-problem by semidefinite programs and cutting planes," Mathematical Programming, vol.82, pp.291–315, 1998.

[11] P.L. Ivănescu, "Some network flow problems solved with pseudo-Boolean programming," Operations Research, vol.13, pp.388–399, 1965.

[12] K. Katayama and H. Narihisa, "Performance of genetic approach using only two individuals," Proc. of the 1999 IEEE International Conference on Systems, Man, and Cybernetics, Oct.12–15, Tokyo, Japan, vol.1, pp.I-677–I-682, 1999.

[13] K. Katayama and H. Narihisa, "Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem," Working Paper, 1999. (available from the first author at Dept. of Information and Computer Eng., Okayama University of Science)

[14] K. Katayama and H. Narihisa, "On fundamental design of iterated local search heuristic for binary quadratic programming," Technical Report of IEICE (COMP), Kyoto University, May, 2000. (in Japanese)

[15] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell System Technical Journal, vol.49, pp.291–307, 1970.

[16] J. Krarup and P.M. Pruzan,"Computer aided layout design," Mathematical Programming Study, vol.9, pp.75–94, 1978.

[17] S. Lin and B.W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Operations Research, vol.21, pp.498–516, 1973.

[18] A. Lodi, K. Allemand, and T.M. Liebling, "An evolutionary heuristic for quadratic 0-1 programming," European Journal of Operational Research, vol.119, pp.662–670, 1999.

[19] R.D. McBride and J.S. Yormark, "An implicit enumeration algorithm for quadratic integer programming," Management Science, vol.26, no.3, pp.282–296, 1980.

[20] P. Merz and B. Freisleben, "Genetic algorithms for binary quadratic programming," Proc. of the 1999 Genetic and Evolutionary Computation Conference, vol.1, pp.417–424, 1999.

[21] P. Merz and B. Freisleben, "Greedy and local search heuristics for unconstrained binary quadratic programming," Technical Report No.99-01, Informtik-Berichte, 1999.

[22] P.M. Pardalos and G.P. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero-one programming," Computing, vol.45, pp.131–144, 1990.

[23] P.M. Pardalos and G.P. Rodgers, "A branch and bound algorithm for the maximum clique problem," Computers and Operations Research, vol.19, no.5, pp.363–375, 1992.

[24] P.M. Pardalos and J. Xue, "The maximum clique problem," Journal of Global Optimization, vol.4, pp.301–328, 1994.

[25] A.T. Phillips and J.B. Rosen, "A quadratic assignment formulation of the molecular conformation problem," Journal of Global Optimization, vol.4, pp.229–241, 1994.

[26] G. Syswerda, "Uniform crossover in genetic algorithms," Proc. of the 3rd International Conference on Genetic Algorithms, pp.2–9, 1989.