A Study of Multi-parent Crossover Operators in a Memetic Algorithm

Yang Wang, Zhipeng Lü*, and Jin-Kao Hao

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
{yangw,lu,hao}@info.univ-angers.fr

Abstract. Using unconstrained binary quadratic programming problem as a case study, we investigate the role of multi-parent crossover operators within the memetic algorithm framework. We evaluate the performance of four multi-parent crossover operators (called MSX, Diagonal, U-Scan and OB-Scan) and provide evidences and insights as to why one particular multi-parent crossover operator leads to better computational results than another one. For this purpose, we employ several indicators like population entropy and average solution distance in the population.

Keywords: multi-parent crossover, memetic algorithm, unconstrained quadratic programming, performance analysis.

1 Introduction

Memetic algorithms (MA) are known to be one of the highly effective metaheuristic approaches for solving a large number of constraint satisfaction and optimization problems [1]. One of the most important features of a MA is the crossover operator for generating offspring solutions. In general, meaningful crossover operators help to create healthy diversification in the population and to avoid a premature convergence of the population.

In this paper, we provide a case study of multi-parent crossover operators within memetic algorithms for the unconstrained binary quadratic programming (UBQP) that can be written

$$\text{UBQP: Maximize } f(x) = x'Qx$$
$$x \text{ binary}$$

where Q is an n by n matrix of constants and x is an n-vector of binary variables.

The formulation UBQP is notable for its ability to represent a wide range of important problems [2]. The literature reports a number of evolutionary and memetic algorithms with two-parent crossover operators for solving the UBQP problem ([3,4,5,6]). However, one finds no studies concerning multi-parent crossover operators for UBQP, where multi-parent crossover operators generate offspring solutions by combining more than two parent solutions. In this work, we are particularly interested in investigating the role of multi-parent crossover operators as well as a number of related important questions: why does one particular multi-parent crossover operator lead to better computational results than

* Corresponding author.

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 556–565, 2010.

another one? What are the main characteristics of a good multi-parent crossover operator? To what extent can the crossover operators influence the performance of the memetic algorithms?

Without claiming to answer all these questions, we present an experimental analysis of various multi-parent crossover operators within a memetic algorithm. For this purpose, we use four multi-parent crossover operators, respectively called MSX, Diagonal, U-Scan and OB-Scan. The last three ones are well known in the literature while the first one is recently proposed in [7]. The analysis shows that the computational results are strongly correlated with the characteristics of the corresponding crossover operators, such as the entropy and average solution distance of the population, the average solution quality in the population. Furthermore, the analysis sheds light on how a tradeoff between local search and crossover operator can be achieved when using different crossover operators.

2 Multi-parent Crossover within Memetic Algorithms

2.1 Main Scheme and Initial Population

This study is based on the general memetic framework described in Algorithm 1 that alternates between a combination operator and a local improvement procedure. The combination operator (Section 2.4) is used to generate new offspring solutions while the local improvement procedure based on tabu search (Section 2.2) aims at optimizing each offspring solution. As soon as an offspring solution is improved by tabu search, the population is accordingly updated based on two criteria: the solution quality and the diversity of the population. The individuals of the initial population are generated randomly (i.e., each variable x_i of the n-vector x receives a value of 0 or 1 with equal probability).

Algorithm 1. Pseudo-code of the memetic algorithm

Input: matrix Q
Output: the best solution x^* found so far
$P = \{x^1, \ldots, x^p\} \leftarrow$ Population_Initialization( )
for $i = \{1, \ldots, p\}$ **do**
  $x^i \leftarrow$ Tabu_Search(x^i)
end for
$x^* = arg\ max\{f(x^i)|i = 1, \ldots, p\}$
repeat
  randomly choose a subset of individuals E ($|E| \in [4, 8]$) from P
  $x^0 \leftarrow$ Crossover_Operator(E)
  $x^0 \leftarrow$ Tabu_Search(x^0)
  if $f(x^0) > f(x^*)$ **then**
    $x^* = x^0$
  end if
  $P \leftarrow$ Pool_Updating(x^0, P)
until a stop criterion is met

2.2 Tabu Search Procedure

In this paper, we employ a simple tabu search algorithm as our local search procedure. Our tabu search procedure uses a *neighborhood* defined by the simple *one-flip move*, which consists of changing (flipping) the value of a single variable x_i to its complementary value $1 - x_i$. The implementation of this neighborhood uses a fast incremental evaluation technique [9] to calculate the cost (move value) of transitioning to each neighboring solution.

Tabu search incorporates a *tabu list* as a "recency-based" memory structure to assure that solutions visited within a certain span of iterations, called the tabu tenure, will not be revisited [10]. In our implementation, we elected to set the tabu tenure as $TabuTenure(i) = tt + rand(10)$, where tt is a given constant $(n/100)$ and rand(10) takes a random value from 1 to 10. Our tabu search method stops when a given number α of moves are reached, called *depth* of tabu search.

2.3 Pool Updating

In our memetic algorithm, after an offspring x^0 is obtained by the crossover operator and improved by tabu search, we decide whether the improved offspring should be inserted into the population, replacing the existing *worst* solution. For this purpose, we define a quality-and-distance goodness score of the offspring x^0 with respect to the population. The main idea is to favor the inclusion of x^0 in the population if x^0 is "good enough" (in terms of its objective function evaluation) and is not too *similar* to any solution currently in the population.

Our aim is not only to maintain a pool of good quality solutions but also to emphasize the importance of the diversity of the solutions to avoid a premature convergence of the population. Therefore, if the goodness score of the offspring solution is good enough, it will have high probability to replace the worst solution in the population. Interested readers are referred to [7] for more details.

2.4 Combination Operators

In this paper, we use four multi-parent crossover (or combination) operators to generate offspring solutions, including a "logic" multi-parent combination operator (MSX), a diagonal multi-parent crossover (Diagonal), a multi-parent uniform scanning crossover (U-Scan), a multi-parent occurrence based scanning crossover (OB-Scan). Note that except MSX which is recently proposed for UBQP [7], the last three ones have been widely used for other combinatorial optimization problems in the literature [11,12].

All the four combination operators used in our algorithm are applied on a set E of s ($|E| = s$) parent solutions randomly selected from the current population, i.e., $E = \{x^{(1)}, \ldots, x^{(s)}\}$, where $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$. In our implementation, we set s to be a random number between 4 and 8.

MSX Crossover (MSX): we define a weight $w(i)$ for the solution $x^{(i)}$ and a strength value $Strength(j)$ for variable x_j as: $w(i) = 1/\sum_{j=1}^{n} x_j^{(i)}$ and $Strength(j) = \sum_{i=1}^{s} w(i) x_j^{(i)}$.

The value $Strength(j)$ gives a relative indication of the tendency of the solutions in E to favor $x_j = 1$ or $x_j = 0$. That is, we may say that the larger the value of $Strength(j)$, the greater is the degree that "E favors $x_j = 1$". Then, we take an average of the $sum(i)$ values over E to get a value for the number of x_j components that should be 1 in an "average" solution, denoted by $Avg = \sum_{i=1}^{s} sum(i)/s$.

Thus, the variables with the first Avg largest $Strength$ values receive assignment 1 and other variables receive assignment 0. In practice, it is preferable to shift Avg slightly in one direction or another to increase the diversity of the generated offspring solutions [7].

Diagonal Crossover (Diagonal): it is a generalization of the one-point crossover. For s parent solutions, diagonal crossover divides each parents into s sections through $s-1$ crossover points. Each section has the same length except the last section containing the surplus variables when divided unequally. The offspring is constructed through extracting in a diagonal way respectively one section from each parent. The formal definition is given as follows.

Given set E with s solutions and $s-1$ crossover points $\{y_1, y_2, ..., y_{s-1}\}$, where $y_i = i * n/s$ and $0 < i < s$. Diagonal crossover reproduces the offspring $c = (c_1, c_2, ...c_s)$ by

$$c_k = \begin{cases} x_j^{(1)}, 1 \le j < y_1 & k = 1; \\ x_j^{(k)}, y_{k-1} \le j < y_k & 1 < k < s; \\ x_j^{(s)}, y_{s-1} \le j \le n & k = s. \end{cases} \quad (1)$$

Uniform Scanning Crossover (U-Scan): this is a generalization of the two-parent uniform crossover. U-Scan uses a scheme that one of the parents selected randomly determines the value of the offspring. Thus each parent has the same probability to be the dominator of the value inherited by the offspring. It breaks the limitation of traditionary two parents and can extend the number of parents to an arbitrary number.

Given set E with s solutions, U-Scan generates offspring solution $c = (c_1, c_2, ... c_n)$ as follows. Value c_j is obtained by $c_j = x_j^{(i)}$ where $x_j^{(i)}$ denotes the jth value of parent $x^{(i)}$ and i is randomly selected from 1 to s with probability $1/s$.

Occurrence-Based Scanning Crossover (OB-Scan): OB-Scan relies on parental occurrence on determining the offspring values. Generally speaking, each parent votes and the values inherited will be the one favored by the majority of parents. As for UBQP problem, the value equals either one or zero, thus we record the frequency of each variable's value equal to one appearing in the parents. If this frequency surpasses or is less than the half of the number of parents, then the value of offspring in this position is assigned to one or zero, respectively. Otherwise, the variable is assigned to be one or zero randomly. The following gives a formal definition of OB-Scan.

Given set E, OB-scan reproduces the offspring $c = (c_1, c_2, ...c_n)$ by

$$c_j = \begin{cases} 0, & \sum_{i=1}^{s} x_j^{(i)} < s/2; \\ 1, & \sum_{i=1}^{s} x_j^{(i)} > s/2; \\ rand(0,1), \text{ otherwise.} \end{cases} \quad (2)$$

where $rand(0,1) \in \{0,1\}$ is a binary random function.

3 Experimental Results

3.1 Instances and Experimental Protocol

To evaluate the MSX, Diagonal, U-Scan and OB-Scan crossover operators, we carry out experiments on a set of 15 large random instances with 3000 to 5000 variables from the literature [13]. Our algorithm is programmed in C and compiled using GNU GCC on a PC running Windows XP with Pentium 2.66GHz CPU and 512MB RAM. Given the stochastic nature of the algorithm, problem instances are independently solved 10 times. The stop condition for a single run is respectively set to be 5, 10 and 20 minutes on our computer for instances with 3000, 4000 and 5000 variables, respectively. Note that when performing experiments on each crossover, the only difference consists in the crossover operator and other components of the algorithm are kept unchanged. The parameters are set as follows: population size $p = 30$, *depth* of tabu search $\alpha = 2n$. The experimental results are summarized in Tables 1 and 2.

3.2 Computational Results

Tables 1 and 2 report the best objective values (in parentheses number of hits over 10 runs) and the average objective values using the four crossover operators,

Table 1. Computational results on the 15 large random instances with 3000 to 5000 variables: best values (succ rate)

Instance	MSX	U-Scan	OB-Scan	Diagonal
p3000.1	3931583(9)	**3931583(10)**	3931583(8)	3931583(9)
p3000.2	5193073(10)	5193073(10)	5193073(10)	5193073(10)
p3000.3	5111533(7)	5111533(7)	5111533(7)	5111533(6)
p3000.4	5761822(10)	5761822(10)	5761822(9)	5761822(10)
p3000.5	**5675625(6)**	5675625(1)	5675598(1)	5675625(4)
p4000.1	6181830(10)	6181830(10)	6181830(10)	6181830(10)
p4000.2	**7801355(7)**	7801355(6)	7801355(4)	7801355(6)
p4000.3	7741685(9)	7741685(9)	7741685(9)	7741685(7)
p4000.4	**8711822(10)**	8711822(9)	8711822(7)	8711822(9)
p4000.5	8908979(4)	**8908979(7)**	8908979(3)	8908979(2)
p5000.1	8559015(1)	**8559312(1)**	8559210(3)	8559210(4)
p5000.2	10835437(1)	10835832(3)	10835437(3)	**10835437(5)**
p5000.3	10488783(10)	**10489137(3)**	10489137(1)	10489137(1)
p5000.4	12251211(1)	12251211(2)	12251211(2)	12251211(1)
p5000.5	**12731803(10)**	12731803(1)	12731803(1)	12731803(1)

respectively. We observe that MSX and U-Scan perform slightly better in terms of the best objective values since both two crossovers obtain the best values for 4 out of the 15 instances. OB-Scan seems to be the worst in terms of the best objective value and the success rate. Table 2 indicates that MSX seems to be superior to other three crossover operators in terms of the average objective values since it reaches the best results for 7 out of the 15 instances. Moreover, U-Scan and Diagonal perform quite well on 3 and 2 instances, respectively. In addition, the results also disclose that the performance of various crossover operators depend on the instances to be solved. For example, MSX operator dominates the other three ones on instances p3000.5 and p5000.5; U-Scan obtains excellent results on instance p5000.3; Diagonal performs the best on instances p5000.2.

Table 2. Computational results on the 15 large random instances with 3000 to 5000 variables: average objective function values over 10 runs

Instance	MSX	U-Scan	OB-Scan	Diagonal
p3000.1	3931522	**3931583**	3931368	3931418
p3000.2	5193073	5193073	5193073	5193073
p3000.3	**5111403**	5111307	5111292	5111194
p3000.4	5761822	5761822	5761784	5761822
p3000.5	**5675360**	5675041	5674950	5675130
p4000.1	6181830	6181830	6181830	6181830
p4000.2	**7801170**	7800556	7799766	7800731
p4000.3	7741493	**7741653**	7741557	7741541
p4000.4	**8711822**	8711775	8711410	8711582
p4000.5	**8908438**	8908189	8907347	8906880
p5000.1	8558848	8558945	8558916	**8558963**
p5000.2	10834470	**10835107**	10834762	10834987
p5000.3	**10488783**	10487995	10487865	10487941
p5000.4	12250012	12250161	12249935	**12250559**
p5000.5	**12731803**	12730255	12730454	12730563

4 Analysis

The above computational results show that for certain instances, some crossover operators perform better than other ones in terms of the solution quality. In this section, we attempt to explain what causes the effectiveness or weakness of a crossover operator. For this purpose, we introduce the following evaluation criteria to characterize the search capacity of different crossover operators: population entropy, average solution distance and average solution quality in the population. We argue that a good crossover operator should help the population to maintain a good diversity (high entropy and high average solution distance) and good average solution quality. We also perform an experiment to show how different crossover operators and local search jointly influence the performance of the memetic algorithms.

As an example, the experiments are presented on the large random instance p5000.5. The stopping criterion is the number of generations which is limited to 100. From Tables 1 and 2, one observes that for this instance MSX performs the best, while U-Scan and OB-Scan are much worse than MSX and even Diagonal.

4.1 Evolution of Solution Quality

We first study one of the most important characteristics for the four crossover operators, i.e., the solution gaps to the best known value evolving with the generation iterations, denoted by g_b. g_b is defined as the average value of solution gaps between the best solution in the current population and the best known objective value over 10 independent runs. The smaller is this value, the higher quality the best solution in the population has. Figure 1 shows how this value evolves with the generation iterations for the four crossover operators.

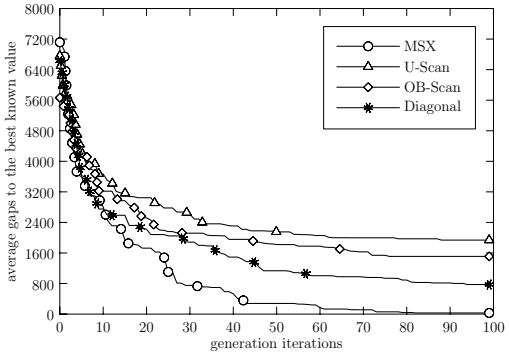

Fig. 1. Best population solution quality evolving with the generation iterations

One observes that at the first generations, the four crossover operators have no clear difference in terms of this criterion. However, with the search progresses, MSX performs much better than other three ones. Note that Diagonal also performs very well and U-Scan is the worst among the four operators. This observation coincides very well with the results reported in Tables 1 and 2, showing the advantage of the crossover operators of MSX and Diagonal, as well as the weakness of U-Scan and OB-Scan for this problem instance.

4.2 Population Entropy and Distance

In our second experiment, we observe the two characteristics of the four multi-parent crossover operators in terms of the population diversity: the *population entropy* vs. the number of generations; the *average solution distance* in the population vs. the number of generations.

The entropy, taking into account the value of each variable in each individual of the population P, is calculated as follows [14]:

$$entropy(P) = \frac{-\sum_{i=1}^{n}\sum_{j=0}^{1}\frac{n_{ij}}{p}log\frac{n_{ij}}{p}}{nlog2} \tag{3}$$

where n is the number of variables and n_{ij} is the number of times the variable x_i is set to j in P. In this definition, $entropy(P) \in [0, 1]$. $entropy(P) = 0$ indicates a population of identical individuals whereas $entropy(P) = 1$ means that all possible assignments are almost uniformly distributed in the population.

The average solution distance in the population is calculated:

$$\bar{d}(P) = \frac{2}{p(p-1)} \sum_{i=1}^{p} \sum_{j=i+1}^{p} d_{ij} \tag{4}$$

where d_{ij} is the Hamming distance between any two solutions $x^{(i)}$ and $x^{(j)}$ in the current population P.

Figure 2 shows how the population entropy (left) and average solution distance of the population (right) evolve with the number of generations. We see that the population diversity measured in terms of these two characteristics is better preserved during the evolution process for MSX and Diagonal than for U-Scan and OB-Scan, especially after the first 60 generations.

Following the spirit of scatter search and path-relinking, an efficient solution combination operator is one that ensures not only high quality solutions but also a good diversity of the population. In other words, the diversification of the population induced by MSX and Diagonal allows the algorithm to benefit from a better exploration of the search space and prevents the population from stagnating in poor local optima.

4.3 Tradeoff between Intensification and Diversification

In this section, we turn our attention to study another important aspect of the memetic algorithms, i.e., the tradeoff between local search and crossover operators. In fact, the performance of the memetic algorithm is influenced by the value of the *depth* of tabu search α. Under a limited computational resource, the *depth* of tabu search reflects the relative proportion of combination operators and tabu search in the algorithm. In this section, we analyze the influence of the parameter α on the performance of the memetic algorithm.

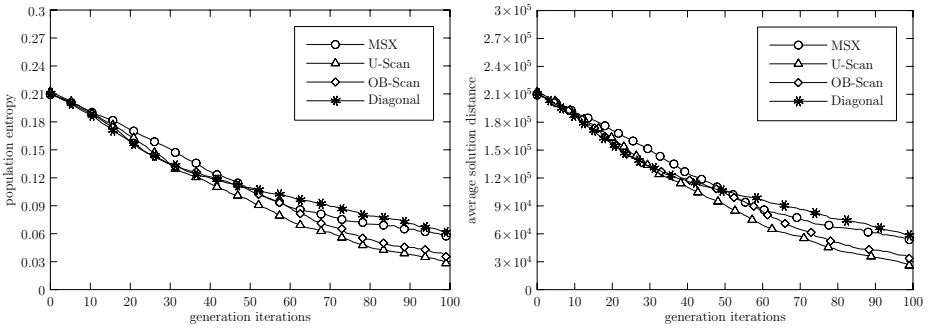

Fig. 2. Comparisons of the four crossover operators in terms of population diversity

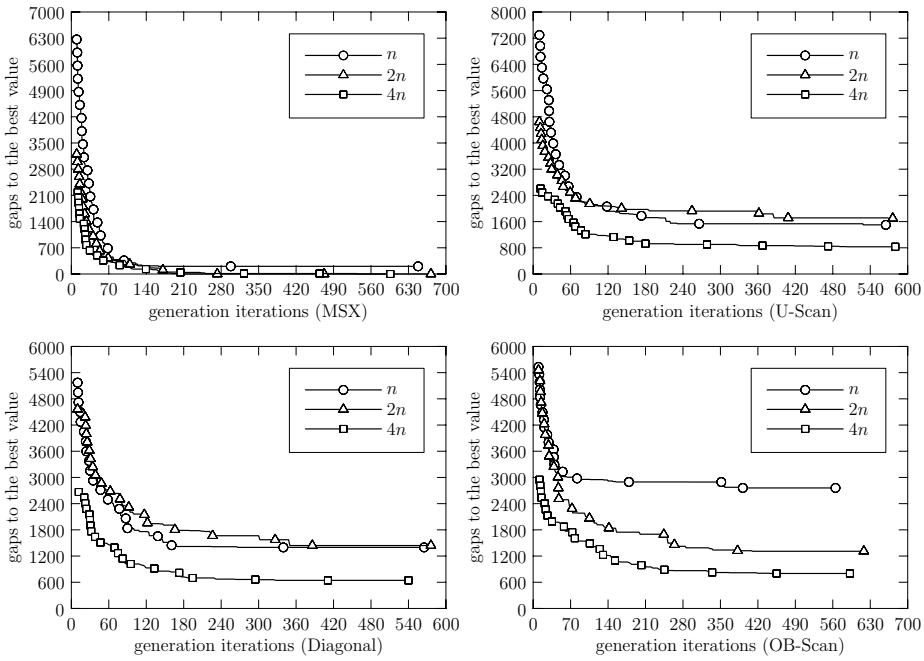

Fig. 3. Tradeoffs between TS and crossover operator

To implement this experiment, we consider 3 different values of the parameter α: $\alpha = n$, $\alpha = 2n$ and $\alpha = 4n$. For each value, we perform 10 independent runs, each run being given 20 minutes CPU time. Figure 3 shows the average evolution of the best solution gaps during the search obtained with these three α values and four crossover operators.

From Figure 3, we first notice that the memetic algorithm performs much worse with $\alpha = $ n and $\alpha = $ 2n than with $\alpha = $ 4n in three cases (Diagonal, U-Scan and OB-Scan), which means that tabu search is an essential part in the memetic algorithm when using these three operators and stronger tabu search can eclipse the role of the crossover. However, when it comes to the MSX crossover operator, one observes that MSX is not really sensitive to various α values, showing that MSX plays a real driving role for the search process. This experiment shows a clear advantage of MSX and the importance of setting an appropriate α value for other crossover operators in order to achieve a desired tradeoff between intensification and diversification.

5 Conclusions

Understanding and explaining the performance of crossover operators within a memetic algorithm is an important topic. In this paper, we presented an attempt

to analyze the intrinsic characteristics of four crossover operators for the UBQP problem. To this end, we employed several evaluation indicators to characterize the search capability of a crossover operator. The experimental analysis allowed us to understand to some extent the relative advantages and weaknesses of the four studied crossover operators within the memetic framework.

Acknowledgement

We are grateful for comments by the referees that help us to improve the paper. The work is partially supported by "Pays de la Loire" Region (France) through RaDaPop and LigeRO projects (2009-2012).

References

1. Moscato, P.: Memetic algorithms: a short introduction. In: New Ideas in Optimization, pp. 219–234. Mcgraw-Hill Ltd., Maidenhead (1999)
2. Kochenberger, G.A., Glover, F., Alidaee, B., Rego, C.: A unified modeling and solution framework for combinatorial optimization problems. OR Spectrum 26, 237–250 (2004)
3. Borgulya, I.: An evolutionary algorithm for the binary quadratic problems. Advances in Soft Computing 2, 3–16 (2005)
4. Katayama, K., Tani, M., Narihisa, H.: Solving large binary quadratic programming problems by an effective genetic local search algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), pp. 643–650. Morgan Kaufmann, San Francisco (2000)
5. Lodi, A., Allemand, K., Liebling, T.M.: An evolutionary heuristic for quadratic 0-1 programming. European Journal of Operational Research 119(3), 662–670 (1999)
6. Merz, P., Katayama, K.: Memetic algorithms for the unconstrained binary quadratic programming problem. BioSystems 78, 99–118 (2004)
7. Lü, Z., Hao, J.K., Glover, F.: A study of memetic search with multi-parent crossover for UBQP. In: Cowling, P., Merz, P. (eds.) EvoCOP 2010. LNCS, vol. 6022, pp. 154–165. Springer, Heidelberg (2010)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
9. Glover, F., Kochenberger, G.A., Alidaee, B.: Adaptive memory tabu search for binary quadratic programs. Management Science 44, 336–345 (1998)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
11. Eiben, A.E., Raué, P.E., Ruttkay, Z.: Genetic algorithms with multi-parent recombination. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 78–87. Springer, Heidelberg (1994)
12. Ting, C.K.: Design and analysis of multi-parent genetic algorithms, PhD Thesis, Univerity of Paderborn (2005)
13. Palubeckis, G.: Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Annals of Operations Research 131, 259–282 (2004)
14. Fleurent, C., Ferland, J.A.: Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 619–652 (1996)