Soft Comput (2010) 14:1117–1139
DOI 10.1007/s00500-009-0506-1

ORIGINAL PAPER

Evaluating a local genetic algorithm as context-independent local search operator for metaheuristics

Carlos García-Martínez · Manuel Lozano

Published online: 8 October 2009

Abstract Local genetic algorithms have been designed with the aim of providing effective intensification. One of their most outstanding features is that they may help classical local search-based metaheuristics to improve their behavior. This paper focuses on experimentally investigating the role of a recent approach, the binary-coded local genetic algorithm (BLGA), as context-independent local search operator for three local search-based metaheuristics: random multi-start local search, iterated local search, and variable neighborhood search. These general-purpose models treat the objective function as a black box, allowing the search process to be context-independent. The results show that BLGA may provide an effective and efficient intensification, not only allowing these three metaheuristics to be enhanced, but also predicting successful applications in other local search-based algorithms. In addition, the empirical results reported here reveal relevant insights on the behavior of classical local search methods when they are performed as context-independent optimizers in these three well-known metaheuristics.

Keywords Local evolutionary algorithms · Local search-based metaheuristics · Context-independent local search · Intensification · Discrete parameter optimization

C. García-Martínez (✉)
Department of Computing and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain
e-mail: cgarcia@uco.es

M. Lozano
Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

1 Introduction

About 30 years ago, a new family of search and optimization algorithms arose based on extending basic heuristic methods by including them into an iterative framework augmenting their exploration capabilities. This group of advanced approximate algorithms received the name *metaheuristics* (MHs) (Glover and Kochenberger 2003; Siarry and Michalewicz 2008) and an overview of various existing methods is found in Blum and Roli (2003). MHs have proven to be highly useful for approximately solving difficult optimization problems in practice because they may obtain good solutions in a reduced amount of time.

Most MHs are flexible and applicable to a wide range of optimization problems, allowing the design of general-purpose (also called context-independent) optimizers (Campos et al. 2005; Gortazar et al. 2008), which have a long tradition in the field of operations research. Context-independent refers to methods that do not take advantage of problem structure to search the solution space because they have no knowledge of specific characteristics of the objective function. They operate by treating the objective function evaluation as a black box. The solution representation is the only information that could be considered part of the problem context. Nowadays, general-purpose optimization is an issue in several commercial tools such as OptQuest (by OptTek Systems, Inc.) and Evolver (by Palisade Corp.).

Intensification and diversification are two major issues when designing MHs. Diversification generally refers to the ability to visit different parts of the search space when needed, in order to avoid wasting computation time, whereas intensification concerns the exploitation of the collected search experience to obtain high-quality solutions. A search algorithm should strike a tactical balance

between these two sometimes-conflicting goals. Blum and Roli (2003) define an *I&D component* as any algorithmic or functional component that has an intensification and/or diversification effect on the search process. Examples are genetic operators, perturbations of probability distributions, the use of tabu lists, or changes in the objective function. Thus, I&D components are operators, actions, or strategies of MHs.

Local search (LS) algorithms start from one initial solution and iteratively try to replace the current solution by a better one in an appropriately defined neighborhood of the current solution (they may be categorized as trajectory MHs (Blum and Roli 2003), because the search process performed by these methods is characterized by a trajectory in the search space). LS algorithms may effectively and quickly explore the basin of attraction of optimal solutions, finding an optimum with a high degree of accuracy and within a small number of iterations, i.e., they show a clear intensification trend. Despite of their distant origins (Dunham et al. 1963), LS methods remain the object of study of many researchers (Boros et al. 2007; Brimberg et al. 2008; Fournier 2007). The main interest on these algorithms comes from the fact that they are used as intensification component of specific MHs (*LS-based MHs*) that are state-of-the-art for many optimization problems. LS-based MHs settle in the *Multi-start Methods* framework (Marti 2003; Martí et al. 2009) because they intend to exploit a local search procedure, by applying it from multiple initial solutions. LS-based MHs include, among others, Random Multi-start LS (Boender et al. 1982; Marti 2003), Greedy Randomized Adaptive Search Procedures (Resende and Ribeiro 2003), Ant Colony Optimization (Dorigo and Stützle 2004), Iterated LS (ILS) (Lourenço et al. 2003), Variable Neighborhood Search (VNS) (Hansen and Mladenović 2002), and Scatter Search (Laguna 2003).

Genetic Algorithms (GAs) (Goldberg 1989; Holland 1975) are population-based MHs that mimic the metaphor of natural biological evolution. These algorithms have recently received increased interest because they offer practical advantages to researchers facing difficult optimization problems (they may locate high performance regions of vast and complex search spaces). Other advantages include the simplicity of the approach, their flexibility, their robust response to changing circumstances, and the existence of public libraries that implement them, such as ECJ[1] or JCLEC (Ventura et al. 2008), which is included in the KEEL data mining library (Alcalá-Fdez et al. 2009).

Precisely, the flexibility offered by the GA paradigm allows specialized models to be obtained with the aim of providing intensification (García-Martínez and Lozano 2008; Lozano and García-Martínez 2010). Some components of GAs may be specifically designed and their strategy parameters tuned, in order to provide an effective refinement. In fact, several GAs that specialize in intensification have been presented with this purpose (Kazarlis et al. 2001; Lozano et al. 2004; Noman and Iba 2008). They are denominated *local GAs* (LGAs) (García-Martínez and Lozano 2008). Initial studies on these algorithms have shown they present two appealing features:

- *LGAs may improve the performance of classical LS algorithms*. Most LS algorithms lack the ability to follow the proper path to the optimum on complex search spaces. This difficulty becomes much more evident when the search space contains very narrow paths of arbitrary direction, also known as *ridges*. That is due to most LS algorithms attempt successive steps along orthogonal directions that do not necessarily coincide with the direction of the ridge. However, it was observed that LGAs are capable of following ridges of arbitrary direction in the search space regardless of their direction, width, or even, discontinuities (Kazarlis et al. 2001).
- *LGAs may help classical LS-based MHs to improve their behavior*. LGAs may easily assume the role of LS operator in LS-based MHs (i.e., they may be employed as intensification components), obtaining, in this way, a new class of MHs, which may be called LGA-based MHs. Most examples appeared in the literature are approaches of Memetic Algorithms (Gang et al. 2007; Kazarlis et al. 2001; Lozano et al. 2004; Mutoh et al. 2006; Noman and Iba 2008; O'Reilly and Oppacher 1995; Soak et al. 2006) and Random Multi-start LS (García-Martínez and Lozano 2008; García-Martínez et al. 2006). Nevertheless, the outstanding role played by LS-based MHs at present justifies the investigation of new alternative LGA-based MH approaches, being able to enhance their performance. In this way, the study of LGA-based MHs is, nowadays, an insightful line of research.

This paper focuses on a recent approach of LGA, *Binary-coded Local Genetic Algorithm* (BLGA) (García-Martínez and Lozano 2008; García-Martínez et al. 2006). It is a GA with binary representation that applies a crowding replacement method in order to favor the formation of *species* occupying the *niches* of the fitness landscape. Then, BLGA performs an oriented LS process to a new solution, in the niches close to that solution.

Our main objective is to provide new results and insights on the application of LGAs as LS operator for LS-based MHs. Specifically, we present an empirical analysis of BLGA as context-independent LS operator for Random Multi-start LS (Boender et al. 1982; Marti 2003), ILS

[1] http://cs.gmu.edu/eclab/projects/ecj/.

(Lourenço et al. 2003), and VNS (Hansen and Mladenović 2002). They are particularly interesting, because they allow us to analyze the combination of the intensification power of BLGA with very specific, but general and well-understood, diversification techniques: *random* (Random Multi-start LS), *adjustable* (ILS), and *adaptive* (VNS) mechanisms. On the other hand, these MHs were never considered as basis for designing LGA-based MHs. In our opinion, these results are relevant to forecast the advantages of applying BLGA in other LS-based MHs, such as Ant Colony Optimization or Memetic Algorithms.

The paper is set up as follows. In Sect. 2, we give an overview of the existing research on LGAs. In Sect. 3, we provide a complete description of BLGA. In Sect. 4, we design the empirical framework, describing the test functions and four baseline LS methods. In Sect. 5, we compare the performance of a Random Multi-start LS algorithm using BLGA with other MH approaches based on classical LS methods. We also investigate the way effectiveness and efficiency of BLGA affect the operation of the Random Multi-start LS algorithm. In Sect. 6, we study the benefits of an ILS method that uses BLGA as local optimizer in comparison to other ILS methods based on classical LS methods. In Sect. 7, we analyze the BLGA behavior within a VNS algorithm, putting special attention on examining the way it couples with the underlying diversification strategy in VNS that attempts to renew, at every iteration, the best solution so far. In Sect. 8, we develop an empirical study comparing previous algorithms when they have much computation resources at their disposal. Finally, in Sect. 9, we provide the main conclusions of this work and examine future research lines.

2 Local genetic algorithms

GAs (Goldberg 1989; Holland 1975) rely on the concept of a *population* of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as crossover, mutation, and selection to evolve toward increasingly better fitness values of the individuals.

Traditionally, GAs have been applied following two different ways:

- *As autonomous MHs*, i.e. carefully designing their components to provide both diversification and intensification with the aim of obtaining reliable and accurate solutions at the same time.
- *As diversification algorithms* that are combined with LS procedures, forming hybrid MHs (Talbi 2002). An example are *Memetic Algorithms* (Krasnogor and Smith 2005; Moscato 1999).

However, due to the flexibility of the GA architecture, it is also possible to design GA models specifically providing intensification (Lozano and García-Martínez 2010). In fact, several studies use carefully designed GA models with the unique purpose of obtaining accurate solutions (Kazarlis et al. 2001; Lozano et al. 2004; Noman and Iba 2008). These GA models are called LGAs (García-Martínez and Lozano 2008). Next, we review some outstanding LGA models presented in the literature.

Earlier LGA approaches are GAs with enhanced intensification abilities that work together with other independent GAs in a *cooperative manner within a distributed framework* (where a migration mechanism produces a chromosome exchange between them). For example, Potts et al. (1994) use four GAs, denoted as *species I–IV*, which apply different mutation probabilities. Species IV is a LGA that attempts to achieve high intensification by using a low mutation probability. Tsutsui et al. (1997) combine an *explorer* GA and an *exploiter* one. The former explores the search space, whereas the second one searches the neighborhood of the best solution obtained so far. The exploiter GA uses a *fine-grained mutation* and a population with half the size of the explorer one. Finally, Herrera and Lozano (2000) propose to combine several real-coded GAs that apply different real-parameter crossover operators. These operators are differentiated according to their associated exploration and exploitation properties and the degree thereof (the distance between offspring and parents is gradually different from one GA to another). In this case, the exploitative real-coded GAs may be clearly categorized as LGAs.

Later, different authors stressed the convenience of employing LGAs as LS procedures for *Memetic Algorithms*. These algorithms combine a master Evolutionary Algorithm in charge of the global search with a LS operator, which is executed within the Evolutionary Algorithm run, looking for a synergy that takes benefits from both. The classic scheme of Memetic Algorithms (Krasnogor and Smith 2005; Moscato 1999) applies the LS procedure to the solutions obtained by the Evolutionary Algorithm with the aim of improving the accuracy of the population members. Several Memetic Algorithms have been presented that use *micro* GAs (GAs with a small population and short evolution) to refine the members of the population:

- The micro GA presented in Kazarlis et al. (2001) is a GA with five individuals that encode perturbations of a solution given by the Evolutionary Algorithm.
- The one in Lozano et al. (2004) is a crossover hill-climbing operator that is specifically designed to tackle real parameter optimization. This operator is a micro selecto-recombinative real-coded GA that maintains a pair of parents (one parent is the solution to be refined

and the other is the current best solution in the population) and performs repeatedly crossover on this pair until some number of offspring is reached. Then, the best offspring is selected and replaces the worst parent, only if it is better. The key idea of crossover hill-climbing operator is to take advantage of the self-adaptive ability of some crossover operators for real coding, which sample offspring according to the parent distribution without any adaptive parameter, to induce an effective local tuning.

- Other researchers have extended this LGA model (Wang et al. 2009). In particular, in Noman and Iba (2008), an adaptive crossover hill-climbing operator, which adaptively adjusts the length of the refinement process, is proposed for Memetic Algorithms based on Differential Evolution.
- Recent micro GA models embedded in Memetic Algorithms may be found in Mutoh et al. (2006) and Soak et al. (2006).

The feature that makes these LGA models well suited as operators for Memetic Algorithms is that they allow high-quality solutions to be reached with few fitness function evaluations. Other LGA approaches have been proposed as subordinate algorithm for Memetic Algorithms. In Gang et al. (2007), a Memetic Algorithm is presented for the traveling salesman problem, which includes a LGA aimed at refining partial subtours within the solutions of the master algorithm.

Nowadays, researchers notice that LGAs may be incorporated into other LS-based MHs, playing the same role as the LS component but more satisfactorily. This methodology stands for a prospective line of research to design *integrative hybrid MHs* (Raidl 2006; Talbi 2002) that was not very explored in the past. We have only found two examples in the literature: *Binary-coded Local GA* (García-Martínez et al. 2006; García- Martínez and Lozano 2008), described in the next section and *Global and Local Real-coded GAs* (García-Martínez et al. 2008). In the latter case, the authors propose a procedure that determines female and male parents in the population of real-coded GAs that apply *parent-centric crossover operators*. This procedure makes possible the design of Global Real-coded GAs and Local Real-coded GAs (i.e. LGAs), which are differentiated according to the considered number of female members. Furthermore, they combine these GA models according to the *GA then LS* idea (Chelouah and Siarry 2003; Harada et al. 2006): first, they run the Global Real-coded GA during a determinate percentage of the available evaluations, and then, they perform the Local Real-coded GA.

Finally, there is also promising progress on the design of Evolutionary Algorithms for intensification from *Estimation of Distribution Algorithms* (Lima et al. 2006; Sastry and Goldberg 2004), *Ant Colony Optimization Algorithms* (Blum 2002; Kong et al. 2008; Randall 2006), and *Evolution Strategies* applying covariance matrix adaptation (CMA-ES) (Auger and Hansen 2005). All these models can be gathered into a new framework called *Local Evolutionary Algorithms*. The interested reader is also referred to Lozano and García-Martínez (2010) for an overview on Evolutionary Algorithms specializing in intensification and diversification.

3 Binary-coded local genetic algorithm

In this section, we describe BLGA, a recent LGA approach that may be used as context-independent LS in LS-based MHs. It is a steady-state GA (Sywerda 1989; Whitley 1989) that inserts one single new member into the population (P) at each iteration. It uses a *crowding replacement method* (restricted tournament selection (Harik 1995)) in order to force a member of the current population to perish and to make room for the new offspring. It is important to know that the selected replacement method favors the formation of *species* in P occupying the *niches* of the fitness landscape. In nature, a niche is an environmental subspace that can support different types of life (species, or organisms). In a multimodal optimization domain, each peak can be thought of as a niche, and solutions that locate in that region form a species. Therefore, species consist of similar solutions populating promising regions of the search space (see Fig 2).

BLGA performs LS by orienting the search in the nearest niches to an external chromosome, the leader chromosome (C_L). In particular, it iteratively crosses over C_L with individuals of the population belonging to species close to C_L, and then, the best solution between C_L and the offspring becomes the new leader solution, and the other one is inserted in the population by means of the replacement method (see Fig. 1).

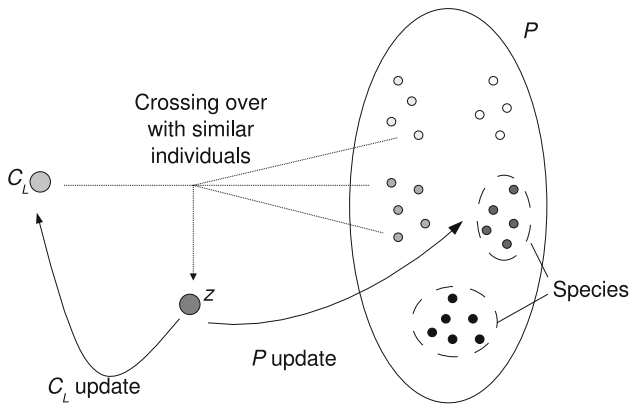

Fig. 1 General schema of BLGA

An outstanding feature of BLGA is that it performs LS by describing a *trajectory* in the search space, as classical LS procedures do. Most LS algorithms follow a *hill-climbing* paradigm; they commence from a single solution and, at each step, a candidate solution is generated using a move operator of some sort. They simply move the search from the current solution to a candidate solution if the candidate has better fitness. The basic idea of BLGA is to use hill-climbing as the move accepting criterion of the search and crossover as the move operator. This scheme of LS based on crossover was first suggested by Jones (1995) and O'Reilly and Oppacher (1995), and it has been followed to obtain different LGA approaches (Lozano et al. 2004; Noman and Iba 2008). The main novelty of BLGA concerns the acquisition of information about the location of the best search regions (by favoring the formation of species), which is then employed to generate individuals around C_L by means of the crossover operator.

An important aspect when using BLGA in LS-based MHs is that P should undergo initialization only once, at the beginning of the corresponding MH run, and not at every invocation as LS operator. This way, BLGA may form stable species and employ accumulated search experience from previous refinements to enhance future ones. In the following sections, we explain, in detail, the main components of BLGA.

3.1 General scheme of BLGA

Let us suppose that a particular LS-based MH applies BLGA as LS procedure. When this MH calls BLGA to refine a particular solution, BLGA will consider this solution as C_L and then, the following steps are carried out:

1. *Mate selection*: m chromosomes $(Y^1, Y^2, \ldots, Y^m)$ are selected from the population applying m times the *positive assortative mating* (Sect. 3.2).
2. *Crossover*: C_L is crossed over with $Y^1, Y^2, \ldots, Y^m$ by the *multi-parent uniform crossover operator with short term memory*, generating an offspring Z (Sect. 3.3).
3. *Update of the leader solution and replacement*: if Z is better than C_L, then C_L is inserted into the population using the *restricted tournament selection* (Sect. 3.4) and Z becomes the new C_L. Otherwise, Z is inserted in the population using the same replacement scheme.

All these steps are repeated until a *termination condition* is attained (Sect. 3.5).

3.2 Positive assortative mating

Assortative mating is the natural occurrence of mating between individuals of similar phenotype more or less often than expected by chance. Mating between individuals with similar phenotype more often is called positive assortative mating and less often is called negative assortative mating. Fernandes and Rosa (2001, 2008) implement these ideas to design two mating selection mechanisms. A first parent is selected by the roulette wheel method and n_{ass} chromosomes are selected with the same method (in BLGA, all the candidates are selected at random). Then, the similarity between each of these chromosomes and the first parent is computed (similarity between two binary-coded chromosomes is defined as the Hamming distance between them). If assortative mating is negative, then the one with less similarity is chosen. If it is positive, the chromosome more similar to the first parent is chosen to be the second parent. BLGA always applies positive assortative mating. In addition, the first parent is always C_L and the method is repeated m times, in order to obtain m parents.

In BLGA, positive assortative mating is in charge of locating the relevant information for the task of optimizing C_L. The selected method identifies this information as the one represented by the closest species to C_L. Figure 2 shows the effect of crossing over C_L with individuals from the nearest species (dark circles). This way, the vicinity of C_L is explored in a biased way (Ishibuchi et al. 2009).

3.3 Multi-parent uniform crossover operator with short term memory

BLGA applies a multi-parent version of *parametrized uniform crossover* (Spears and De Jong 1991; Sywerda 1989) to create offspring in the proximity of C_L. Parametrized uniform crossover creates offspring from two parents, by choosing genes from the first parent with probability p_f. Notice that, applying a high p_f value, the offspring is generated near to the first parent. The multi-parent approach receives C_L and a set of mates $(Y = \{Y^1, Y^2, \ldots, Y^m\})$.

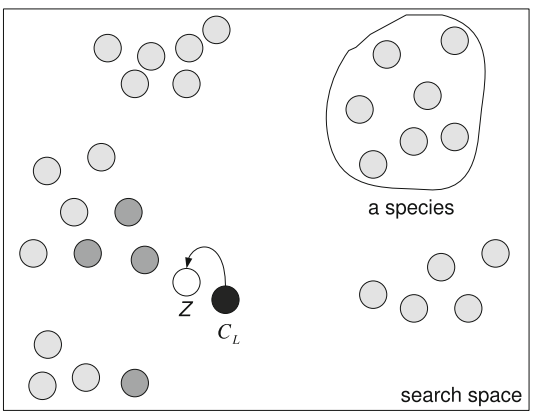

Fig. 2 C_L optimization by BLGA

Then, it creates offspring with genes from C_L, with probability p_f, and genes from members of the set of mates.

In addition, two mechanisms are added to ensure that new genetic material is provided:

- A *short term memory* mechanism ensures that new offspring are sampled in different regions, from those where previous offspring have been sampled, within the proximity of C_L. To do this, the mechanism stores in a memory (M) the genes where there are differences between C_L and any previously generated offspring of the current C_L ($M = \{i : z_i^k \neq c_i^L, \forall Z^k$ sampled offspring from current C_L $\}$). Then, differences between C_L and the new offspring in those genes are avoided. At the beginning, and every time C_L is replaced by a better offspring, M is emptied.
- At last, if Z is equal to C_L (there were no changes), a random gene z_i such that i is not in M, is flipped.

The pseudocode of the crossover operator with short term memory is shown in Fig. 3, where $U(0,1)$ is a random number in $[0,1]$, $RI(1,m)$ is a random integer in $\{1,2,\ldots,m\}$, and p_f is the probability for choosing genes from C_L. In addition, Fig. 4 shows an application example. There, M has been represented as a binary vector where, M_i equal to 1 indicates that the ith gene is in M, and therefore, z_i should be equal to c_i^L; and Z incorporates genes from random mates with low probability, which update M when they produce a change with regards to C_L. Finally, to sum up, it creates the offspring Z as follows:

- z_i is equal to c_i^L for all i in M.
- If i is not in M, then z_i is also equal to c_i^L with probability p_f. Otherwise, z_i is equal to the ith gene of a randomly chosen parent Y^j. The memory is updated if z_i and c_i^L are different.
- Finally, if Z and C_L are equal, then a random chosen gene z_i such that i is not in M, is flipped and the memory is updated.

```
crossover(C_L, Y^1, ..., Y^m, M, p_f){

   For (i = 1, ..., n){

      If (i ∈ M OR U(0,1) < p_f)
         z_i ← c_i^L;

      Else{
         k ← RI(1,m);
         z_i ← y_i^k;

         If (z_i ≠ c_i^L)
            M ← M ∪ i;
      }
   }

   If (Z is equal to C_L){
      i ← RI(1,n) such that i ∉ M;
      M ← M ∪ i;
      flip bit z_i;
   }

   Return Z;
}
```

Fig. 3 Pseudocode of multi-parent uniform crossover operator with short term memory

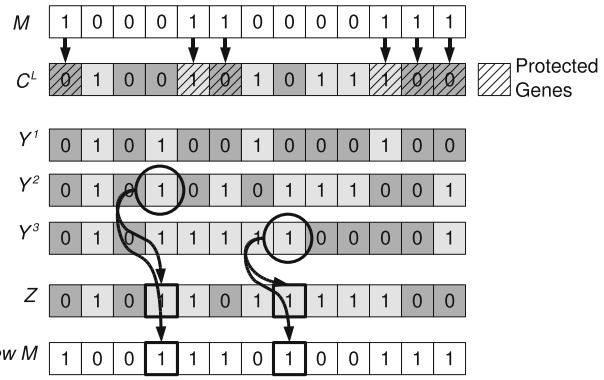

Fig. 4 Crossover application example

3.4 Restricted tournament selection

Crowding methods (Mahfoud 1992) attempt to maintain stable species within the niches of the fitness landscape by means of the replacement procedure as follows: new individuals are more likely to replace existing individuals in the parent population that are similar to themselves based on genotypic similarity. They have been used for locating and preserving multiple local optima in multimodal functions.

BLGA uses *restricted tournament selection* (Harik 1995), a crowding method that replaces the chromosome more similar to the one being inserted in the population, from a set of n_T randomly selected ones (pseudocode is shown in Fig. 5). The application of this crowding method together with the use of a high population size may favor the creation of fitted species in the population. Later, BLGA uses them as relevant information sources for the task of locally optimizing C_L.

3.5 Stop condition

It is important to notice that when all the genes of C_L have been marked by the short term memory (Sect. 3.3), BLGA will not further improve C_L, because crossover operator will create new solutions exactly equal to C_L. Therefore,

```
restrictedTS(P, S){

    G_T ← Select n_T random members from P;
    R ← the member most similar to S from G_T;

    If (S is better than R)
       replace R with S;
}
```

Fig. 5 Restricted tournament selection

this condition will be used to stop the refinement process and to return the resultant C_L to the LS-based MH.

4 Experimental framework

We have carried out different experiments to assess the performance of BLGA as context-independent LS in three different *BLGA-based MHs*: Random Multi-start LS, ILS, and VNS. In order to do this, we have compared BLGA with, to our knowledge, the most representative context-independent LS methods for binary combinatorial optimization problems, and their corresponding LS-based MHs:

- *First LS* (FirstLS) (Blum and Roli 2003; Davis 1991; Dunham et al. 1963) flips a random selected bit of the current solution and accepts the sampled solution if it is better.
- *Best LS* (BestLS) (Blum and Roli 2003; Davis 1991; Dunham et al. 1963) flips each bit of the current solution and chooses the one that makes the best improvement.
- *K-opt LS* (Kopt) (Katayama and Narihisa 2005; Merz and Katayama 2004) is a variant from *Lin-Kernighan LS* (Lin and Kernighan 1973), which has shown being a powerful component of Memetic Algorithms when solving the traveling salesman problem (Nguyen et al. 2007; Ray et al. 2007; Tsai et al. 2004). Kopt tries to change a variable number of bits of the current solution to choose the best sampled one. In every iteration, it produces a sequence of *Dim* solutions (*Dim* is the problem dimension) by 1-flip based sub-moves leading from one solution to another. The flipped bit is the one making the best improvement, even when it results in a worse solution. At the end of the iteration, the best solution in the sequence is adopted as the new current solution for the next iteration. Such process is repeated until no better solution is found. To produce the sequence, the 1-flip based moves are sequentially performed so that each bit of the current solution is flipped no more than once. All *Dim* solutions in the sequence are different and each solution differs one to *Dim* bits from the current solution. Since the best solution is selected from the resulting sequence, the hamming distance between that solution and the current one is variable in each iteration of the algorithm.
- *RandK-opt LS* (RandK) (Katayama and Narihisa 2001; Merz and Katayama 2004) performs the same as Kopt but, the sequence is produced by flipping a random bit that makes an improvement, if it exists, or the one that makes the less detriment, otherwise.

Table 1 Tackled problems

Fnc	Name	Dim	FEs	f^*
1	Deceptive	39	10^5	390
2	Trap(1)	36	10^5	220
3	M-Sat(100,1200,3)	100	10^5	1[a]
4	M-Sat(100,2400,3)	100	10^5	1[a]
5	NkLand(48,4)	48	10^5	1[a]
6	NkLand(48,12)	48	10^5	1[a]
7	PPeaks(50,50)	50	10^5	1
8	PPeaks(50,100)	100	10^5	1
9	PPeaks(50,150)	150	10^5	1
10	PPeaks(50,200)	200	10^5	1
11	PPeaks(50,250)	250	10^5	1
12	PPeaks(100,100)	100	10^5	1
13	BQP(50)	50	10^5	2,098[b]
14	BQP(100)	100	10^5	7,970[b]
15	BQP(250)	250	10^5	45,607[b]
16	BQP(500)	500	10^6	116,586[b]
17	Maxcut(G10)	800	10^6	2485.08[e]
18	Maxcut(G12)	800	10^6	621[d]
19	Maxcut(G17)	800	10^6	Not known
20	Maxcut(G18)	800	10^6	1063.4[c]
21	Maxcut(G19)	800	10^6	1082.04[e]
22	Maxcut(G43)	1,000	10^6	7,027[d]

[a] 1 is the maximum possible fitness value, however, there may not exist any optimal solution with that fitness value, depending on the current problem instance

[b] Best known values presented in Beasley (1998)

[c] Upper bounds presented in the literature

[d] Upper bounds presented in the literature

[e] Best known values presented in Helmberg and Rendl (2000)

To assess the performance of BLGA as a context-independent LS operator, we will carry out experiments on a test suite composed by 22 binary optimization problems of different nature, selected from the literature (Appendix 1). Table 1 shows the name of the tackled problems, their dimension, maximum number of evaluations allowed, and fitness value of the global optimum. All of them have been formulated as maximization problems.

We should indicate that one advantage of LS procedures over other heuristics is that, when solving some problems, the search space can be searched very efficiently: instead of

calculating the objective value of a new sampled solution, it is sufficient to calculate the difference Δf with regards to the fitness of the current solution by utilizing problem-specific knowledge. Δf can often be computed much faster than the objective value of the new solution. For example, in the unconstrained binary quadratic programming problem, the calculation of Δf takes time $O(n)$, while the calculation of the fitness takes $O(n^2)$. Despite of this, optimization can be performed on several of the tackled problems [and many others (Merz 2001)], we will not apply them in our experiments because they use problem-specific knowledge, i.e., LS procedures applying them are not context-independent. In future works, we intend to study the application of BLGA on specific problems making use of this fitness calculation technique. This is possible because multi-parent uniform crossover operator with short term memory, applying a high p_f value, creates offspring near the first parent, i.e., few bits are changed.

In most real-world optimization problems, the evaluation of a solution requires a simulation process that is usually very time-consuming, i.e., solution evaluation is the bottleneck of the search process. For that reason, in order to perform a fair comparison between different algorithms as context-independent optimizers, we will run every algorithm with the same budget of fitness evaluations. In general, each run of an algorithm (LS-based MH with a specific LS method) on a test function, will perform 10^5 or 10^6 fitness evaluations according to the tackled problem (see Table 1) (10^6 evaluations have been allowed for problems where significant improvements still occur, in all the algorithms, after the first 10^5 evaluations). The performance measure is the average of the best fitness value found over 50 independent runs.

Non-parametric tests can be used for comparing the results of different search algorithms (Garcia et al. 2008, 2009). Given that the non-parametric tests do not require explicit conditions for being conducted, it is recommended that the sample of results are obtained following the same criterion, which is, to compute the same aggregation (average, mode, etc.) over the same number of runs for each algorithm and problem.

We have considered the following procedure based on non-parametric tests to analyze the experimental results:

1. Application of *Iman and Davenport* test to ascertain whether there are significant statistical differences among the algorithms in a certain group.
2. If differences are detected, then *Holm* test is employed to compare the best algorithm (control algorithm) against the remaining ones.
3. Utilization of *Wilcoxon* matched-pairs signed-ranks test to compare the best algorithm with those for which Holm test does not find significant differences.

We explain, in detail, these statistical tests in Appendix 3.

The parameter setting of BLGA is the following: *Population size* and n_T are set to 500 individuals and 15, respectively, because these values favor the formation of species in the population. The parameter n_{ass} is set to 5 as it is usually done in papers applying assortative mating (Fernandes and Rosa 2001; García-Martínez et al. 2008). Regarding m, which sets the number of mates to be crossed over with C_L, and p_f, which controls the distance between offspring and C_L, we performed an empirical study on a subset of previous problems, applying BLGA from random starting points with every combination of the values $\{2, 5, 10, 15\}$ for m and $1 - \{1, 2, 4, 7, 10, 15\}/Dim$ for p_f. Though this experiment did not reveal significant differences between the performances of the parameter settings tested, the values 10 for m and $1 - 7/Dim$ for p_f seemed to perform well on most of the used test functions.

5 BLGA-based random multi-start local search

Random multi-start LS (RMLS) (Boender et al. 1982; Marti 2003) is the easiest LS-based MH. It restarts a LS method a given number of times, or until a stop condition is fulfilled, from different initial solutions and the best local optimum reached is returned.

In this section, we focus our attention on a RMLS method that uses BLGA as context-independent LS operator. In particular, we compare its performance with the one of other RMLS algorithms based on classical LS methods and investigate the way the effectiveness and efficiency of BLGA affect the operation of the RMLS algorithm. We have implemented five RMLS methods, named RMLS–{First, Best, Kopt, RandK, BLGA} according to the applied LS algorithm. We have forced initial solutions to be the same for all the RMLS algorithms. Table 15, in Appendix 2, shows the results of the algorithms when tackling each test function.

We should point out that the original idea of RMLS is to perform a number of independent LS invocations that have not any type of relation each other. That is not really true for RMLS–BLGA: since BLGA population is not reinitiated before every BLGA invocation, it accumulates information that may be employed for subsequent invocations.

5.1 RMLS–BLGA versus RMLS algorithms with classical LS methods

We applied a non-parametric statistical test to detect whether there exist significant differences among the results of RMLS–BLGA and the ones of the RMLS algorithms that use classical LS methods.

Table 2 shows the Iman–Davenport statistic (see Appendix 3) and its critical value at the 5% level when the average rankings [computed by the Friedman test (Appendix 3)] of the RMLS methods on the tackled problems are compared. We may observe the existence of significant differences among the results because the statistic value is greater than the critical one (2.48).

Attending to this result, we compare the best ranked algorithm (RMLS–BLGA) and the other RMLS methods by means of a post-hoc statistical analysis. Table 3 shows the average rankings of the RMLS algorithms (low rankings are better) and Holm procedure (columns i, p-value, and α/i) at 0.05 level of significance (the best ranked algorithm, highlighted with the * character, is the control algorithm). Last column indicates whether the equality hypothesis are rejected (R) (the control algorithm performs significantly better than the corresponding algorithm) or not rejected (N) (the corresponding algorithm might perform equivalently to the control algorithm).

The results presented in Table 3 reveal that RMLS–BLGA achieves the best ranking, and Holm test confirms that there are significant differences between the performance of RMLS–BLGA and the other RMLS algorithms. In this way, we may conclude that BLGA allows RMLS to improve its results as general-purpose optimizer with regards to the use of other classical LS methods.

5.2 Behavior of BLGA in the RMLS algorithm

Next, we attempt to discover those behavioral characteristics that allow BLGA to decisively affect the RMLS performance. In particular, we attempt to see whether BLGA may really employ accumulated search experience from previous refinements to enhance future ones. In order to do this, we investigate the way BLGA behaves throughout the run, when it is performed in the RMLS framework, in terms of:

- *Effectiveness*: we study the quality of the solutions reached by the refinements that each LS technique performs.
- *Efficiency*: we compare the number of evaluations that each LS procedure consumes to perform a refinement.

We have considered a fixed test set of 50 random initial solutions (S_{is}) and studied the results of using BLGA to refine them at different stages of the run of a RMLS that performs LS on 250 randomly chosen solutions (training set). Every 50 refinements in the RMLS algorithm, BLGA is applied to the solutions in S_{is}, and the average final fitness value reached and the average number of consumed evaluations are recorded. Figures 6 and 7 display these measures, respectively, for one run on two different test problems, BQP(250) and Maxcut(G17). The results obtained by similar RMLS algorithms with classical LS methods are included as well. We should point out that solutions resulting from the local tuning of the individuals in S_{is} are never introduced in the BLGA population, i.e., these individuals are uniquely used to *test* the performance of BLGA.

Taking into consideration Figs. 6 and 7, we may make the following comments:

- The performance of BLGA on the solutions in S_{is} becomes better as RMLS–BLGA processes more solutions (see Fig. 6). This fact indicates that the knowledge BLGA acquired (forming species around promising niches) may be used, in a fruitful manner, to guide the local tuning of new solutions.
- The trend for improvement exhibited by BLGA (Fig. 6) does not appear when using other LS methods. Just as we expected, since they do not incorporate any memory method, their behavior is not affected by decisions made previously. In fact, in general, they show a similar behavior throughout the whole execution. We may only remark the changeable conduct of RMLS–First for BQP(250). In this LS algorithm, the positions to be flipped are examined according to a random ordering and thus, its behavior may become very different, even when it is applied to the same solutions at different moments (which may be clearly seen on this problem).
- Kopt and RandK LS procedures achieve the best results according to effectiveness (Fig. 6), but they consume too much evaluations per refinement (Fig. 7). We will conduct another experiment later, in order to ascertain if this high consumption may be the reason for RMLS–Kopt and RMLS–RandK to be unable to achieve as good results as RMLS–BLGA (Table 3).

Table 2 Iman–Davenport test on the RMLS algorithms

Statistic	Critical value
9.518	2.48

Table 3 Ranking and Holm procedure on the RMLS algorithms

i	Algorithm	Ranking	p-value	α/i	Result
4	RMLS–Best	4.136	3.378e-7	0.0125	**R**
3	RMLS–RandK	3.341	5.981e-4	0.017	**R**
2	RMLS–First	3	0.007	0.025	**R**
1	RMLS–Kopt	2.818	0.019	0.05	**R**
*	RMLS–BLGA	1.705			

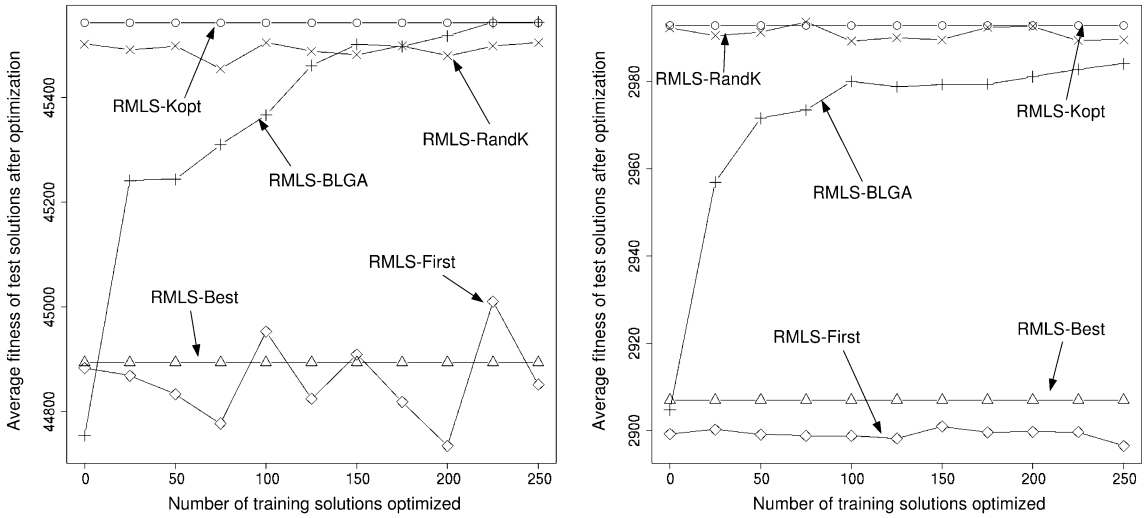

Fig. 6 Fitness evolution for BQP(250) (*left*) and Maxcut(G17) (*right*)

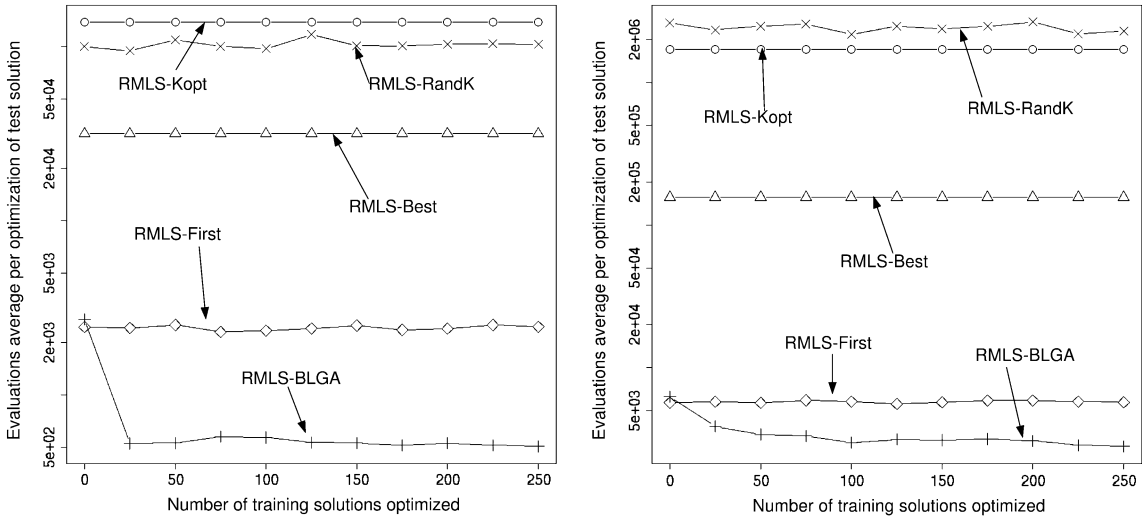

Fig. 7 Evolution of the evaluations consumption for BQP(250) (*left*) and Maxcut(G17) (*right*)

- BLGA starts requiring a number of evaluations per refinement that is similar to the one of FirstLS (see Fig. 7). Later, BLGA reduces it throughout the run, while classic LS methods keep a similar consumption of evaluations. At the early stages, BLGA population does not possess useful information about the search space and the refinement process is guided at random, in a similar way as FirstLS does. Afterwards, BLGA drives LS towards promising zones that were previously located, speeding up convergence considerably and causing the search process to be more efficient.

The employment of an efficient LS algorithm is primordial to ensure that the RMLS algorithm may achieve an acceptable level of *reliability*. When a limited number of evaluations are available, the use of an efficient LS algorithm may allow a high number of restarts to be accomplished, favoring the exploration of the search space. Next, we count the number of times the initially analyzed RMLS algorithms (Sect. 5.1), which considered a maximum number of evaluations of 10^5 or 10^6 per execution, make a restart, i.e., the number of initial solutions that were refined within the available number of evaluations. Table 4 outlines the value of this measure for the RMLS algorithms on each problem. We have included a column with the dimension of the problem (second column), which decisively affects the number of restarts of the algorithms. In addition, we have highlighted, in bold face, the highest values for each row (i.e., the highest number of restarts for each problem).

Several comments are worth being mentioned from Table 4:

Table 4 Number of restarts of RMLS with each LS method

Problem	Dim	Best	First	Kopt	RandK	BLGA
1	39	238.8	820.9	43.1	46.3	**1,331.5**
2	36	213.4	870.7	53.3	46.2	**1,580.2**
3	100	33.6	179.4	5.8	4.9	**411.1**
4	100	30.3	158.1	5.4	5.3	**407.1**
5	48	155.3	535.6	24.6	20.5	**878.5**
6	48	293.8	693.5	26.5	22.5	**1,006.2**
7	50	106.1	555.0	34.5	29.0	**940.0**
8	100	25.2	228.5	9.0	8.0	**348.2**
9	150	11.0	137.6	4.3	4.0	**190.3**
10	200	6.1	96.7	2.8	2.0	**120.6**
11	250	4.0	73.9	1.8	2.0	**88.5**
12	100	25.9	229.5	8.7	8.0	**306.8**
13	50	103.8	442.5	23.4	20.2	**1,096.9**
14	100	23.5	159.9	5.6	6.2	**459.7**
15	250	3.9	41.5	1.3	1.5	**182.0**
16	500	8.6	151.0	1.9	2.3	**905.1**
17	800	4.9	86.0	1.0	1.0	**471.7**
18	800	8.1	207.3	1.0	1.0	**326.8**
19	800	7.0	176.4	1.0	1.1	**329.9**
20	800	6.0	149.8	1.0	1.0	**457.1**
21	800	6.0	149.4	1.0	1.0	**457.0**
22	1,000	3.4	82.8	1.0	1.0	**359.4**

- RMLS–BLGA was always the algorithm that refined more solutions, within the maximum number of evaluations for every problem. The ability of BLGA to converge faster throughout the run of RMLS (observed in Fig. 7) allows it to be finally the most efficient algorithm on all the test problems and to improve the reliability of its results.
- RMLS–Kopt and RMLS–RandK conducted too few restarts. When the size of the problem is high (dimensions higher than 200), they were not able to complete one or two refinements (it is very probable that the limit of evaluations is reached before they start the second or third refinement). Only for problems with low dimensions (50 or less variables), they might make a significant number of restarts. For the case of Kopt, though the expensive computational cost of each refinement was rewarded by obtaining promising final results (RMLS–Kopt is the algorithm with best ranking after RMLS–BLGA in Table 3), the effectiveness-efficiency tradeoff does not yield a proper balance for RMLS to obtain as good results as RMLS–BLGA does under the running conditions assumed.

We may conclude that BLGA is the most appropriate context-independent refinement method for RMLS, because it combines two determinant features (which derive from its ability to retain and exploit valuable information about search space): (1) it is the most efficient LS algorithm because it consumes few evaluations per refinement, and (2) it is able to reach a promising level of effectiveness throughout the run (Fig. 6). The union of these two prominent aspects allow RMLS–BLGA to keep a profitable balance between intensification and diversification, offering two main advantages at the same time: better reliability and accuracy. This kind of balance is not achieved by the other LS techniques. For example, probably, the low degree of efficiency associated with Kopt causes RMLS–Kopt to be incapable of attaining the necessary reliability to outperform RMLS–BLGA (Table 3).

6 BLGA-based iterated local search

Essential idea of ILS (Hoos and Stützle 2004; Lourenço et al. 2003) is to perform a biased, randomized walk in the space of locally optimal solutions instead of sampling the space of all possible candidate solutions. This walk is built by iteratively applying first a *perturbation* to a locally optimal solution, then applying a LS algorithm, and finally using an acceptance criterion, which determines to which locally optimal solution the next perturbation is applied. Despite its simplicity, it is at the basis of several state-of-the-art algorithms for real-world problems.

A high-level description of ILS as it is described in Lourenço et al. (2003) is given in Fig. 8. The algorithm starts by applying LS to an initial solution and iterates a procedure where a perturbation is applied to the current solution S^* in order to move it away from its local neighborhood; the solution so obtained is then considered as initial point for a new LS processing, resulting in another locally optimal solution S_{LS}. Then, a decision is made between S^* and S_{LS} to decide from which solution the next iteration continues.

The perturbation operator is a key aspect to consider, because it allows ILS to reach a new solution from the set of local optima by escaping from the basin of attraction of the previous local optimum. The perturbation is usually nondeterministic in order to avoid cycling. For example, for the case of binary problems, the *standard perturbation*

```
Iterated Local Search
    1. S_0 ← GenerateInitialSolution()
    2. S* ← LocalSearch(S_0)
    3. while (termination conditions not met) do
        4. S_P ← Perturbation(σ_p, S*, history)
        5. S_LS ← LocalSearch(S_P)
        6. S* ← AcceptanceCriterion(S*, S_LS, history)
    7. return S*
```

Fig. 8 Pseudocode algorithm for ILS

operator flips the bits with a fixed probability. Its most important characteristic is the perturbation strength (σ_p), roughly defined as the amount of changes made on the current solution. The perturbation strength should be large enough such that the LS does not return to the same local optimum at the next iteration. However, it should not be too large; otherwise the search characteristics will resemble those of a RMLS algorithm.

In this section, we experimentally show the benefits of an ILS method that uses BLGA as context-independent local optimizer in comparison to other ILS algorithms based on classical LS methods. We have implemented several ILS algorithms denoted as ILS-σ_p -{First, Best, Kopt, RandK, BLGA} according to the applied perturbation strength and LS algorithm. The perturbation operator is the standard one, applied on the best solution so far. The experimental setup was described in Sect. 4.

Next, we study the influence of the diversification introduced by the perturbation strength in every ILS algorithm. Then, we compare the best ILS algorithms among themselves.

6.1 Influence of the perturbation strength

In this section, we investigate the influence of σ_p on the performance of the ILS algorithms. In particular, we analyze the behavior of these algorithms when different values for this parameter are considered ($\sigma_p = 0.1$, 0.25, 0.5, and 0.75). Tables 15 and 16, in Appendix 2, show the results of every ILS algorithm when tackling each test problem. For each ILS algorithm, Fig. 9 shows the average ranking obtained by its runs with different σ_p values when compared among themselves.

Two important remarks from Fig. 9 are:

- The best ranked algorithms for ILS-BLGA, ILS-Kopt, and ILS-RandK use $\sigma_p = 0.5$. Due to their effectiveness, BLGA, Kopt, and RandK successfully affront high perturbation strengths, which means that it becomes possible to jump to new unexplored search regions and eventually find (by means of the LS operators) improved solutions. This is essential to guarantee reliability on the ILS search process.
- ILS-First and ILS-Best obtain their best results with $\sigma_p = 0.1$ and 0.25, respectively. This indicates that the behavior of FirstLS and BestLS is better when ILS does not severely disrupt the current solution. However, this may give rise to a serious drawback: ILS may be unable to explore sufficient regions of the search space, being unable to obtain reliable solutions in complex problems.

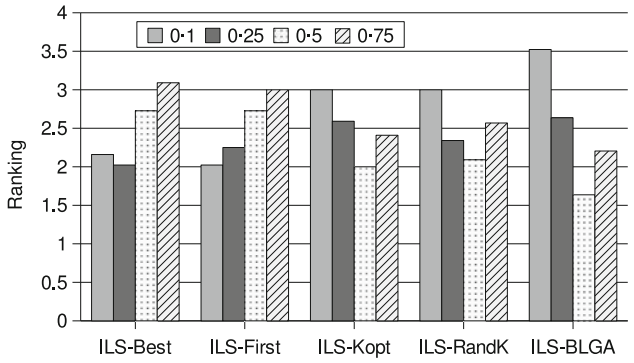

Fig. 9 Rankings obtained by the ILS algorithms with different σ_p values

6.2 ILS-BLGA versus ILS algorithms with classical LS methods

In this section, we compare the results of the best performing ILS algorithms for each LS method: ILS-0.25-Best, ILS-0.1-First, ILS-0.5-Kopt, ILS-0.5-RandK, and ILS-0.5-BLGA. Table 5 shows Iman–Davenport statistic and its critical value at 5% level when comparing the rankings of the selected ILS algorithms. We observe significant differences and proceed performing a post-hoc study. Table 6 shows the average rankings of the algorithms and the Holm test at 0.05 level of significance. The best ranked algorithm (ILS-0.5-BLGA) is the control one. For brevity, intermediate Holm steps (i, p-value and α/i) have been omitted. In addition, we have added a pairwise Wilcoxon test (see Appendix 3) between the control algorithm and the corresponding ones at 0.05 level of significance, in those cases where Holm test has not found statistical differences. The last three columns show R^-, associated with the control algorithm, R^+, with the corresponding algorithm, and the result of this test (the critical value is 65). The result will be '+' if the performance of the control algorithm is statistically better than the one of

Table 5 Iman–Davenport test on the ILS algorithms

Statistic	Critical value
8.052	2.480

Table 6 Holm and Wilcoxon tests on the ILS algorithms

Algorithm	Ranking	Holm	Wilcoxon (65)		
			R^+	R^-	Result
ILS-0.25-Best	4.023	**R**			
ILS-0.5-RandK	3.5	**R**			
ILS-0.1-First	3	**R**			
ILS-0.5-Kopt	2.636	N	62	191	+
*ILS-0.5-BLGA	1.841				

the corresponding algorithm, '−' if the performance of the corresponding algorithm is statistically better than the one of the control algorithm, and '∼' if no significant differences were found.

From Table 6, we clearly notice that ILS-0.5-BLGA obtained improvements with regards to the other ILS algorithms, which are statistically significant. The ability of this algorithm to process high diversity levels ($\sigma_p = 0.5$) may explain, in part, its advantage over ILS-0.25-Best and ILS-0.1-First, whereas the superior efficiency of BLGA over Kopt and RandK (Sect. 5.2) may justify its advantage over the ILS algorithms based on these LS methods. Thus, we may remark that BLGA has arisen as a context-independent LS technique that works suitably in the ILS framework, as attested by the very competitive results compared to other LS methods previously presented in the literature.

7 BLGA-based variable neighborhood search

VNS (Hansen and Mladenović 2002) is an ILS variation (Fig. 8) including a specific perturbation technique that adapts the perturbation strength with the aim of providing the right diversification that lets the next LS refinement to find a solution better than the best one so far. In VNS, an ordered set of neighborhoods, usually nested, is given ($\{N^1, \ldots, N^{\max}\}$) (first neighborhoods relate with weak perturbation strengths and last ones with strong strengths). A new initial solution is sampled from the current neighborhood N^k of the best found solution ($N^k(s^{\text{best}})$) and the LS operator optimizes it. If the new solution is better than the best one so far, the current neighborhood becomes the first one (N^1), otherwise N^{k+1} is selected. Initially, k is set to 1 and a random solution is refined by the LS method.

In this section, we intend to analyze the BLGA behavior within a VNS algorithm, called VNS-BLGA, which applies it as context-independent LS method. In particular, we undertake an experimental study to ascertain if the effectiveness and efficiency associated to BLGA (Sect. 5.2) and its response to different perturbation strengths (Sect. 6.1) are suitable to let VNS to accomplish its main objective: to produce improvements constantly. We will also compare the quality of the solutions obtained by VNS-BLGA with regards to the ones of other VNS algorithms using classical LS methods (VNS-{First, Best, Kopt, RandK}). All VNS methods use the standard perturbation operator to generate the neighborhoods. They will iterate on nine perturbation strengths (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9), returning to 0.1 if there is no improvement after using 0.9, until the maximum number of evaluations is reached. The experimental setup was described in Sect. 4.

7.1 Synergy between VNS and BLGA

VNS uses a *heuristic* to adjust the value of σ_p with the objective of locating, at every iteration and after the LS refinement, a solution better than the best one so far. In this section, we attempt to determine whether the use of BLGA as LS operator for VNS allows its heuristic to act usefully, which will mean that there exists a positive synergy between VNS and BLGA.

In order to investigate the synergy between VNS and BLGA, we have counted the number of *successful restarts* (times the perturbation operator provided a new solution that, after being refined by LS, improved the best solution so far) produced throughout the run of VNS-BLGA, and compared it with the ones for RMLS–BLGA and ILS-0.5-BLGA (the best ILS algorithm based on BLGA found in Sect. 6.1). In addition, we calculated this measure for VNS algorithms based on classical LS operators and compared it with their corresponding versions of RMLS and best ILS algorithms. Table 17, in App B, shows the results (the average over 50 runs).

Figure 10 displays the average ranking of every VNS algorithm according to the number of successful restarts (i.e., to the results in Table 17), when being compared with the other LS-based MHs using the same LS method. In addition, Tables 7 and 8 apply a statistical analysis on these results. Notice that Table 8 just compares the LS-based MHs for which Iman–Davenport test finds statistical differences (BLGA, FirstLS, and BestLS based MHs).

We may remark the following facts:

- VNS-BLGA and VNS-Best obtain the best ranking when confronting them with their versions of RMLS and ILS based on BLGA and BestLS, respectively. In addition, in Table 8, we observe that these improvements are statistically significant. We may conclude that the heuristic embedded in VNS to adapt σ_p results profitable when using BLGA and BestLS, with regards to the utilization of a constant value for σ_p (case of ILS) and the generation of initial solutions at random (case

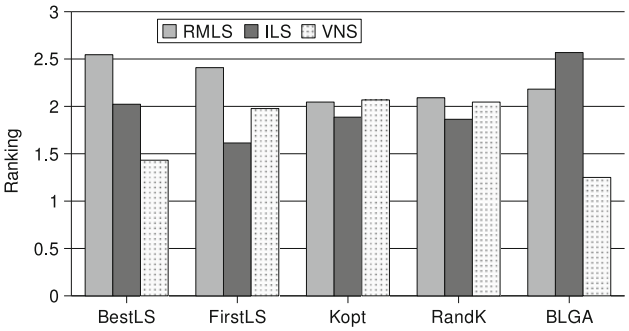

Fig. 10 Rankings of the LS-based MHs according to the number of successful restarts

Table 7 Iman–Davenport tests on the number of successful restarts of the LS-based MHs

MHs	Statistic	Critical value
BLGA-based MHs	17.831	3.220
FirstLS-based MHs	3.958	3.220
BestLS-based MHs	9.454	3.220
Kopt-based MHs	0.208	3.220
RandK-based MHs	0.308	3.220

Table 8 Holm and Wilcoxon tests on the number of successful restarts of the LS-based MHs

LS-based MH	Ranking	Holm	Wilcoxon (65)		
			R^+	R^-	Result
ILS-0.5-BLGA	2.568	**R**			
RMLS–BLGA	2.182	**R**			
*VNS-BLGA	1.25				
RMLS–First	2.41	**R**			
VNS-First	1.977	NR	117.5	135.5	∼
*ILS-0.1-First	1.614				
RMLS–Best	2.545	**R**			
ILS-0.25-Best	2.023	NR	36.5	216.5	+
*VNS-Best	1.432				

of RMLS). Therefore, there exists an adequate synergy between VNS and these two LS techniques.

- VNS-First did not achieve higher number of successful restarts than ILS-0.1-First, which attains the best ranking with regards to this measure. This indicates that a positive synergy among VNS and FirstLS was not produced. In Sect. 6.1, we saw that this LS technique responded poorly to high levels of diversification. In this way, increasing σ_p, when the best solution so far is not improved, does not produce the expected effect.
- VNS was not favored with either the integration of Kopt or RandK. There are two possible reasons for this circumstance: (1) they are refinement methods that consume too many evaluations per refinement (few restarts are accomplished) and then, practically, they do not allow the σ_p adaptation process to have real effects throughout the run and (2) they are very effective, being able to achieve very accurate solutions from the first refinement (for example, we may compare the results of RMLS–Kopt and RMLS–RandK on Maxcut(G10), where they completed only one refinement, with the ones for the other algorithms), which makes difficult to find better solutions at posterior iterations.

7.2 VNS-BLGA versus VNS algorithms with classical LS methods

Finally, we compare the quality of the solutions obtained by VNS-BLGA and the ones by other VNS algorithms using classical LS methods. Table 16, in Appendix 2, has the results of every VNS algorithm for each test problem. Table 9 shows the Iman–Davenport statistic and its critical value at 5% when comparing the average rankings of the VNS algorithms. We notice the existence of significant differences because the statistic is greater than the critical value. Table 10 compares the results of the best ranked algorithm (VNS-BLGA) with the ones of the other VNS algorithms by means of the Holm and Wilcoxon tests.

We clearly see that VNS-BLGA obtains results that are statistically better than the ones of the other VNS algorithms. The prospective synergy that BLGA and VNS produce allowed VNS-BLGA to exhibit overall better performance than the other VNS algorithms. This synergy is possible due to: (1) the effectiveness of BLGA, which allows better solutions to be reached even with high σ_p values and (2) its efficiency, which allows VNS to quickly achieve σ_p values producing improvements (i.e., the number of evaluations wasted by failed inspections is inferior). This became determinant to overtake the other VNS algorithms. Since classical LS methods are less efficient than BLGA, their corresponding VNS methods did not dispose of sufficient evaluations to locate solutions with better quality than the ones found by VNS-BLGA.

8 Analysis on long runs

From previous experiments, we have realized that some of the applied LS methods consume too many evaluations per refinement, hindering the corresponding LS-based MHs to

Table 9 Iman–Davenport test on the VNS algorithms

Statistic	Critical value
13.695	2.48

Table 10 Holm and Wilcoxon tests on the VNS algorithms

Algorithms	Ranking	Holm	Wilcoxon (65)		
			R^+	R^-	Result
VNS-Best	3.841	**R**			
VNS-RandK	4.045	**R**			
VNS-Kopt	3.068	**R**			
VNS-First	2.364	N	62	191	+
*VNS-BLGA	1.682				

perform enough restarts on large problems. This fact makes that those LS-based MHs are unable to work properly for context-independent optimization, when the number of available evaluations is limited.

In this section, we intend to analyze the results and behavior of previous algorithms when they have much more evaluations at their disposal. In particular, previous algorithms will be run on the test functions described in Sect. 4, with a budget of 10^8 function evaluations. Preliminary experiments showed that even MHs based on expensive LS methods performed a *reasonable* (50 approximately) number or restarts, when tackling the largest problems. The performance measure is the average of just 4 runs, because of the extreme computation times of these experiments.

Table 11 shows the Iman–Davenport statistics and their critical values at 5% when comparing the average rankings of the algorithms according to the group they belong. We see that Iman–Davenport test does not find significant differences among the RMLS and the VNS algorithms. This means that those algorithms tend to achieve similar results when there are enough evaluations available. Table 12 shows the results of Holm and Wilcoxon tests on the ILS algorithms, the only ones for which Iman–Davenport test finds statistical differences. We can see that ILS-0.5-RandK and ILS-0.5-Kopt obtain the first and second rankings, respectively. However, the statistical analysis does not find statistical differences between the performances of ILS-0.5-RandK (best ranked algorithm) and ILS-0.5-BLGA. The conclusion is that, MHs based on expensive LS methods are not able to obtain statistically better results than those reached by BLGA-based MHs, even when they have a large number of function evaluations at their disposal.

Table 11 Iman–Davenport tests on results with 10^8 evaluations

MHs	Statistic	Critical value
RMLS algorithms	2.066	2.48
ILS algorithms	2.745	2.48
VNS algorithms	2.13	2.48

Table 12 Holm and Wilcoxon tests on results with 10^8 evaluations

Algorithms	Ranking	Holm	Wilcoxon (65)		
			R^+	R^-	Result
ILS-0.1-First	3.705	**R**			
ILS-0.25-Best	3.364	N	49.5	203.5	+
ILS-0.5-BLGA	2.864	N	96.5	156.5	~
ILS-0.5-Kopt	2.727	N	97.5	155.5	~
*ILS-0.5-RandK	2.341				

Next, we study the behavior of the algorithms along the whole run. Figures 11, 12, and 13 show convergence graphs of each group of algorithms, RMLS, ILS, and VNS, respectively, when tackling all the problems. In order to obtain those graphs, which summarize the behavior of the algorithms on all the problems, we have accomplished the following two steps: (1) taking into consideration the highest and lowest fitness values achieved by all the algorithms on each test problem, we have normalized every result, along the whole run, in the interval $[0, 1]$; (2) then, mean values, over the 22 problems, have been obtained for each algorithm, along the 10^8 evaluations. Notice that evaluation axis is logarithmic scaled.

We can see that, BLGA-based MHs are always able to reach the best results in the range from 10^4 to 10^6 evaluations, which is usually considered as a reasonable range of evaluations when dealing with black-box optimization problems. Besides, from previous studies, we know that those results are statistically different from the ones of the other algorithms. From 10^6 evaluations onward, RandK

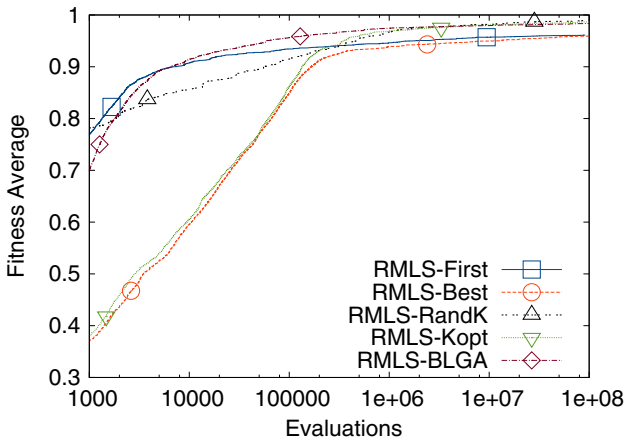

Fig. 11 Convergence graphs for RMLS algorithms

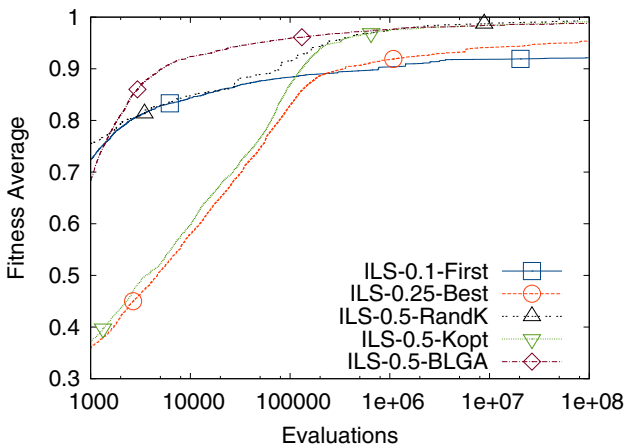

Fig. 12 Convergence graphs for ILS algorithms

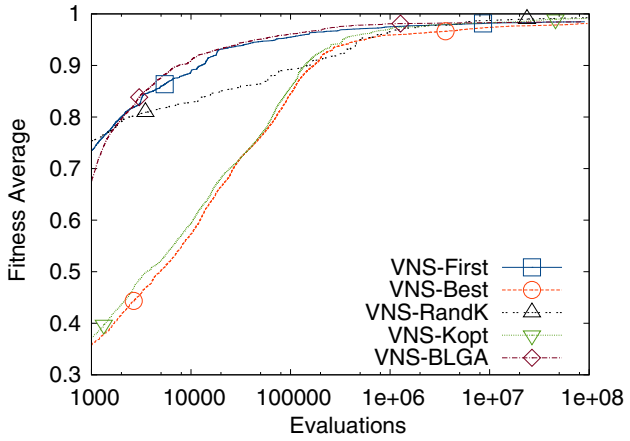

Fig. 13 Convergence graphs for VNS algorithms

and Kopt based MHs usually obtain better results, however, they are not statistically different from the ones of BLGA-based MHs, at least at the level applied in the whole paper (5%). Therefore, the conclusion is that BLGA-based MHs are more appropriate when dealing with black-box optimization problems, where evaluating a new solution requires invoking the objective function, and this number of invocations is limited, than the other algorithms. Otherwise, almost any other studied algorithm performs well.

9 Conclusions

In this paper, different studies were conducted with the aim of investigating the suitability of BLGA as context-independent LS operator for three well-known LS-based MHs: RMLS, ILS, and VNS. In order to do this, the benefits of this novel LS technique in comparison to other LS methods, considered as classical models, were experimentally shown. We realized that:

- BLGA achieved a high level of efficiency, which allowed many restarts to be accomplished. In addition, it showed a profitable intensification power, which was able to cope with strong diversification levels coming from an operator with high perturbation strength (high σ_p value). These two features are possible because BLGA has the ability to guide the LS using knowledge on zones of the search space visited in previous invocations.
- The joint effects of these two desirable properties made possible RMLS, ILS, and VNS to face the conflict between accuracy and reliability in a fruitful manner to enhance their performance with regards to the use of classical LS methods. Then, BLGA arise as an algorithm that shows promise as context-independent LS operator for LS-based MHs.

In addition, we should point out that our extensive study provided, as well, new insights on the behavior of classical LS methods when they are embedded in LS-based MHs tackling black-box optimization problems. In fact, an important remark is that Kopt is a LS procedure able to reach outstanding solutions regardless the used initial solutions. Its main disadvantage comes by the fact that it consumes too much evaluations per refinement, when it is applied as a context-independent optimizer. This unbalanced effectiveness-efficiency tradeoff makes its application in LS-based MHs inappropriate, when the number of evaluations is limited. Otherwise, when there are enough evaluations available, almost every studied algorithm perform well.

The research line focused in this paper is indeed worthy of further studies. We are currently extending our investigation to analyze the role of BLGA as LS operator of other LS-based MHs, such as Memetic Algorithms, Greedy Randomized Adaptive Search Procedures, and Ant Colony Optimization algorithms. We also mention that BLGA might be modified in order to imitate other trajectory methods such as Simulated Annealing or Tabu Search. While, at present, BLGA is aimed only at intensification, Simulated Annealing and Tabu Search are complete MHs pursuing a wide space exploration. In the near future, we intend to study the benefits of using GA concepts on these methods.

Acknowledgments This work was supported by Research Projects TIN2008-05854 and P08-TIC-4173.

Appendix 1: A test suite

The test suite that we have used for the experiments consists of 22 binary-coded test problems. They are described in the following sections.

Deceptive problem

In deceptive problems (Goldberg et al. 1989), there are certain schemata that guide the search toward some solution that is not globally competitive. The schemata that have the global optimum do not bear significance and so they may not proliferate during the genetic process. The used deceptive problem consists of the concatenation of k subproblems of length 3. The fitness for each 3-bit section of the string is given in Table 13. The overall fitness is the sum of the fitness of these deceptive subproblems.

Table 13 Deceptive order-3 problem

Chromosomes	000	001	010	100	110	011	101	111
Fitness	28	26	22	14	0	0	0	30

We have used a deceptive problem with 13 subproblems.

Trap problem

Trap problem (Thierens 2004) consists of misleading subfunctions of different lengths. Specifically, the fitness function $f(x)$ is constructed by adding subfunctions of length 1 (F_1), 2 (F_2), and 3 (F_3). Each subfunction has two optima: the optimal fitness value is obtained for an all-ones string, while the all-zeroes string represents a local optimum. The fitness of all other string in the subfunction is determined by the number of zeroes: the more zeroes, the higher the fitness value. This causes a large basin of attraction toward the local optimum. The fitness values for the subfunctions are specified in Table 14 where the columns indicate the number of ones in the subfunctions F_1, F_2, and F_3. The fitness function $f(x)$ is composed of 4 F_3 subfunctions, 6 F_2 subfunctions, and 12 F_1 subfunctions. The overall length of the problem is thus 36. There are 2^{10} optima of which only one is the global optimum: the string with all ones having a fitness value of 220.

$$f(x) = \sum_{i=0}^{3} F_3(x_{[3i:3i+2]}) + \sum_{i=0}^{5} F_2(x_{[2i+12:2i+13]}) + \sum_{i=0}^{11} F_1(x_{24+i})$$

Max-Sat Problem

The satisfiability problem in propositional logic (SAT) (Smith et al. 2003) is the task to decide whether a given propositional formula has a model. More formally, given a set of m clauses $\{C_1, \ldots, C_m\}$ involving n boolean variables $X_1, \ldots, X_n$ the SAT problem is to decide whether an assignment of values to variables exists such that all clauses are simultaneously satisfied.

Table 14 Fitness values of the subfunctions F_i

	0	1	2	3
F_3	4	2	0	10
F_2	5	0	10	
F_1	0	10		

Max-Sat is the optimization variant of SAT and can be seen as a generalization of the SAT problem: given a propositional formula in conjunctive normal form (CNF), the Max-Sat problem then is to find a variable assignment that maximizes the number of satisfied clauses. It returns the percentage of satisfied clauses.

We have used two set of instances of the Max-Sat problem with 100 variables, 3 variables by clause, and 1,200 and 2,400 clauses, respectively. They have been obtained from De Jong et al. (1997). They are denoted as M-Sat(n, m, l), where l indicates the number of variables involved in each clause (3). Each run i, of every algorithm, uses a specific seed (seed$_i$) for generating the M-Sat(n, m, l) instance, i.e. ith execution of every algorithm uses the same seed$_i$, whereas jth execution uses seed$_j$.

NK-landscapes

In the *NK* model (Kauffman 1989), N represents the number of genes in a haploid chromosome and K represents the number of linkages each gene has to other genes in the same chromosome. To compute the fitness of the entire chromosome, the fitness contribution from each locus is averaged as follows:

$$f(s) = \frac{1}{N}\sum_{i=1}^{N} f(\text{locus}_i) \quad (1)$$

where the fitness contribution of each locus, $f(\text{locus}_i)$, is determined by using the (binary) value of gene i together with values of the K interacting genes as an index into a table T_i of size 2^{K+1} of randomly generated numbers uniformly distributed over the interval $[0, 1]$. For a given gene i, the set of K linked genes may be randomly selected or consist of the immediately adjacent genes.

We have used two set of instances of the *NK*-Landscape problem: one with $N = 48$ and $K = 4$, and another with $N = 48$ and $K = 12$. They are denoted as NKLand (N, K). They have been obtained from De Jong et al. (1997). Each run i, of every algorithm, uses a different seed (seed$_i$) for generating the NKLand (N, K) instance, i.e. the ith execution of every algorithm has used the same seed$_i$, whereas the jth execution has used seed$_j$.

P-peak problems

P-peak problem generator (Spears 2000) creates instances with a certain number of peaks (the degree of multimodality). For a problem with P peaks, P bit strings of length L are randomly generated. Each of these string is a peak (a local optima) in the landscape. Different heights can be assigned to different peaks based on various schemes (equal height, linear, logarithm-based, and so on).

To evaluate an arbitrary solution S, first locate the nearest peak in Hamming space, call it $\text{Peak}_n(S)$. Then, the fitness of s is the number of bits the string has in common with $\text{Peak}_n(S)$, divided by L, and scaled by the height of the nearest peak. In case there is a tie when finding the nearest peak, the highest peak is chosen.

We have used different groups of P-Peak instances denoted as PPeaks(P, L). Each run i, of every algorithm, uses a different seed (seed_i) for generating the PPeaks(P , L) instance. Linear scheme have been used for assigning heights to peaks in $[0.6, 1]$.

Max-cut problem

The Max-cut problem (Karp 1972) is define as follows: Let an undirected and connected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E \subset \{(i,j) : 1 \leq i < j \leq n\}$, be given. Let the edge weights $w_{ij} = w_{ji}$ be given such that $w_{ij} = 0$ $\forall (i,j) \notin E$, and in particular, let $w_{ii} = 0$. The max-cut problem is to find a bipartition (V_1, V_2) of V so that the sum of the weights of the edges between V_1 and V_2 is maximized.

We have used 6 instances of the max-cut problem (G10, G12, G17, G18, G19 G43), obtained from Helmberg and Rendl (2000).

Unconstrained binary quadratic programming problem

The objective of the Unconstrained Binary Quadratic Programming (BQP) (Beasley 1998) 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:

$$f(x) = x^t Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j, \quad x_i \in \{0, 1\} \tag{2}$$

We have used four instances with different values for n. They have been taken from the OR-Library (Beasley 1990). They are the first instances of the BQP problems in the files 'bqp50', 'bqp100', 'bqp250', and 'bqp500'. They are BQP(50), BQP(100), BQP(250), and BQP(500), respectively.

Appendix 2: Results

Tables 15 and 16 show the fitness values obtained by the algorithms studied in the empirical analysis in Sects. 5, 6, and 7. Best results of every group of algorithms (RMLS, ILS, and VNS) are boldfaced. Table 17 displays the number of successful restarts obtained by the algorithms in Sect. 7.1. Best results of every group with the same LS method are marked.

Appendix 3: Statistical analysis

In this section, we explain the basic functionality of each non-parametric test applied in this study together with the aim pursued with its use:

- *Friedman test*: Although we will not use this test, because of its conservative undesirably effect, we describe it because it is the basis of the following one. Friedman test is a non-parametric equivalent of test of repeated-measures ANOVA. It computes the ranking of the observed results for each algorithm (r_j for the algorithm j with k algorithms) for each function, assigning to the best of them the ranking 1, and to the worst the ranking k. Under the null hypothesis, formed from supposing that the results of the algorithms are equivalent and, therefore, their average rankings are also similar, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{3}$$

  is distributed according to χ_F^2 with $k-1$ degrees of freedom, being $R_j = 1/N \sum_i r_j^i$, and N the number of functions. The critical values for the Friedman statistic coincide with the established in the χ^2 distribution when $N > 10$ and $k > 5$. In a contrary case, the exact values can be seen in Zar (1999).
- *Iman and Davenport test* (Iman and Davenport 1980): It is a metric derived from the Friedman statistic given that this last metric produces a conservative undesirably effect. The statistic is

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{4}$$

  and it is distributed according to a F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.
- *Holm method* (Holm 1979): If the null hypothesis is rejected in Iman–Davenport test, we can proceed with a post-hoc test. The test of Holm is applied when we want to compare a control algorithm (the one with the best average Friedman ranking) opposite to the remainders. Holm test sequentially checks the hypotheses ordered according to their significance. We will denote the p-values ordered by $p_1, p_2, \ldots$, in the way that $p_1 \leq p_2 \leq \cdots \leq p_{k-1}$. Holm method compares each p_i with $\alpha/(k-i)$ starting from the most significant p-value. If p_1 is below than $\alpha/(k-1)$, the corresponding hypothesis is rejected and it leaves us to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, we continue with the process. As soon as a certain hypothesis cannot be rejected, all the remaining hypotheses are maintained as accepted. The statistic for comparing algorithm i with algorithm j is:

Table 15 Results of the RMLS, ILS-0.1, and ILS-0.25 algorithms with each refinement procedure

Problem	RMLS					ILS-0.1					ILS-0.25				
	BestLS	FirstLS	Kopt	RandK	BLGA	BestLS	FirstLS	Kopt	RandK	BLGA	BestLS	FirstLS	Kopt	RandK	BLGA
1	386	381	**390**	375	380	**390**	387	**390**	373	388	**390**	383	**390**	375	385
2	219	213	**220**	202	**220**	**220**	**220**	**220**	196	**220**	**220**	**220**	**220**	197	**220**
3	0.955	0.958	0.959	0.959	**0.960**	0.957	0.958	0.958	0.958	0.957	0.957	0.959	0.959	0.959	0.958
4	0.935	0.936	**0.937**	**0.937**	**0.937**	0.935	**0.937**	0.936	0.936	0.936	0.936	**0.937**	**0.937**	**0.937**	0.936
5	0.760	0.762	0.771	0.767	**0.774**	0.773	**0.774**	0.769	0.769	0.773	0.773	0.773	0.773	0.769	**0.774**
6	0.742	0.745	**0.763**	0.749	0.739	0.761	**0.768**	**0.768**	0.753	0.763	0.752	0.753	0.764	0.751	0.754
7	**1**	**1**	**1**	0.999	**1**	0.892	0.877	0.959	0.847	0.908	0.996	0.997	0.980	0.958	0.991
8	0.997	**1**	0.993	0.978	**1**	0.871	0.843	0.930	0.830	0.873	0.883	0.908	0.926	0.840	0.924
9	0.990	**1**	0.971	0.948	**1**	0.861	0.857	0.931	0.855	0.883	0.861	0.861	0.923	0.855	0.891
10	0.973	**1**	0.964	0.915	**1**	0.867	0.831	0.922	0.838	0.869	0.867	0.831	0.921	0.838	0.869
11	0.937	**0.999**	0.919	0.877	**0.999**	0.855	0.821	0.884	0.827	0.858	0.855	0.821	0.886	0.827	0.858
12	0.997	**1**	0.990	0.980	**1**	0.881	0.859	0.927	0.861	0.914	0.905	0.929	0.929	0.871	0.958
13	2,094	**2,098**	**2,098**	**2,098**	**2,098**	2,080	**2,098**	**2,098**	2,095	2,093	**2,098**	**2,098**	**2,098**	2,096	**2,098**
14	7,849	7,899	7,938	7,886	**7,946**	7,871	7,915	7,913	7,867	7,887	7,925	7,934	7,940	7,868	7,903
15	45,168	45,545	45,516	45,511	**45,563**	45,277	**45,590**	45,534	45,490	45,520	45,379	45,584	45,534	45,490	45,524
16	114,792	115,939	115,717	**116,251**	115,923	115,422	116,454	115,651	116,211	115,808	115,802	**116,537**	115,680	116,214	115,824
17	1,762	1,819	1,871	1,887	**1,897**	1,829	**1,921**	1,869	1,886	1872	1,823	1,916	1,869	1,886	1,877
18	421	437	**534**	529	508	482	517	**535**	529	522	449	471	**535**	529	526
19	2,920	2,931	2,991	2,987	**3,008**	2,969	2,992	2,991	2,986	3,002	2,940	2,954	2,991	2,986	3,007
20	843	866	916	929	**953**	900	950	920	929	935	885	922	920	929	947
21	759	782	831	842	**869**	810	861	830	839	846	795	844	830	839	855
22	6,410	6,460	6,506	6,548	**6,572**	6,472	**6,582**	6,504	6,545	6,548	6,452	6,546	6,504	6,545	6,560

Table 16 Results of the ILS-0.5, ILS-0.75, and VNS algorithms with each refinement procedure

Problem	ILS-0.5					ILS-0.75					VNS				
	BestLS	FirstLS	Kopt	RandK	BLGA	BestLS	FirstLS	Kopt	RandK	BLGA	BestLS	FirstLS	Kopt	RandK	BLGA
1	386	381	**390**	376	380	382	380	**390**	376	380	389	383	**390**	376	383
2	219	212	**220**	201	**220**	211	205	**220**	206	**220**	**220**	218	**220**	205	**220**
3	0.955	0.957	0.959	0.959	**0.960**	0.953	0.956	0.959	0.959	0.958	0.957	**0.959**	0.958	**0.959**	**0.959**
4	0.935	0.936	**0.937**	**0.937**	**0.937**	0.933	0.935	**0.937**	0.936	0.936	0.936	**0.937**	**0.937**	**0.937**	**0.937**
5	0.761	0.762	0.770	0.767	**0.774**	0.753	0.755	0.770	0.766	0.767	0.769	0.773	0.773	0.768	**0.774**
6	0.743	0.748	0.765	0.749	0.738	0.742	0.749	0.762	0.747	0.736	0.755	0.754	**0.764**	0.751	0.752
7	**1**	**1**	**1**	0.997	**1**	**1**	**1**	0.999	0.995	**1**	**1**	**1**	0.999	0.990	**1**
8	0.997	**1**	0.994	0.975	**1**	0.997	**1**	0.989	0.981	0.999	0.994	**1**	0.976	0.919	**1**
9	0.989	**1**	0.981	0.960	**1**	0.988	0.999	0.975	0.948	0.997	0.981	0.998	0.936	0.855	**0.999**
10	0.980	**1**	0.962	0.913	**1**	0.973	0.997	0.959	0.916	0.997	0.946	0.997	0.921	0.838	**0.999**
11	0.946	0.999	0.918	0.896	**1**	0.946	0.996	0.922	0.894	0.997	0.869	0.994	0.884	0.827	**0.996**
12	0.998	**1**	0.993	0.977	**1**	0.995	0.999	0.991	0.977	**1**	0.994	1	0.978	0.933	**1**
13	2,095	**2,098**	**2,098**	2,095	**2,098**	2,006	2,059	**2,098**	2,094	2,095	**2,098**	**2,098**	**2,098**	2,096	**2,098**
14	7,836	7,897	7,944	7,884	**7,946**	7,699	7,827	**7,946**	7,880	7,920	7,880	7,909	7,934	7,876	**7,951**
15	45,104	45,524	45,534	45,488	45,551	44,880	45,452	45,534	45,484	45,583	45,356	45,573	45,534	45,490	**45,589**
16	114,815	115,890	115,686	116,261	115,862	113,823	115,207	115,638	116,095	116,024	115,475	**116,477**	115,650	116,228	116,079
17	1,764	1,822	1,869	1,886	1,908	1,821	1,916	1,869	1,886	1,893	1,826	**1,920**	1,869	1,886	1,905
18	422	437	**535**	529	510	446	472	**535**	529	528	458	501	**535**	529	523
19	2,920	2,931	2,991	2,986	3,009	2,939	2,953	2,991	2,986	**3,010**	2,953	2,978	2,991	2,986	**3,006**
20	845	868	920	929	**956**	880	924	920	929	946	894	937	920	929	**948**
21	764	781	830	839	**868**	800	842	830	839	859	806	854	830	839	**857**
22	6,410	6,456	6,504	6,545	6,571	6,448	6,551	6,504	6,545	6,571	6,472	6,569	6,504	6,545	**6,572**

Table 17 Number of successful restarts for each LS-based MH

Problem	BestLS			FirstLS			Kopt			RandK			BLGA		
	RMLS	ILS-0.25	VNS	RMLS	ILS-0.1	VNS	RMLS	ILS-0.5	VNS	RMLS	ILS-0.5	VNS	RMLS	ILS-0.5	VNS
1	3.2	**5.5**	5.2	3.8	**8.1**	5.0	0.0	0.0	0.0	1.9	2.2	**2.5**	3.8	3.7	**5.1**
2	3.2	3.6	**4.0**	4.5	**6.9**	6.4	0.0	0.0	0.0	2.2	2.4	**3.0**	5.3	5.3	**6.3**
3	2.5	**5.0**	4.8	3.7	**6.7**	6.2	1.0	1.0	**1.3**	**1.1**	0.9	1.0	6.1	6.1	**7.3**
4	2.1	**4.4**	4.2	4.5	**5.0**	**5.0**	0.9	1.0	**1.2**	1.0	1.2	**1.5**	**6.7**	6.3	6.5
5	4.2	**7.3**	6.9	6.0	**10.0**	8.4	2.5	2.7	**3.0**	2.3	2.5	**2.9**	9.6	9.3	**9.8**
6	5.2	**6.2**	**6.2**	5.4	**8.2**	6.9	3.0	2.8	**3.2**	2.6	**3.1**	2.6	6.5	6.0	**7.1**
7	2.2	1.8	**2.3**	**2.8**	0.2	**2.8**	1.9	**2.0**	1.8	**2.7**	2.4	2.3	3.1	**3.3**	3.2
8	**2.2**	0.1	2.1	**3.0**	0.0	**3.0**	1.5	**1.6**	0.9	1.6	**1.7**	0.7	3.0	2.9	**3.7**
9	**2.0**	0.0	1.3	**3.1**	0.0	2.7	**1.1**	0.9	0.2	**1.1**	1.0	0.0	**3.5**	3.1	3.3
10	**1.1**	0.0	0.7	**3.1**	0.0	2.5	**0.7**	0.5	0.0	0.4	**0.5**	0.0	3.1	2.6	**3.4**
11	**0.8**	0.0	0.2	**3.2**	0.0	2.8	**0.4**	0.3	0.0	0.4	**0.5**	0.0	**3.1**	2.9	**3.1**
12	2.5	0.2	**2.6**	**3.6**	0.0	3.3	1.5	**1.6**	1.1	**1.8**	1.6	0.7	3.7	3.3	**3.9**
13	3.3	3.4	**3.6**	**4.0**	3.8	3.5	0.3	**0.4**	**0.4**	1.3	**1.5**	1.4	**4.2**	3.6	4.0
14	3.2	**4.3**	3.3	4.6	**5.5**	5.4	**1.2**	**1.2**	1.0	**1.6**	**1.6**	**1.6**	6.4	6.7	**7.7**
15	0.9	2.2	**2.4**	3.4	**5.2**	4.3	0.0	0.0	0.0	**0.2**	0.1	0.1	5.6	**5.8**	5.7
16	1.6	5.8	**6.7**	4.6	**11.7**	9.5	0.2	0.3	**0.4**	0.5	0.6	**0.9**	15.3	14.6	**17.3**
17	1.0	2.9	**6.6**	3.9	**18.3**	14.1	0.0	0.0	0.0	0.0	0.0	0.0	26.8	29.5	**39.5**
18	1.3	3.5	**4.7**	4.3	**20.1**	14.7	0.0	0.0	0.0	0.0	0.0	0.0	13.2	14.3	**23.3**
19	1.2	2.9	**5.4**	4.3	**17.5**	11.8	0.0	0.0	0.0	0.0	0.0	**0.1**	20.5	20.5	**32.0**
20	1.4	3.5	**7.8**	4.6	**25.3**	16.7	0.0	0.0	0.0	0.0	0.0	0.0	25.9	28.0	**38.4**
21	1.2	3.4	**7.6**	4.4	**22.8**	17.0	0.0	0.0	0.0	0.0	0.0	0.0	28.1	26.6	**36.8**
22	0.5	1.8	**5.1**	4.0	**22.8**	15.6	0.0	0.0	0.0	0.0	0.0	0.0	30.2	30.1	**45.6**

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6N}} \tag{5}$$

The value of z is used for finding the corresponding probability from the table of the normal distribution, which is compared with the corresponding value of α.

- *Wilcoxon signed rank test*: This is the analogous of the paired t test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between the results of two algorithms. Let d_i be the difference between the performance scores of two algorithms on the ith out of N functions (we have normalized the results on every function to be in $[0, 1]$ according to the best and worst results obtained by all the algorithms). The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + 1/2 \sum_{d_i = 0} \text{rank}(d_i) \tag{6}$$

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + 1/2 \sum_{d_i = 0} \text{rank}(d_i) \tag{7}$$

Let T be the smallest of the sums, $T = min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N degrees of freedom [Table B.12 in Zar (1999)], the null hypothesis of equality of means is rejected.

References

Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrel JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318

Auger A, Hansen N (2005) Performance evaluation of an advanced local search evolutionary algorithm. In: Corne D, Michalewicz Z, McKay B, Eiben G, Fogel D, Fonseca C, Greenwood G, Raidl G, Tan KC, Zalzala A (eds) Proceedings of the IEEE international conference on evolutionary computation, vol 2. IEEE, New York, pp 1777–1784

Beasley JE (1990) OR-library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072. http://people.brunel.ac.uk/mastjjb/jeb/info.html

Beasley JE (1998) Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College

Blum C (2002) ACO applied to group shop scheduling: a case study on intensification and diversification. In: Dorigo M, Di Caro G, Sampels M (eds) ANTS. LNCS, vol 2463. Springer, Heidelberg, pp 14–27

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 35(3):268–308

Boender CGE, Rinnooy-Kan AHG, Stougie L, Timmer GT (1982) A stochastic method for global optimization. Math Program 22:125–140

Boros E, Hammer PL, Tavares G (2007) Local search heuristics for quadratic unconstrained binary optimization (QUBO). J Heuristics 13(2):99–132

Brimberg J, Mladenović N, Urošević D (2008) Local and variable neighborhood search for the k-cardinality subgraph problem. J Heuristics 14(5):501–517

Campos V, Laguna M, Martí R (2005) Context-independent scatter and tabu search for permutation problems. INFORMS J Comput 17(1):111–122

Chelouah R, Siarry P (2003) Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. Eur J Oper Res 148(2):335–348

Davis L (1991) Bit-climbing, representational bias, and test suite design. In: Belew R, Booker LB (eds) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 18–23

De Jong K, Potter MA, Spears WM (1997) Using problem generators to explore the effects of epistasis. In: Bäck T (ed) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 338–345

Dorigo M, Stützle T (2004) Ant colony optimization. MIT, Cambridge

Dunham B, Fridshal D, Fridshal R, North JH (1963) Design by natural selection. Synthese 15(1):254–259

Fernandes C, Rosa A (2001) A study on non-random mating and varying population size in genetic algorithms using a royal road function. In: Proceedings of the congress on evolutionary computation. IEEE, New York, pp 60–66

Fernandes C, Rosa AC (2008) Self-adjusting the intensity of assortative mating in genetic algorithms. Soft Comput 12(10):955–979

Fournier NG (2007) Modelling the dynamics of stochastic local search on k-sat. J Heuristics 13(6):587–639

Garcia S, Molina D, Lozano M, Herrera F (2008) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. J Heuristics. doi:10.1007/s10732-008-9080-4

Garcia S, Fernández A, Luengo J, Herrera F (2009) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13(10):959–977

García-Martínez C, Lozano M (2008) Local search based on genetic algorithms. In: Siarry P, Michalewicz Z (eds) Advances in metaheuristics for hard optimization. Natural computing. Springer, Heidelberg, pp 199–221

García-Martínez C, Lozano M, Molina D (2006) A local genetic algorithm for binary-coded problems. In: Runarsson TP, Beyer H-G, Burke E, Merelo-Guervós JJ, Whitley LD, Yao X (eds) Proceedings of the international conference on parallel problem solving from nature. LNCS, vol 4193. Springer, Heidelberg, pp 192–201

García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez AM (2008) Global and local real-coded genetic algorithms based on parent-centric crossover operators. Eur J Oper Res 185(3):1088–1113

Glover F, Kochenberger G (eds) (2003) Handbook of metaheuristics. Kluwer, Dordrecht

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley/Longman, Menlo Park/London

Goldberg DE, Korb B, Deb K (1989) Messy genetic algorithms: motivation, analysis, and first results. Complex Syst 3:493–530

Gortazar F, Duarte A, Laguna M, Martí R (2008) Context-independent scatter search for binary problems. Technical report, Colorado LEEDS School of Business, University of Colorado at Boulder

Hansen P, Mladenović N (2002) Variable neighborhood search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Dordrecht, pp 145–184

Harada K, Ikeda K, Kobayashi S (2006) Hybridization of genetic algorithm and local search in multiobjective function optimization: recommendation of GA then LS. In: Cattolico M (ed) Proceedings of the genetic and evolutionary computation conference. ACM, New York, pp 667–674

Harik G (1995) Finding multimodal solutions using restricted tournament selection. In: Eshelman LJ (ed) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 24–31

Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J Optim 10(3):673–696

Herrera F, Lozano M (2000) Gradual distributed real-coded genetic algorithms. IEEE Trans Evol Comput 4(1):43–63

Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6:65–70

Hoos HH, Stützle T (2004) Stochastic local search. Morgan Kaufmann Publishers, San Francisco

Iman RL, Davenport JM (1980) Approximations of the critical region of the Friedman statistic. In: Communications in statistics. pp 571–595

Ishibuchi H, Hitotsuyanagi Y, Tsukamoto N, Nojima Y (2009) Use of biased neighborhood structures in multiobjective memetic algorithms. Soft Comput 13(8–9):795–810

Jones T (1995) Crossover, macromutation, and population-based search. In: Eshelman L (ed) Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 73–80

Karp RM (1972) Reducibility among combinatorial problems. In: Miller R, Thatcher J (eds) Complexity of computer computations. Plenum, NY, pp 85–103

Katayama K, Narihisa H (2001) A variant k-opt local search heuristic for binary quadratic programming. Trans IEICE (A) J84-A(3):430–435

Katayama K, Narihisa H (2005) An evolutionary approach for the maximum diversity problem. In: Recent advances in memetic algorithms. Springer, Heidelberg, pp 31–47

Kauffman SA (1989) Adaptation on rugged fitness landscapes. Lec Sci Complex 1:527–618

Kazarlis SA, Papadakis SE, Theocharis JB, Petridis V (2001) Microgenetic algorithms as generalized hill-climbing operators for GA optimization. IEEE Trans Evol Comput 5(3):204–217

Kong M, Tian P, Kao Y (2008) A new ant colony optimization algorithm for the multidimensional knapsack problem. Comput Oper Res 35(8):2672–2683

Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: Model, taxonomy and design issues. IEEE Trans Evol Comput 9(5):474–488

Laguna M (2003) Scatter search. Kluwer, Boston

Lima CF, Pelikan M, Sastry K, Butz M, Goldberg DE, Lobo FG (2006) Substructural neighborhoods for local search in the bayesian optimization algorithm. In: Proceedings of the international conference on parallel problem solving from nature. LNCS, vol 4193, pp 232–241

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21(2):498–516

Lourenço HR, Martin O, Stützle T (2003) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics, Kluwer, Dordrecht, pp 321–353

Lozano M, García-Martínez C (2010) Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput Oper Res 37:481–497

Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273–302

Mahfoud SW (1992) Crowding and preselection revised. In: Männer R, Manderick B (eds) Parallel problem solving from nature, vol 2. Elsevier Science, London, pp 27–36

Marti R (2003) Multi-start methods. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Dordrech, pp 355–368

Martí R, Moreno-Vega JM, Duarte A (2009) Advanced multi-start methods, 2nd edn. In: Handbook of metaheuristics. Springer, Heidelberg

Merz P (2001) On the performance of memetic algorithms in combinatorial optimization. In: Second workshop on memetic algorithms, genetic and evolutionary computation conference. Morgan Kaufmann, Menlo Park, pp 168–173

Merz P, Katayama K (2004) Memetic algorithms for the unconstrained binary quadratic programming problem. Biosystems 79(1–3):99–118

Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, NY, pp 219–234

Mutoh A, Tanahashi F, Kato S, Itoh H (2006) Efficient real-coded genetic algorithms with flexible-step crossover. Trans Electron Inf Syst 126(5):654–660

Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M (2007) Implementation of effective hybrid GA for large-scale traveling salesman problems. IEEE Trans Syst Man Cybern B 37(1):92–99

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125

O'Reilly UM, Oppacher F (1995) Hybridized crossover-based search techniques for program discovery. In: Proceedings of the world conference on evolutionary computation, vol 2, pp 573–578

Peng G, Ichiro I, Shigeru N (2007) Application of genetic recombination to genetic local search in TSP. Int J Inf Technol 13(1):57–66

Potts JC, Giddens TD, Yadav SB (1994) The development and evaluation of an improved genetic algorithm based on migration and artificial selection. IEEE Trans Syst Man Cybern 24:73–86

Raidl GR (2006) A unified view on hybrid metaheuristics. In: Almeida F, Aguilera MJB Blesa, Blum C, Vega JM Moreno, Pérez M Pérez, Roli A, Sampels M (eds) Hybrid metaheuristics. LNCS, vol 4030. Springer, Heidelberg, pp 1–126

Randall M (2006) Search space reduction as a tool for achieving intensification and diversification in ant colony optimisation. In: Ali M, Dapoigny R (eds) LNCS, vol 4031. Springer, Heidelberg, pp 254–262

Ray SS, Bandyopadhyay S, Pal SK (2007) Genetic operators for combinatorial optimization in TSP and microarray gene ordering. App Intell 26(3):183–195

Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Dordrecht, pp 219–249

Sastry K, Goldberg DE (2004) Designing competent mutation operators via probabilistic model building of neighborhoods. In: Deb K, Poli R, Banzhaf W, Beyer H-G, Burk EK, Darwen PJ, Dasgupta D, Floreano D, Foster JA, Harman M, Holland O, Lanzi PL, Spector L, Tettamanzi A, Thierens D, Tyrrel AM (eds) Proceedings of the conference on genetic and evolutionary computation. LNCS, vol 3103, pp 114–125

Siarry P, Michalewicz Z (eds) (2008) Advances in metaheuristics for hard optimization. Natural Computing, Springer

Smith K, Hoos HH, Stützle T (2003) Iterated robust tabu search for MAX-SAT. In: Carbonell JG, Siekmann J (eds) Proceedings of the Canadian society for computational studies of intelligence conference. LNCS, vol 2671. Springer, Heidelberg, pp 129–144

Soak S-M, Lee S-W, Mahalik NP, Ahn B-H (2006) A new memetic algorithm using particle swarm optimization and genetic algorithm. In: Knowledge-based intelligent information and engineering systems. LNCS, vol 4251. Springer, Berlin, pp 122–129

Spears WM (2000) Evolutionary algorithms: the role of mutation and recombination. Springer, Heidelberg

Spears WM, De Jong KA (1991) On the virtues of parameterized uniform crossover. In: Belew R, Booker LB (eds) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 230–236

Sywerda G (1989) Uniform crossover in genetic algorithms. In: Schaffer JD (ed) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 2–9

Talbi EG (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8(5):541–564

Thierens D (2004) Population-based iterated local search: restricting neighborhood search by crossover. In: Deb K, Poli R, Banzhaf W, Beyer H-G, Burk EK, Darwen PJ, Dasgupta D, Floreano D, Foster JA, Harman M, Holland O, Lanzi PL, Spector L, Tettamanzi A, Thierens D, Tyrrel AM (eds) Proceedings of the genetic and evolutionary computation conference. LNCS, vol 3103. Springer, Heidelberg, pp 234–245

Tsai H-K, Yang J-M, Tsai Y-F, Kao C-Y (2004) An evolutionary algorithm for large traveling salesman problems. IEEE Trans Syst Man Cybern 34(4):1718–1729

Tsutsui S, Ghosh A, Corne D, Fujimoto Y (1997) A real coded genetic algorithm with an explorer and an exploiter population. In: Bäck T (ed) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 238–245

Ventura S, Romero C, Zafra A, Delgado JA, Hervás-Martínez C (2008) JCLEC: A java framework for evolutionary computation. Soft Comput 12(4):381–392

Wang H, Wang D, Yang S (2009) A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. Soft Comput 13(8–9):763–780

Whitley D (1989) The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Schaffer JD (ed) Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, Menlo Park, pp 116–121

Zar JH (1999) Biostatistical analysis. Prentice Hall, Englewood Cliffs