2009 Ninth International Conference on Intelligent Systems Design and Applications

Continuous Variable Neighbourhood Search Algorithm Based on Evolutionary Metaheuristic Components: A Scalability Test

Carlos García-Martínez
Dept. of Computing and Numerical Analysis
Univ. of Córdoba, 14071 Córdoba, Spain
cgarcia@uco.es

Manuel Lozano
Dept. of Computer Science and Artificial Intelligence
CITIC-UGR (Research Center on Information and Communications Technology)
Univ. of Granada, 18071 Granada, Spain
lozano@decsai.ugr.es

Abstract

Variable Neighbourhood Search *is a metaheuristic combining three components: generation, improvement, and shaking components. In this paper, we describe a continuous Variable Neighbourhood Search algorithm based on three* specialised Evolutionary Algorithms*, which play the role of each aforementioned component: 1) an EA specialised in generating a good starting point as generation component, 2) an EA specialised in exploiting local information as improvement component, 3) and another EA specialised in providing local diversity as shaking component. We adopt the experimental framework proposed for the Special Session on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test, for the ISDA'09 conference, to test the ability of the model of being scalable for high-dimensional problems.*

1. Introduction

Nowadays, high-dimensional optimisation problems arise as a very interesting field of research, because they appear in many important new real-world problems (biocomputing, data mining, etc.). Unfortunately, the performance of most available optimisation algorithms deteriorates rapidly as the dimensionality of the search space increases.

Metaheuristics (MHs) [11] are a family of search and optimisation algorithms based on extending basic heuristic methods by including them into an iterative framework augmenting their exploration capabilities. MHs coordinate *subordinate components* (such as probability distributions, tabu lists or genetic operators among others) with the aim of performing an effective and efficient process in searching for the global optimum of a problem. Over the last years, a large number of search algorithms were reported that do not purely follow the concepts of one single classical MH, but they attempt to obtain the best from a set of MHs that perform together and complement each other to produce a profitable synergy from their combination. These approaches are commonly referred to as *hybrid MHs* [22].

Evolutionary Algorithms (EAs) [3] are stochastic search methods that mimic the metaphor of natural biological evolution. EAs rely on the concept of a population of individuals, which undergo probabilistic operators to evolve toward increasingly better fitness values of the individuals.

A novel method to build hybrid MHs, recently introduced in [19], concerns the incorporation of *specialised EAs* into classical MHs, replacing determinate components, but preserving the essence of the original MH as much as possible. The idea is to build customised EAs playing the same role as particular MH components, but more effectively, i.e., *evolutionary MH components*. In this way, a classical MH is transformed into an integrative hybrid MH (because one of its components is another MH). In the literature, we find some proposals that follow this idea: In [1] and [6], two EAs are applied to perform local search processes within a multi-start MH; and in [18, 19], an evolutionary ILS-perturbation technique is presented (ILS is for Iterated Local Search).

Variable Neighbourhood Search (VNS) [21] is a MH that exploits systematically the idea of neighbourhood change (from a given set of neighbourhood structures N_k, $k = 1, \ldots, k_{max}$), both in the descent to local minima and in the escape from the valleys which contain them. It mainly consists of the following three components, which work on a single candidate solution, the *current solution* ($\boldsymbol{s^c}$) (see Fig. 1):

1. *Generation component:* Firstly, a method is executed to generate $\boldsymbol{s^c}$ within the search space.

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.68

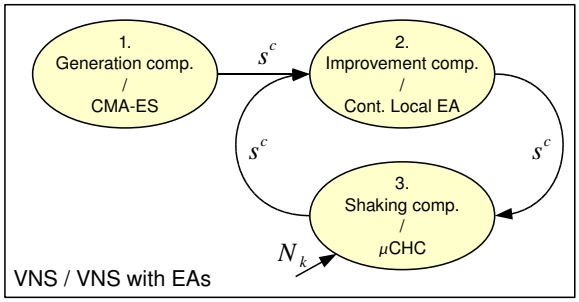

Figure 1. VNS model / VNS based on EAs

2. *Improvement component:* Secondly, $\boldsymbol{s^c}$ is refined, usually by a local search method.

3. *Shaking component:* Then, shaking is performed to escape from the valley where $\boldsymbol{s^c}$ lies. It selects a random solution in the kth neighbourhood of $\boldsymbol{s^c}$, which becomes a new starting solution for improvement component. At the beginning of the run, and every time last improvement process improved the best found solution, k is set to one; otherwise, k is set to $k + 1$.

In this work, we study a novel continuous VNS model, presented in [7, 8], that is based on three specialised EAs playing the role of each VNS component. Experiments are carried out to test the ability of the algorithm of being scalable for high-dimensional problems. With this aim, we adopt the experimental framework proposed for the Special Session on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test, for the ISDA'09 conference.

In Sect. 2, we describe the general scheme of the VNS model based on evolutionary MH components. In Sect. 3, we introduce the generation component. In Sect. 4, we detail the improvement component. In Sect. 5, shaking component is described. In Sect. 6, we present the empirical study. Conclusions are presented in Sect. 7.

2. VNS based on Specialised EAs

In this study, we focus on a continuous VNS model based on three specialised EAs, which play the role of each VNS component (see Fig. 1):

1. *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [12, 13] generates initial $\boldsymbol{s^c}$ (Sect. 3).

2. *Continuous Local EA* improves $\boldsymbol{s^c}$ (Sect. 4).

3. *Micro Cross-generational elitist sel., Heterogeneous recombination, and Cataclysmic mutation* (μCHC) performs shaking to scape from the valley where $\boldsymbol{s^c}$ lies, according to the kth neighbourhood (Sect. 5).

Adaptation of parameter k is performed the same way original VNS algorithm does.

Neighbourhoods structures are defined by using metric ρ:

$$N_k(\boldsymbol{s^c}) = \{\boldsymbol{s} \mid r_{k-1} \leq \rho(\boldsymbol{s^c}, \boldsymbol{s}) \leq r_k\} \quad (1)$$

where r_k is the radius of $N_k(\boldsymbol{s^c})$ monotonically nondecreasing with k. Hereafter, we will always consider the following distance metric because of computational reasons:

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i| \quad (2)$$

Bound constraints are checked every time a new solution is sampled. Every value that lies outside the ranges of its corresponding variable is recomputed to its symmetric value with regards to the violated bound. This process is repeated until no bound constraint is violated.

Stop condition is reached when k surpasses k_{max} (in our case, k_{max} is set to 20). At this point, the algorithm performs a *restart*. In this way, exploiting larger number of function evaluations may increase the chance to achieve better function values. In our case, a restart means that VNS is run once again from the CMA-ES execution, generating a new $\boldsymbol{s^c}$.

3. CMA-ES as Generator Component

The proposal applies CMA-ES [12, 13] to obtain a good point in the design space from which the algorithm carries out its search process. We have applied the C code, available at `http://www.lri.fr/~hansen/cmaes_inmatlab.html` with the suggested parameter values from a random solution.

4. Continuous Local EA as Improvement Component

Continuous Local EA plays the role of improvement component for the VNS model. It is an EA based on the principles of Binary Local Genetic Algorithm [10], which obtained promising results in efficacy and efficiency against classical improvement methods.

Continuous Local EA is a steady-state *real-coded genetic algorithm* [14, 15] that inserts one single new member into the population (Pop) in each iteration. It uses a replacement method 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 favours the acquisition of information about

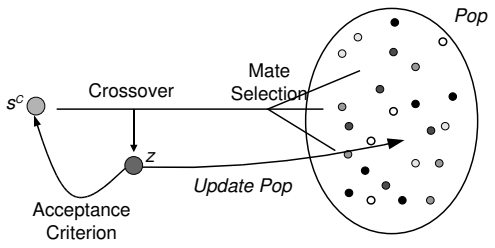

Figure 2. Continuous Local EA

the search space. Then, the algorithm exploits local information around $\boldsymbol{s^c}$, provided by the individuals in Pop close to it, to orientate the improvement process.

Let's suppose that Continuous Local EA is to improve the given $\boldsymbol{s^c}$. Then, the following steps are carried out (see Fig. 2):

- *Mate selection*: A solution from Pop is selected as $\boldsymbol{s^{mate}}$ by means of *nearest improving solution selection method* (Sect. 4.1).
- *Crossover*: $\boldsymbol{s^{mate}}$ and $\boldsymbol{s^c}$ are crossed over by means of *parent-centric BLX-*α, creating offspring $\boldsymbol{z}$ (Sec. 4.2).
- *Acceptance and replacement*: If $\boldsymbol{z}$ is accepted (Sect. 4.3), it replaces $\boldsymbol{s^c}$ and is inserted into the population forcing the individual with lowest information contribution to perish (Sect. 4.4).

All these steps are repeated until a maximum number of iterations without improving the best visited $\boldsymbol{s^c}$ is reached (we stop the run after 100 iterations without improving best $\boldsymbol{s^c}$). At last, the best found $\boldsymbol{s^c}$ is returned.

An important aspect when using Continuous Local EA in the proposed VNS model is that Pop undergoes initialisation only once, at the beginning of the run, and not at every invocation to improve $\boldsymbol{s^c}$. In this way, the algorithm may gather up valuable information about the search space and its optima, and employ accumulated search experience from previous refinements to enhance future ones.

4.1. Nearest Improving Solution Selection

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 et al. [5] implement these ideas to design two mating selection mechanisms.

We introduce a new assortative mating mechanism to be used in Continuous Local EA, *nearest improving solution selection*. Its idea is to select the individual in Pop that provides valuable information of the search space (location and fitness value) for the task of locally improving $\boldsymbol{s^c}$. On the one hand, this mechanism dismisses individuals worse than $\boldsymbol{s^c}$, because they reduce chances for improving $\boldsymbol{s^c}$. On the other hand, nearest solution is selected because it provides more information about the search space region where $\boldsymbol{s^c}$ is located. We also include a distance threshold to avoid choosing a solution too similar to $\boldsymbol{s^c}$.

Therefore, first parent is always $\boldsymbol{s^c}$, then, second parent is the individual in Pop most similar to $\boldsymbol{s^c}$ (at phenotypic level) fulfilling two conditions: 1) it has a better fitness value than $\boldsymbol{s^c}$ and 2) similarity between both solutions is above a given threshold. Mating threshold is initially set to $1e-2$. Then, after every VNS restart, it is set to half its value to improve precision.

An important remark is that this selection mechanism may return no mate for $\boldsymbol{s^c}$. It occurs when the mechanism has dismissed all solutions in Pop because either they are worse than $\boldsymbol{s^c}$, or their distances to $\boldsymbol{s^c}$ are under mating threshold. Crossover is not performed in this case. Offspring is generated by perturbing $\boldsymbol{s^c}$ instead (Sect. 4.2).

4.2. PBX-α Crossover Operator

PBX-α [20] is an instance of *parent-centric crossover operators*, which have arisen as a meaningful and efficient way of solving real-parameter optimisation problems [2, 9].

Given $\boldsymbol{s^c}= (s_1^c \cdots s_n^c)$ and $\boldsymbol{s^{mate}}= (s_1^{mate} \cdots s_n^{mate})$, PBX-$\alpha$ generates offspring $\boldsymbol{z}= (z_1 \cdots z_n)$, where z_i is a random (uniform) number from interval $[s_i^c - I \cdot \alpha, s_i^c + I \cdot \alpha]$, with $I = |s_i^c - s_i^{mate}|$. Parameter α is initially set to 0.5. Then, after every VNS restart, it is set to $0.5/log_e(number\ of\ restarts + 1)$ to improve precision.

As mentioned before, crossover operation occurs when mate selection returns a solution from Pop. Otherwise, offspring is generated by adding a normal random vector to $\boldsymbol{s^c}$, whose standard deviation is the product of the current values of aforementioned parameters α and mating threshold.

Bounds constraints are checked as in the general VNS framework (Sect. 2), either when PBX-α is applied or when a vector of random values is added to $\boldsymbol{s^c}$.

4.3. Acceptance Criterion

Once offspring $\boldsymbol{z}$ has been generated either by PBX-α or perturbation, Continuous Local EA decides which solution, between $\boldsymbol{z}$ and $\boldsymbol{s^c}$, becomes the new $\boldsymbol{s^c}$. It follows a similar idea to the one behind *Simulated Annealing* [16] to overcome rugged landscapes and local optima. In particular, it always accepts $\boldsymbol{z}$ if it is better than a specific computed bound. At the beginning, that bound is set equal to the fitness value of $\boldsymbol{s^c}$, only allowing improving offspring.

Then, every time a new best s^c is found, the bound is computed as the average of the fitness of the new best s^c and the old bound. This simple mechanism provides some selection pressure on the search process that also lets the algorithm to accept uphill moves, and overcome small local optima.

4.4. Lowest Information Contribution Replacement

One of the objectives of Continuous Local EA is to gather up valuable information about the search space and its optima, to enhance future refinements. To carry out this objective, Pop should maintain a set of good solutions properly spread over the search space. The problem of maintaining such a set of solutions recalls the problem of attaining a good set of non-dominated solutions in multi-objective problems. In [17], Kukkonen and Deb propose a method for pruning of non-dominated solutions in many-objective problems. The basic idea is to eliminate the most crowded members of a non-dominated set one by one, and update the crowding information of the remaining members after each removal.

Continuous Local EA applies a similar method for updating Pop, when solving mono-objective problems. The mechanism firstly inserts the new offspring in Pop, then, it removes the most crowded solution, keeping population size constant (we use $|Pop| = 100$). To determine the solution to be removed, everyone in Pop holds a pointer to its nearest solution as well as the distance between them, at phenotypic level. Since, according to this definition, there are always, at least, two solutions holding the minimum distance, the worst of them is the one to be removed. Pointers and distances are updated every time Pop changes, i.e., when inserting the new solution and when removing the most crowded one. Notice that the best found solution is never removed.

5. μCHC as Shaking Component

The VNS model applies μCHC as the shaking component. μCHC was firstly presented as an *evolutionary ILS-perturbation technique* for binary-coded problems [18, 19]. The role of μCHC is to receive s^c, provide local diversity, and generate a solution that is then considered as starting point for the next improvement process (see Fig. 3). The reason for choosing CHC, as the base model, is that it suitably combines powerful diversification mechanisms with an elitist selection strategy. The filtering of high diversity by means of high selective pressure favours the creation of *useful diversity*: many dissimilar solutions are produced during the run and only the best ones are conserved in the population, allowing diverse and promising solutions to be maintained. From our point of view, this behaviour is desirable for an EA assuming the work of a shaking operator.

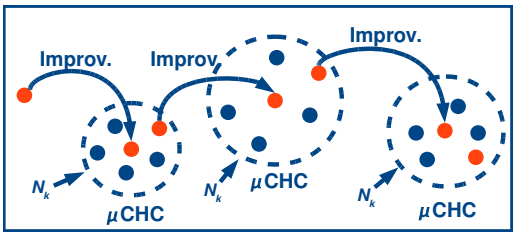

Figure 3. μCHC as shaking module

5.1. μCHC Model

We have conceived μCHC to be an effective *explorer* in the neighbourhood of s^c to perform shaking process, because it provides local diversity. At the beginning, s^c is used to create its initial population (Sect. 5.2). Then, it is performed throughout a predetermined number of fitness function evaluations. The best reached individual is then considered as starting point for the next improvement process (see Fig. 3). Every time a new solution is sampled, bound constraints are checked as in the general VNS framework (Sect. 2).

The main components of the algorithm are:

- *Population size:* μCHC manages a population with few individuals ($|Pop| = 5$), and thus, it may be seen as micro EA. In standard VNS models, the number of evaluations required by the shaking mechanism is very low as compared with the one for the improvement method. With the aim of preserving, as far as possible, the essence of VNS, we have considered an EA with a low sized population; for being able to work adequately under the requirement of spending few evaluations.

- *Number of evaluations:* In particular, the number of evaluations assigned to μCHC for a particular invocation will be a fixed proportion (p_{evals}), of the number of evaluations consumed by the previously performed improvement method (Continuous Local EA). It is worth noting that p_{evals} should be set to a low value. We will use p_{evals} equals to 0.5.

- *Elitist selection:* Current population is merged with offspring population obtained from it and the best $|Pop|$ individuals are selected for the new population.

- *Incest prevention mechanism*: Before mating, Hamming distance between the corresponding Gray-coding strings of paired individuals (with 20 bits per variable) is calculated. Only paired individuals whose distance exceed a difference threshold d are allowed to undergo

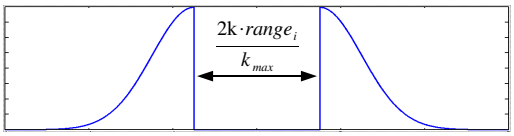

Figure 4. Prob. distribution for initialisation

crossover operation. Aforementioned threshold is initialised to $L/4$ (with L being the number of bits coding a potential solution). If no offspring is obtained in one generation, difference threshold is decremented by one.

- *BLX-α crossover operator*: Paired individuals allowed to produce offspring undergo BLX-α crossover operator [4], producing one offspring. Given $\boldsymbol{y^1}$ and $\boldsymbol{y^2}$, BLX-α generates offspring $\boldsymbol{z}$, where z_i is a random (uniform) number from $[y^{min} - I \cdot \alpha, y^{max} + I \cdot \alpha]$ interval, with $y^{min} = min(y_i^1, y_i^2)$, $y^{max} = max(y_i^1, y_i^2)$, and $I = |y_i^1 - y_i^2|$. This parameter α is set to 0.5.

- *Mating with $\boldsymbol{s^c}$:* μCHC incorporates a strategy of recombining $\boldsymbol{s^c}$ with another solution. In addition to the typical recombination phase of CHC, our algorithm mates $\boldsymbol{s^c}$ with a member of Pop (selected at random) and, if they are finally crossed over (attending on the incest prevention mechanism), the resulting offspring will be introduced into the offspring population.

- *Cataclysmic mutation:* μCHC, as CHC, uses no mutation in the classical sense of the concept, but instead, it goes through a process of cataclysmic mutation when the population has converged. The difference threshold is considered to measure the stagnation of the search, which happens when it has dropped to one. Then, the population is reinitialised with individuals generated by perturbing $\boldsymbol{s^c}$ attending to the kth neighbourhood structure (Sect. 5.2).

5.2. Initialisation and Cataclysmic Mutation

Every individual in the initial μCHC population is generated by perturbing $\boldsymbol{s^c}$ according to a relaxed idea of the given neighbourhood structure N_k (see equation 1). In particular, each individual is generated by adding a vector of random values from the following equation (see Fig. 4):

$$N(0,1) \cdot range_i / k_{max} \pm k \cdot range_i / k_{max} \qquad (3)$$

where $N(0,1)$ is a normal random value with mean 0 and variance 1, $range_i$ is the range of ith variable domain, k_{max} is the maximum number of neighbourhood structures VNS manages, and k is the current neighbourhood index; $\pm$ is chosen according to the sign of the random value.

Table 1. Test problems

	Name	Interval	f^*
f_1	Shifted Sphere	[-100, 100]	-450
f_2	Schwefel	[-100, 100]	-450
f_3	Shifted Rosenbrock	[-100, 100]	390
f_4	Shifted Rastrigin	[-5, 5]	-330
f_5	Shifted Griewank	[-600, 600]	-180
f_6	Shifted Ackley	[-32, 32]	-140
f_7	Schwefel 2.22	[-10, 10]	0
f_8	Schwefel 1.2	[-65536, 65536]	0
f_9	Extended F10	[-100, 100]	0
f_{10}	Bohachevsky	[-15, 15]	0
f_{11}	Schaffer	[-100, 100]	0

Table 2. Parameter setting

Parameter	Value	Parameter	Value
k_{max}	20	CMA-ES parameters	Suggested in code
|CLEA's Pop|	100	CLEA stop cond.	100 its without improv.
|μCHC's Pop|	5	CLEA mating th.	1e-2 initially
BLX-α	0.5	PBX-α	0.5 initially
p_{evals}	0.5	μCHC bits/var	20

Cataclysmic mutation fills the population with individuals created by the same way as initial population is built, but preserving the best performing individual found in the previous generation. After applying cataclysmic mutation, the difference threshold is set to: $\sigma \cdot (1 - \sigma) \cdot L$, with L being the number of bits coding a potential solution of the problem (20 bits per variable) and σ is the average of the minimal radius of neighbourhoods N_k and N_{k+1} ($(2k+1)/(2k_{max})$).

6. Results

The continuous VNS model with evolutionary MH components has been tested on the experimental framework proposed for the Special Session on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test, for the ISDA'09 conference. It consist of 11 scalable function optimisation problems. Table 1 shows their names, bounds, and optimum values. Functions f_1 to f_6 are defined in [23], and functions f_7 to f_{11}, in [9]. Four dimensions ($D = \{50, 100, 200, 500\}$) are studied. The maximum number of fitness evaluations is $5,000 \cdot D$. Table 2 summarises the parameter setting (parameter values taken from [7]), where CLEA is for Continuous Local EA. Table 3 shows the *error average*, defined as the difference between the best attained fitness and the optimum value, and its standard deviation in brackets over 25 runs for each test function and dimension.

Table 3. Results

	50	100	200	500
f_1	5,7e-14(3e-29)	1,3e-13(3e-14)	2,6e-13(3e-14)	6,0e-13(3e-14)
f_2	2,7e-11(6e-12)	5,8e-9(1e-8)	1,1e-9(6e-10)	3,6e-4(1,1e-4)
f_3	1,5e+0(2,8e+0)	1,8e+1(8,4e+0)	1,0e+2(1,5e+1)	6,1e+4(3,0e+5)
f_4	1,3e+1(4,5e+0)	4,4e+1(1,3e+1)	1,9e+2(1,1e+2)	2,5e+3(4,3e+2)
f_5	3,3e-14(1e-14)	7,7e-14(2e-14)	1,6e-13(1e-14)	3,5e-13(2e-14)
f_6	1,8e+1(7,5e+0)	2,1e+1(5,8e-1)	2,1e+1(1,9e-1)	2,2e+1(4,3e-2)
f_7	1,5e-9(4,0e-9)	7,7e-5(2,4e-4)	4,1e-2(7,2e-2)	5,0e-2(5,1e-2)
f_8	1,2e-14(1e-14)	5,8e-14(1e-14)	2,6e-13(4e-14)	4,9e-4(2,3e-4)
f_9	4,8e+0(9,0e+0)	2,2e+1(1,5e+1)	1,5e+2(8,0e+1)	2,5e+3(6,6e+2)
f_{10}	1,7e-4(1,5e-4)	4,5e-4(2,7e-4)	1,3e-3(5,7e-4)	4,7e-3(6,5e-4)
f_{11}	3,6e+0(6,5e+0)	2,2e+1(1,7e+1)	1,7e+2(7,2e+1)	2,4e+3(6,1e+2)

7. Conclusions

We have presented the continuous VNS model based on three evolutionary MH components: 1) CMA-ES as an EA specialised in generating a good starting point, as generation component, 2) Continuous Local EA, specialised in exploiting local information, as improvement component, and 3) μCHC, which provides local diversity, as shaking component. We have tested the continuous VNS model with evolutionary MH components on the experimental framework proposed for the Special Session on Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test, for the ISDA'09 conference, to study the ability of the algorithm of being scalable for high-dimensional problems.

8 Acknowledgments

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

References

[1] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proc. of the IEEE Int. Conf. Evolutionary Computation*, volume 2, pages 1777–1784. IEEE Press, 2005.

[2] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evol. Comput.*, 10(4):371–395, 2002.

[3] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[4] L. Eshelman and J. Schaffer. Real-coded genetic algorithms and interval-schemata. In L. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. 1993.

[5] C. Fernandes and A. Rosa. A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Proc. of the Congress on Evolutionary Computation*, pages 60–66. IEEE Press, 2001.

[6] C. García-Martínez and M. Lozano. Local search based on genetic algorithms. In P. Siarry and Z. Michalewicz, editors, *Advances in Metaheuristics for Hard Optimization*, Natural Computing, pages 199–221. Springer, 2008.

[7] C. García-Martínez and M. Lozano. A continuous variable neighbourhood search based on specialised EAs: Application to the noiseless BBO-benchmark 2009. In *Genetic Evolutionary Computation Conf.*, pages 2287–2294, 2009.

[8] C. García-Martínez and M. Lozano. A continuous variable neighbourhood search based on specialised EAs: Application to the noisy BBO-benchmark 2009 testbed. In *Genetic Evolutionary Computation Conf.*, pages 2367–2374, 2009.

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

[10] C. García-Martínez, M. Lozano, and D. Molina. A local genetic algorithm for binary-coded problems. In *Proc. of the Int. Conf. on Parallel Problem Solving from Nature*, volume 4193 of *LNCS*, pages 192–201. Springer, 2006.

[11] F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[12] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.*, 11(1):1–18, 2003.

[13] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[14] F. Herrera and M. Lozano. Two-loop real-coded genetic algorithms with adaptive control of mutation step sizes. *App. Intell.*, 13(3):187–204, 2000.

[15] F. Herrera, M. Lozano, and J. Verdegay. Tackling real-coded genetic algorithms: operators and tools for behavioral analysis. *Artif. Intell. Rev.*, 12(4):265–319, 1998.

[16] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. *Sci.*, 220(4598):671–680, 1983.

[17] S. Kukkonen and K. Deb. A fast and effective method for pruning of non-dominated solutions in many-objective problems. In *Proc. of the Int. Conf. on Parallel Problem Solving from Nature, LNCS*, volume 4193, pages 553–562, 2006.

[18] M. Lozano and C. García-Martínez. An evolutionary ILS-perturbation technique. In *Proc. of the Int. Workshop on Hybrid Metaheuristics, LNCS*, volume 5296, pages 1–15, 2008.

[19] M. Lozano and C. García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Comput. Oper. Res.*, In press, 2009.

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

[21] N. Mladenovic and P. Hansen. Variable neighborhood search. *Comput. Oper. Res.*, 24:1097–1100, 1997.

[22] E. Talbi. A taxonomy of hybrid metaheuristics. *J. Heuristics*, 8(5):541–564, 2002.

[23] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation And Applications Laboratory, USTC, China, 2007.