Hybrid Self-Adaptive Evolution Strategies Guided by Neighborhood Structures for Combinatorial Optimization Problems

V. N. Coelho vncoelho@gmail.com
Undergraduate Program of Control and Automation Engineering, Universidade Federal de Ouro Preto, Ouro Preto, MG, 35400-000, Brazil
Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 31270-901, Brazil

I. M. Coelho igor.machado@ime.uerj.br
Department of Computer Science, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 20550-900, Brazil

M. J. F. Souza marcone@iceb.ufop.br
Department of Computer Science, Universidade Federal de Ouro Preto, Ouro Preto, MG, 35400-000, Brazil

T. A. Oliveira thaysoliveira7@gmail.com
Business Graduate Program, Department of Business and Economics, Universidade Federal de Lavras, Lavras, Brazil

L. P. Cota lucianocota@ufmg.br
Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 31270-901, Brazil

M. N. Haddad matheushaddad@ic.uff.br
Computer Science Graduate Program, Universidade Federal Fluminense, Niterói, Brazil

N. Mladenovic nenadmladenovic12@gmail.com
LAMIH, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France
Mathematical Institute, Serbian Academy of Science and Arts, Serbia

R. C. P. Silva rodrigo.silva@mail.mcgill.ca
Department of Electrical and Computer Engineering, McGill University, Montreal, QC, H3A 0E9, Canada

F. G. Guimarães fredericoguimaraes@ufmg.br
Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 31270-010, Brazil

doi:10.1162/EVCO_a_00187

Abstract

This article presents an Evolution Strategy (ES)–based algorithm, designed to self-adapt its mutation operators, guiding the search into the solution space using a Self-Adaptive Reduced Variable Neighborhood Search procedure. In view of the specific local search operators for each individual, the proposed population-based approach also fits into the

Manuscript received: September 2, 2015; revised: March 31, 2016 and April 11, 2016; accepted: May 4, 2016.

context of the Memetic Algorithms. The proposed variant uses the Greedy Randomized Adaptive Search Procedure with different greedy parameters for generating its initial population, providing an interesting exploration–exploitation balance. To validate the proposal, this framework is applied to solve three different $\mathcal{NP}$-Hard combinatorial optimization problems: an Open-Pit-Mining Operational Planning Problem with dynamic allocation of trucks, an Unrelated Parallel Machine Scheduling Problem with Setup Times, and the calibration of a hybrid fuzzy model for Short-Term Load Forecasting. Computational results point out the convergence of the proposed model and highlight its ability in combining the application of move operations from distinct neighborhood structures along the optimization. The results gathered and reported in this article represent a collective evidence of the performance of the method in challenging combinatorial optimization problems from different application domains. The proposed evolution strategy demonstrates an ability of adapting the strength of the mutation disturbance during the generations of its evolution process. The effectiveness of the proposal motivates the application of this novel evolutionary framework for solving other combinatorial optimization problems.

Keywords

Evolution strategies, neighborhood structures, reduced variable neighborhood search, memetic algorithms, open-pit mining operational planning, unrelated parallel machine scheduling, short-term load forecasting.

1 Introduction

In this article, the class of Evolutionary Algorithms (EA) known as Evolution Strategies (ES) (Beyer and Schwefel, 2002) is investigated and a new ES variant is proposed, being able to solve challenging combinatorial optimization problems. We combine the diversity of the population-based algorithms with the power of Reduced Variable Neighborhood Search (RVNS) (Hansen and Mladenović, 2001), a well-known trajectory search algorithm. Moreover, we supply RVNS with adaptive rules, producing a new variant that we call Adaptive RVNS (ARVNS). EA are search and optimization methods inspired by well-known evolutionary principles, such as random mutations and selective pressure for evolution and adaptation of its population. On the other hand, VNS and other trajectory-based approaches such as Tabu Search (TS) (Glover, 1996) and Iterated Local Search (ILS) (Lourenço et al., 2003) take advantage of the flexibility in designing and exploring different Neighborhood Structures (NS) of the problem, using the simple fact that the local minimum with respect to one neighborhood structure is not necessarily so with respect to another.

Biological evolution in nature is an inspiration for the ES, which is mainly guided by operators: mutation and selection. This class of methods has often been applied for solving continuous optimization problems (Kashan et al., 2015; Chaquet and Carmona, 2012; Andersen and Santos, 2012; Aler et al., 2012; Costa and Oliveira, 2001). While EA have already been applied to solve several combinatorial optimization problems (Prado et al., 2014; Qaurooni and Akbarzadeh-T, 2013; Freitas and Guimarães, 2011), only a few articles in the literature address combinatorial optimization problems using ES (Cai and Thierauf, 1996; Rajasekaran, 2006; Kashan et al., 2015). Our current article focuses on a mechanism for guiding the search in the solutions space, mainly based on a weighting system for applying move operations from k_{max} distinct neighborhoods. The motivation to develop an ES for combinatorial optimization, combined with the use of different NS, comes from the successful applications in both fields, numerical and combinatorial optimization.

Adaptive local search techniques have been exploited by researchers (Dong et al., 2015; Li et al., 2015; Schneider et al., 2014; Hosny and Mumford, 2010). This family of

methods has the ability of exploring attraction basins with iterative moves, combining it with smart strategies that, in general, check the success of previous steps done by the methods. In this context, an ILS with self-adaptive shaking procedure was applied for tackling a flow shop problem (Dong et al., 2015) and multi-depot Vehicle Routing Problem (VRP) (Li et al., 2015). Following the same idea, Schneider et al. (2014) proposed an adaptive mechanism for guiding the shaking step of a VNS applied on a VRP. Their approach selected and favored route and vertex according to their success within the search.

By Reduced VNS method a random point from the k-th neighborhood $N_k(s)$ ($k = 1, \ldots, k_{max}$) of the current incumbent solution x is taken, and no descent from there is made. RVNS has been shown to be useful in solving large instances, for which local search is costly (Xiao et al., 2011; Hansen et al., 2009).

The adaptive variant of the RVNS designed in this article, the ARVNS, explores specific parts of each N_k, playing with probabilities evolved through the ES evolutionary process. Our proposal implicitly considers the problem-specific characteristics and the success of a given N_k within the search. There is no need of mechanisms for analyzing previous success of the N_k, since it is inherited through the genes, ES mutation operators, from the parents to the offspring. The latter are generated after mutations over the ARVNS application probabilities and are expected to survive if interesting changes have occurred.

On the other hand, the proposed self-adaptive evolution strategy also fits the context of Memetic Algorithms (MA) (Moscato and Cotta, 2003), comprising a population-based method in which individuals possess specific, unique, or special searching operators. By incorporating problem domain knowledge, such as heuristics, local search techniques, individual learning strategies, or other stochastic search operators, MA evolve their population through the generations. These techniques have shown a great potential for solving $\mathcal{NP}$-Hard combinatorial and continuous optimization problems (Guimaraes et al., 2007; Wanner et al., 2008; de Freitas et al., 2014).

We introduce an evolution strategy–based algorithm, abbreviated as GES, which generates its initial population through a diversified and greedy procedure. Thus, we suggest the use of the Greedy Randomized Adaptive Search Procedures (GRASP) (Resende and Ribeiro, 2010). GRASP construction phase is responsible for filling the initial population with individuals generated with different random greedy parameters. Other versions with simpler solution generation procedures could also be used, such as generating solutions at random or initializing a homogeneous initial population. We kept the ES mutation operator vectors as the main search operator in the solution space, however, being guided by new data structures incorporated with each individual representation. The proposed adaptive operators regulate the rate of moves application of different NS, in a special case of an RVNS search. An optional intensification phase is also suggested for some problems, in order to accelerate the convergence of the algorithm.

Problems already tackled by the authors of this article were used as case studies. We took profit from the optimization framework OptFrame (see Coelho, Munhoz et al., 2011), a computational framework for the development of efficient metaheuristic algorithms for combinatorial optimization problems. Three different (and very challenging) combinatorial optimization problems are considered in this work as case studies:

1. An Open-Pit-Mining Operational Planning (OPMOP) problem (Souza et al., 2010);
2. An Unrelated Parallel Machine Scheduling Problem with Setup Times (UPMSP-ST) (Al-Salem, 2004), implemented in a Java platform with OptFrame ideas;

3. Calibration of a hybrid fuzzy model calibration for the Short-Term Load Forecasting Problem (STLFP) (Coelho et al., 2014a).

Since we are dealing with many $\mathcal{NP}$-Hard problems, exact solution methods have restricted applicability. This fact motivates us to search for solutions by means of metaheuristic procedures.

The abstraction of the concepts involving the proposed discrete ES and its application over the aforementioned problems was not found in the literature and shows up as a novel evolutionary framework for combinatorial optimization problems. The major contributions of the current work are to:

- Use and adapt a population-based algorithm for combinatorial optimization problems, combining it with trajectory search techniques.
  - —Walk through the search space by using discrete moves, following a trajectory provided by random moves in a Reduced Variable Neighborhood Search;
  - —Combine neighborhood structures in order to guide the search for new solutions using self-adaptive mutation operators;
- Introduce a flexible self-adaptive search framework implemented in the core of the open-source optimization framework OptFrame;
- Apply the proposed methodology for solving different $\mathcal{NP}$-Hard combinatorial optimization problems, including two large-scale real case problems (OPMOP and HFVRPMT).

The remainder of this article is organized as follows. Section 2 details the proposed self-adaptive ES. Section 3 describes an application for the OPMOP. Analogously, Sections 4 and 5 describe mutation operators' behavior of the proposed algorithm over the STLFP and UPMSP-ST, respectively. Section 6 draws the final considerations and future works.

2 Self-Adaptive Evolution Strategy

Evolution Strategies (ES) were developed in the 1960s and 1970s by P. Bienert, I. Rechenberg, and H.-P Schwefell at the Technical University of Berlin. The initial version operated with single individuals, subjected to mutation and selection among their descendants. Beyer and Schwefel (2002) provide a detailed description about ES in their comprehensive introduction.

As mentioned by Meyer-Nieberg and Beyer (2007), self-adaptation can be seen as state-of-the-art methods for adjusting the setting of control parameters. A study about self-adaptation in EA applied for Combinatorial Optimization can be found in Smith (2008).

ES uses natural problem-dependent representations according to each problem that is being tackled. One advantage is its searching ability over the evolution process that is guided, primarily, by mutation and selection. Here, we take advantage of the basic principle, introducing the ARVNS, an RVNS guided by probabilities. A wide range of genetic operators can be used in order to generate the offspring population. Novel mechanisms are still being explored and developed in the literature (Chuang et al., 2015). Mutation operators from ES usually can change all components of a parent vector at

the same time, but with minor changes since it is assumed that in the real biological evolution small mutations occur frequently but large ones only rarely.

In the 1990s, Cai and Thierauf (1996) proposed a general ES for solving discrete optimization problems, suggesting that not all components of a parent vector should be mutated, but only a few should be randomly changed every time. This strategy was quite smart, since, in discrete sets, differences between any two adjacent values are usually not small. This approach has been used/followed in different applications (Hasancebi, 2007; Chen and Chen, 2009; Yao et al., 2011). Li et al. (2013) introduced a Mixed Integer ES capable of handling parameters consisting of discrete and integer variables.

Motivated by successful applications in the literature, it was felt that the ES could deal with combinatorial optimization problems that Coelho, Souza et al. (2011) in partnership with a self-adaptive strategy based on moves generated from simple NSs, where neighborhood structures guide the individuals through the solution space. Thus, a special care in the design of the algorithm was given, exploring the RVNS ability of handling random moves in N_k, $k = 1, \ldots, k_{max}$. The details will be discussed later in this article. The use of NS is well known in the literature and has been widely applied for solving $\mathcal{NP}$-Hard problems (Johnson et al., 1988; Kirkpatrick, 1984; Glover, 1989; Mladenović and Hansen, 1997; Lourenço et al., 2003; Lust and Teghem, 2010; Pisinger and Ropke, 2010).

In this sense, we provide a compact and efficient encoding for adapting NS use and strength in connection with individual mutation operators, described in the next section.

2.1 Mutation Operators

According to the problem that is being solved, a desired data structure, or *representation*, is selected and a solution to the problem is defined as s. Here, each individual ind is comprised of two additional mutation vectors, defined as P and A, presented in Equations (1) and (2) respectively, embedded within the solution representation s.

Following this strategy, each individual of the population is defined as described in Equation (3) as a triple formed by s, P, and A.

$$P = [p_1, p_2, \ldots, p_k, \ldots, p_{NSmax}] \tag{1}$$

$$A = [a_1, a_2, \ldots, a_k, \ldots, a_{NSmax}] \tag{2}$$

$$ind = (s, P, A) \tag{3}$$

The first mutation vector, P, represents the likelihood associated with the choice of each NS in a set composed of $NSmax$ neighborhood structures. This vector guides the probability of applying each NS used to walk on the search space. Each position stores the probability $p_k \in [0, 1]$, $p_k \in \mathbb{R}$ of the application of a given move $m \in N_k$.

The second mutation vector, A, stores integer values for controlling the strength of the disturbance, once an NS is selected to be applied and shake the solution s. Each position $a_k \in [0, nap_k]$, $a_k \in \mathbb{N}$ of this vector indicates the number of random moves $m \in N_k$ to be applied from neighborhood k, with nap_k representing the maximum *number of applications* of different moves in each mutation event.

Both vectors are adapted across the generations of the evolutionary process, according to well-known probability distribution functions. The evolution of these two mutation vectors will be discussed for each application described in this current study. Other sets of parameters could also be included for adapting the distributions along the generations.

Algorithm 1: GES

Input: greedy parameter γ, Function $f(.)$, population size μ, offspring size λ, random individuals selected for local searching κ
Input: N neighborhoods
Output: Population Pop

1 **for** $i \leftarrow 1$ ***to*** μ **do**
2 Generate a random number $\gamma \in [0, 1]$
3 $s \leftarrow$ GRASP(γ)
4 $(P, A) \leftarrow$ BuildMutationVectors($|N|$)
5 $ind \leftarrow (s,\ P,\ A)$
6 $Pop_i \leftarrow ind$
7 **end**
8 **while** *stop criterion not satisfied* **do**
9 **for** $i \leftarrow 1$ ***to*** λ **do**
10 $ind \leftarrow$ Random individual Pop_x with $x \in [1, \mu]$
11 $ind' \leftarrow$ UpdateParameters(ind, σ_{real}, $\sigma^{p}_{binomial}$,$\sigma^{n}_{binomial}$, $|N|$)
12 $ind'' \leftarrow$ ARVNS(ind', N)
13 $Pop_i^{offsprings} \leftarrow ind''$
14 **end**
15 **for** $i \leftarrow 1$ ***to*** κ **do**
16 Generate a random number $x \in [1, \lambda]$
17 localSearchProcedure($Pop_x^{offsprings}$) – (optional)
18 **end**
19 Pop = Selection (f, Pop, $Pop^{offsprings}$)
20 **end**
21 **return** *Pop*

2.2 Generic Evolution Strategy Pseudocode Guideline

The proposed self-adaptive evolution strategy algorithm pseudocode is outlined in Algorithm 1. As emphasized by Lust and Teghem (2010), generating an initial population diversified and with good potential is a very important feature for the convergence of population-based algorithms. Thus, we suggest the GRASP procedure in partnership with the proposed algorithm.

The initial population (lines 1 to 7 of Algorithm 1) consists of generating a set of μ individuals. Line 3 calls the GRASP procedure and generates each solution of the initial population with different random greedy parameter γ. Achieving a diversified initial population is an important stage for the algorithm convergence, as can be verified in Lust et al. (2011); therefore, different γ parameters are used in order to control the size of the candidates' list. Thus, a random GRASP is designed here. In the second step (line 4 of Algorithm 1), the self-adaptive mutation vectors P and A are built for each individual. The procedure BuildMutationVectors (outlined in Algorithm 2) describes a generic and simple idea for generating initial values for the mutation vectors. Line 5 merges the triple formed by a GRASP solution s and the mutation operators P and A. In this sense, the following nomenclature is defined: let ind^S be the solution s of the individual ind; ind^P be the probability parameter vector; ind^A be the application parameter vector.

Algorithm 2: BuildMutationVectors

Input: number of neighborhoods max
Output: Mutation parameters vector P and A

1 $P \leftarrow$ Initialize Vector of Probabilities P for r neighborhoods
2 $A \leftarrow$ Initialize Vector of Applications A for r neighborhoods
3 **for** $k \leftarrow 1$ ***to*** max **do**
4 $\quad P_k \leftarrow$ Generate a random number $\in [0, 1]$
5 $\quad A_k \leftarrow$ Generate a random number $\in [1, nap_k]$
6 **end**
7 **return** *P,A*

Algorithm 3: UpdateParameters

Input: Individual ind, Standard Deviation σ_{real} e $\sigma_{binomial}$, number of neighborhoods max
Output: Individual ind updated

1 **for** $k \leftarrow 1$ ***to*** max **do**
2 $\quad ind_k^P \leftarrow ind_k^P + N(0.0, \sigma_{real})$
3 $\quad ind_k^A \leftarrow ind_k^A + B(0.0, \sigma^p_{binomial}, \sigma^n_{binomial})$
4 **end**
5 Check the limits of the operators ind^P and ind^A
6 **return** ind

Algorithm 2 fills the probabilities vector, P, random numbers generated between the interval [0, 1] and the same idea is applied for the vector of applications, A, respecting the range [1, nap_k].

In line 11 of Algorithm 1, individual parameters are updated; the pseudocode of the procedure "UpdateParameters" is described in Algorithm 3. Vectors of mutation parameters A and P are updated and adapted according to a Normal or Binomial Distribution, both centered at mean zero and standard deviation σ_{real} and $\sigma^p_{binomial}$, respectively. For the binomial distribution, an additional parameter $\sigma^n_{binomial}$ indicating the number of trials is required. Parameter $\sigma^p_{binomial}$ regulates the probability of successes in a sequence of $\sigma^n_{binomial}$ independent yes/no experiments. Updates of the vectors ind^P and ind^A can be viewed in lines 2 and 3, respectively, in Algorithm 3. Line 5 checks if the limits of both mutation operators are respected after the mutation. As expected, the maximum value assigned for the application probability, of each cell from vector ind^P, should be between 0% to 100%. Following a similar reasoning, the maximum number of applications, for each cell k from vector ind^A, should be nap_k and no less than 1.

Line 12 of Algorithm 1 calls the mutation procedure, a special case of the classical RVNS, called ARVNS, illustrated in Algorithm 4. In line 2 of Algorithm 4, a random number $z \in [0, 1]$ is generated and then line 3 checks if this number satisfies the probability ind_k^P. In the positive case, the neighborhood structure N_k, allocated in the current index k, is applied ind_k^A times. The neighborhood order of this parameter vector is chosen at random. An optional mutation rate parameter can be added in each individual representation in order to regulate the sequence in which the NS are applied.

Algorithm 4: ARVNS

Input: Individual ind
Input: N neighborhoods
Output: Individual ind

```
1  for k ← 1 to max do
2      Generate a random number z ∈ [0, 1]
3      if z < ind_k^P then
4          for a ← 1 to ind_k^A do
5              s' ← MOVE_k(ind^S)
6              ind^S ← s'
7          end
8      end
9  end
10 return s
```

Line 17 of Algorithm 1 opens the possibility of calling an optional intensification phase, usually done by local search techniques, such as Variable Neighborhood Descent (VND). When triggered, an intensification phase refines κ random solutions in the offspring population. We emphasize this phase as optional since several variations could be implemented here according to the combinatorial optimization problem that is being tackled.

Finally, the selection procedure, line 19 of Algorithm 1, can be any desired selection strategy, as long as it returns a population with cardinality μ. We used two basic forms of competition, both with the same notation of Beyer and Schwefel (2002). In the first one, denoted by $(\mu + \lambda)$, there is competition between parents and offspring. In this strategy, μ best individuals are selected among the union of parents and offspring. In the second selection strategy, denoted by (μ, λ), only the offspring compete for survival. It is clear that using the strategy $(\mu + \lambda)$ as a way of selection, the population of the next generation suffers a considerably higher selective pressure than using the strategy (μ, λ).

3 Open-Pit-Mining Operational Planning Problem

The OPMOP involves the allocation of mining equipment to the pits, which may be of ore or waste rocks, as well as determining the number of trips for each truck so that both the production goals and the desired mineral composition of the ore are fulfilled. The goal is to find a mining rate on every pit that minimizes deviations from production goals, quality, and also the number of trucks required for the process. Dynamic truck allocation is considered, which is the possibility to allocate the trips from a certain truck to a different pit. This allocation system contributes to an increased fleet productivity and, therefore, to reduce the number of trucks needed for the production process. Figure 1 shows a graphical example of the OPMOP, composed of pits with different mineral compositions, two shovels, and different trucks.

A brief literature review is described next.

Costa (2005) developed a heuristic algorithm based on GRASP and VNS using six different types of movements to explore the solution space. A comparison was made between the results obtained by this heuristic algorithm and those found by the solver LINGO, version 7, applied to a mathematical programming model developed in Costa

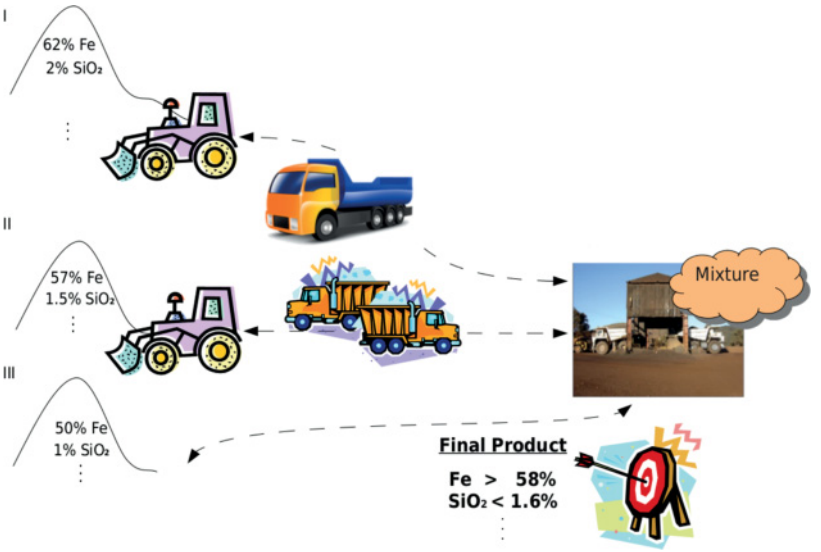


Figure 1: OPMOP example.

et al. (2004). Results showed that the heuristic algorithm was able to find better solutions faster. Guimarães et al. (2007) presented a computer simulation model to validate results obtained by applying a mathematical programming model to determine the mining rate in open-pit mines (e.g., occurrence of queues).

Souza et al. (2010) proposed an algorithm, called GGVNS, which combines the metaheuristics General Variable Neighborhood Search (GVNS) (Hansen et al., 2008) and GRASP procedure. The GVNS was chosen due to its simplicity, efficiency, and capacity of its natural local search to deal with different neighborhoods. The authors compared the results generated by GGVNS with those achieved by CPLEX optimizer 11.0.1, using eight test problems. Computational experiments showed that the algorithm was competitive and, in most instances, capable of finding new optimal solutions—with a gap $< 1\%$—requiring a short computational time.

Coelho et al. (2012) developed the first multi-objective application to the OPMOP. Three multi-objective heuristic algorithms were validated based on Two-phase Pareto Local Search with VNS (2PPLS-VNS), proposed by Lust and Teghem (2010), Multi-objective Variable Neighborhood Search (MOVNS), presented by Geiger (2004), and Non-dominated Sorting Genetic Algorithm II (NSGA-II) developed by Deb et al. (2002). Approximations of Pareto sets generated by the developed algorithms were compared considering the hypervolume and spacing metrics. Computational experiments have shown the superiority of the algorithms based on VNS methods, which were able to find better sets of nondominated solutions, more diversified and with an improved convergence.

3.1 Representation and Evaluation of a Solution

Given a set of mining pits F, a set of trucks T, and a set of shovels K, a solution for OPMOP is represented by a matrix $R = [Y|N]$, where Y is a matrix $|F| \times 1$ and N a matrix $|F| \times |T|$. Each cell y_i of the matrix $Y_{|F|\times 1}$ represents shovel $k \in K$ allocated to the pit $i \in F$. If there aren't trips made to the pit i, the shovel k associated with this

Table 1: Representation of a solution.

	Shovel	$Truck_1$	$Truck_2$	...	$Truck_T$
F_1	$(Shovel_1, 1)$	8	X	...	X
F_2	$(Available, 0)$	0	0	...	0
F_3	$(Shovel_8, 0)$	0	0	...	0
...	...	...	...	...	...
F_F	$(Shovel_5, 1)$	0	9	...	3

pit is considered inactive and it is not penalized for a production below the minimum limit.

In the matrix $N_{|F|\times|T|}$, each cell n_{il} represents the number of trips performed by the truck $l \in T$ to the pit $i \in F$. The value 0 (zero) means no trip to that truck. The value X means that the truck is incompatible with the shovel allocated to the pit.

In Table 1, there is an example of a possible solution to the OPMOP. At the column *Shovel*, line F_1, the pair $(Shovel_1, 1)$, indicates that the loading equipment $Shovel_1$ is allocated to the pit F_1 and the number 1 means that it is operating. At the column *Shovel*, line F_3, the pair $(Shovel_8, 0)$ indicates that the loading equipment $Shovel_8$ is allocated to the pit F_3, but it is not operative. Finally, in line F_2, the value $(Available, 0)$ means that there is no loading equipment allocated to the pit F_2 and, therefore, this pit is available. The other columns represent the number of trips from the truck to the corresponding pit, considering the compatibility between the truck and loading equipment allocated to the front. Cells with values X indicate incompatibility between a truck and its loading equipment.

3.2 Mathematical Model and Solution Evaluation

In the considered formulation of the OPMOP, extracted from Souza et al. (2010), the mono-objective function is given by Equation (4):

$$\min f^{PM}(s) = \sum_{j\in T} \lambda_j^- d_j^- + \sum_{j\in T} \lambda_j^+ d_j^+ + \alpha^- D_O^- + \alpha^+ D_O^+ + \beta^- D_W^- + \beta^+ D_W^+ + \sum_{l\in T} \omega_l U_l. \quad (4)$$

Equation (4) seeks to minimize the positive and negative deviations from the goals of each control parameter j of the mixture, d_j^+ and d_j^-, respectively, as well as the positive and negative deviations from the production goals of ore and waste rocks, represented, in this order, by decision variables D_O^+, D_O^-, D_W^+, and D_W^-. This function also considers the minimization of the number of used trucks, represented by the binary variable U_l, which is 1 if the truck l is used and 0, otherwise.

The constants λ_j^-, λ_j^+, α^-, α^+, β^-, β^+, and ω_l are weights that reflect the importance of each component of the objective function.

Since the movements generated by the used neighborhood structures can lead to infeasible results, a solution is evaluated by a function f to be minimized, composed of two parts. The first one is the actual objective function, f^{PM}, given by Equation (4), and the second one consists of functions that penalize the occurrence of infeasibility in the solution. Thus, the function f, given by Equation (5), measures the deviation of the

goals and penalizes any violation of the constraints in the problem.

$$f(s) = f^{PM}(s) + f^{p}(s) + \sum_{j \in T} f_j^q(s) + \sum_{l \in V} f_l^u(s) + \sum_{k \in C} f_k^c(s), \quad (5)$$

where:

$f^{PM}(s)$ is a function that evaluates s with regard to the production goals and the quality of the final mixture;

$f^{p}(s)$ penalizes s if the limits for the production of ore and waste rocks are not respected;

$f_j^q(s)$ penalizes s if the limits for the j-th control parameter of the mixture are not respected;

$f_l^u(s)$ penalizes s if the maximum utilization rate for the l-th truck is exceeded;

$f_k^c(s)$, penalizes s if the productivity limits for the shovel k are not respected.

3.3 Neighborhood Structures

In order to explore the solution space, as described in Section 2, eight neighborhood structures, introduced by Souza et al. (2010), are used to analyze the convergence of the proposed GES. As detailed in Algorithm 3, the NS are polarized during the GES algorithm execution and may, by chance, "generate/create" new structures. With this new proposed mechanism it is possible to create new ways to shake a given solution, since a solution s', generated from s, is a combination of $[0, nap_k]$ random moves from all $N_k(s)$, as described in Algorithm 4.

A short description of the movements that will guide the GES walk through the solutions space are described next:

Movement number of trips—$NS^{NT}(s)$: This move increases or decreases in one unit the number of trips a truck l performs to a pit i, in which there is a compatible load equipment. Thus, in this movement a cell n_{il} of matrix N has its value increased or decreased by one.

Movement load—$NS^{LD}(s)$: It consists of swapping two distinct cells y_i and y_j of matrix Y, that is, swapping the load equipments allocated to pits i and j, if both pits have an allocated loading equipment. When there is an allocated load equipment on only one of the pits, this movement will relocate the load equipment to the available pit. To maintain compatibility between shovels and trucks, the trips made to the pits are relocated along with the load equipments.

Movement relocate trip from a truck—$NS^{TT}(s)$: In this movement, two cells n_{il} and n_{kl} of matrix N are selected and one unit of n_{il} is transferred to n_{kl}. Thus, the truck l does one trip less to pit i and it does one trip more to pit k. Compatibility between equipments are observed, with relocation of the trip only if there is a match between them.

Movement relocate trip from a pit—$NS^{TP}(s)$: Two cells n_{il} and n_{ik} of matrix N are selected and one unit of n_{il} is relocated to n_{ik}. This move relocates one trip that the truck l performs to pit i for the truck k. Compatibility between equipment restrictions are respected and there is a relocation only when there is a match between them.

Movement pit operation—$NS^{PO}(s)$: This operation consists of removing the load equipment that is operating in pit i. The procedure removes all trips made to the pit i, leaving the shovel equipment allocated to it inactive. The equipment returns to operation only once a new route is associated to the pit.

Movement truck operation—$NS^{TO}(s)$: Consists of removing from operation one truck l that is related to a pit i. Thus, the cell n_{il} of matrix N has its value set to zero.

Table 2: GES proposed variants for the OPMOP.

Acronym	μ	λ	Selection	VND
GES1	30	160	(μ, λ)	
GES2	30	160	$(\mu + \lambda)$	
GES3	100	600	(μ, λ)	
GES4	100	600	$(\mu + \lambda)$	
GES1-VND	30	160	(μ, λ)	$\surd$
GES2-VND	30	160	$(\mu + \lambda)$	$\surd$

Movement swap trips—$NS^{ST}(s)$: Two cells of the matrix N are selected and one unit of a cell is transferred to another one, which means the journey of a truck made to a pit is forwarded to another truck on another pit.

Movement swap shovels—$NS^{SS}(s)$: Two distinct cells y_i and y_k matrix Y have their values exchanged, that is, the load equipments operating on pits i and k are exchanged. Similar to the neighborhood structure NS^{LD}, the load equipments are exchanged, but the trips made to the pits are not. To maintain compatibility between shovels and trucks, the incompatible trips made to the pit are removed.

3.4 Computational Experiments and Analysis

Computational experiments were carried out on a Pentium Core 2 Quad (Q6600) with 8 GB of RAM, and operating system Ubuntu 14.04.

A first batch of experiments, discussed in Section 3.4.1, seeks to find good parameters of population size and selection strategy rates for the OPMOP. Time-to-Target plots (TTTplots) were used in order to find a set of parameters able to find good targeted solutions. This approach was used since we are only trying to demonstrate the convergence of the GES and, if possible, achieve comparable solution with the GGVNS algorithm of Souza et al. (2010). Thus, the main focus of the experiments section is to demonstrate the ability of the proposed algorithm to self-adapt its parameters during the evolution process. This feature is discussed in Section 3.4.2. In addition, the proposed hybrid self-adaptive GES is tested on a set of standard benchmark problem instances from the literature.[1] These test problems were the same used in Souza et al. (2010) to validate the GGVNS algorithm.

3.4.1 GES Calibration Using Time-to-Target Plots

Six variants with different sets of parameters are analyzed here; Table 2 shows the parameters of these GES variants. The variants GES1-VND and GES2-VND include VND as the local search procedure for refining some solutions. In these two variants, a portion of $\kappa = 3$ individuals from the offspring population are chosen for the local search procedure (as suggested in Line 17 of Algorithm 1). Only a small group of those NS described in Section 3.3 is used, namely: NS^{LD}, NS^{NT}, NS^{TT}, and NS^{TP}. The expensive computational cost of the local search justifies this restriction. The maximum number of application was set as $nap_k = 15$ for each $k = 1, \ldots, 8$.

Two TTTplots experiments were performed for checking the efficiency of the proposed variants in achieving targeted solutions. Runtime distributions or TTTplots

[1] Available at http://www.decom.ufop.br/prof/marcone/projects/mining.html

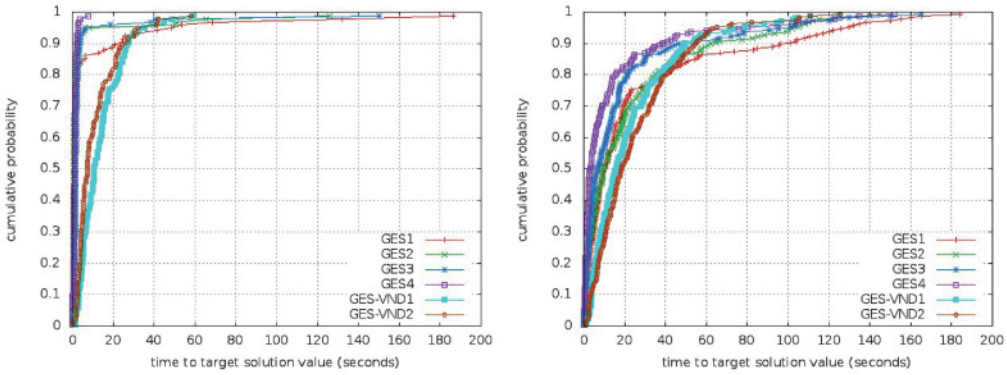


Figure 2: Superimposed empirical distribution.

display, on the ordinate axis, the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis. These plots were first used in Feo et al. (1994). Runtime distributions have been advocated also in Ribeiro and Resende (2011) as a way to characterize the running times of stochastic algorithms for combinatorial optimization.

Aiex et al. (2007) described a Perl program to create TTTplots for measuring times that are assumed to fit a shifted exponential distribution, closely following Aiex et al. (2002). Such plots are very useful to compare different algorithms or strategies for solving a given problem and have been widely used as a tool for algorithm design and comparison.

For a better comparison among the variants, their empirical probability curves were superimposed. On the first experiment, the algorithms were applied to the test problem opm1; the target was set at 230.00 (2% of optimal value), as shown in Figure 2a. In the second one, the algorithms were applied to the instance opm8; the target was set at 164,024.00 (0.0033% of optimal value), as shown in Figure 2b. A battery of 100 executions was made and the performance ended only when the algorithm had found the target value. These times were then sorted in ascending order, and for each algorithm, were associated with the i-th largest running time t_i, a probability $p_i^{TTTplots} = (i - 1/2)/N$ and the points plotted $z_i = (t_i, p_i^{TTTplots})$, for each $i = 1, \ldots, N$. The results of the experiments are shown in Figures 2a and 2b.

Analyzing the empirical probability curves, it is possible to see that the variants that used the selection strategy $(\mu + \lambda)$ prevailed over the versions that used the selection (μ, λ). This fact shows that the competition between parents and offspring made those individuals with a good optimality potential persist for more generations. This result is consistent with the report of Beyer and Schwefel (2002) and Herdy (1992), recommending the use of the $(\mu + \lambda)$ selection in discrete finite size search spaces.

In Figure 2a there is a total supremacy of the GES4, achieved by its selection strategy $(\mu + \lambda)$ combined with population size $\mu = 100$ and $\lambda = 600$. Since the initial instants of the search, the variant GES4 was able to generate better solutions than the other algorithms proposed. However, analyzing the curves of Figure 2b, one can notice that from 60 seconds and on, GES4 lost its performance, being surpassed by the variant

Table 3: Convergence of the estimation of $Pr(X_i \leq X_j)$.

	GES1	GES2	GES3	GES4	GES1-VND	GES2-VND
GES1		48.23%	42.74%	32.17%	66.34%	63.13%
GES2	50.16%		43.77%	34.19%	65.58%	62.65%
GES3	56.05%	53.94%		**37.68%**	72.52%	69.91%
GES4	**65.68%**	**61.72%**	**59.25%**		**80.28%**	**78.68%**
GES1-VND	33.52%	34.15%	27.28%	19.36%		45.85%
GES2-VND	36.83%	37.27%	30.02%	21.20%	54.15%	

GES2-VND, which continues to progress systematically, being the first to reach the desired target with a probability of approximately 100%.

In order to deal with the situation shown in Figure 2b, a probability experiment according to Ribeiro and Rosseti (2009) is presented. Let A1 and A2 be two stochastic search algorithms applied to the same problem and let X1 and X2 be the continuous random variables represeting the time required for algorithms A1 and A2, respectively, to find a solution as good as the given target. Ribeiro and Rosseti (2009) developed a numerical tool to calculate the probability of the runtime of the algorithm A1 being less than or equal to the runtime of the algorithm A2, that is, $Pr(X_1 \leq X_2)$. This tool approximates the absolute error in the integration, by selecting the appropriate value of ϵ. The latter optimizes the resulting approximation errors, called $\Delta(\epsilon)$, in order to make it sufficiently small. Using this tool to validate the analysis of the empirical experiment in Figure 2b, Table 3 was generated.

Analyzing Table 3, it appears that even though the variant GES2-VND has a greater probability of finding the target starting from 60 seconds of execution, the variant GES4 has a probability of 78.68% to have the runtime less than or equal to the variant GES2-VND. In addition, it's clear that the variant GES4 outperforms all other variants.

This first set of experiments allows us to define, at least, a first set of parameters for the hybrid self-adaptive GES able to converge in a similar computational time than the one used by the literature. From now on, the variant GES4 will simply be named GES.

3.4.2 Evolution Strategy Self-Adaptive Mechanism

In order to verify the effect of the maximum number of application for each NS of the OPMOP, a first batch of experiments composed of 1,800 executions, 225 for each of the eight instances, was performed with different limits nap_k (as presented in Section 2.1). Objective functions were normalized for each instance and an Analysis of Variance (ANOVA) test (Shapiro and Wilk, 1964) was done for analyzing the differences between the limits nap_k. Figure 3 shows an effect plot with limits $nap_k = 1, nap_k = 15, nap_k = 100$, $nap_k = 1000$, for $k = 1, \ldots, 8$. As can be noticed, the only significant difference detected, with 95% confidence level, was the worse performance of the GES with strict limits $nap_k = 1$. We believe that the model could be free to adapt the NS application naturally; thus, we might have left it with a large number $nap_k = 1000$. However, in order to keep the disturbances slighter, we decided to keep the maximum number of application for each nap_k as 15.

Instance opm1 was now solved and the mean values of the mutation vectors of typically 120 seconds execution are reported. The mutation operators were able to adapt its application probability during the evolutionary process, as can be seen in Figure 4.

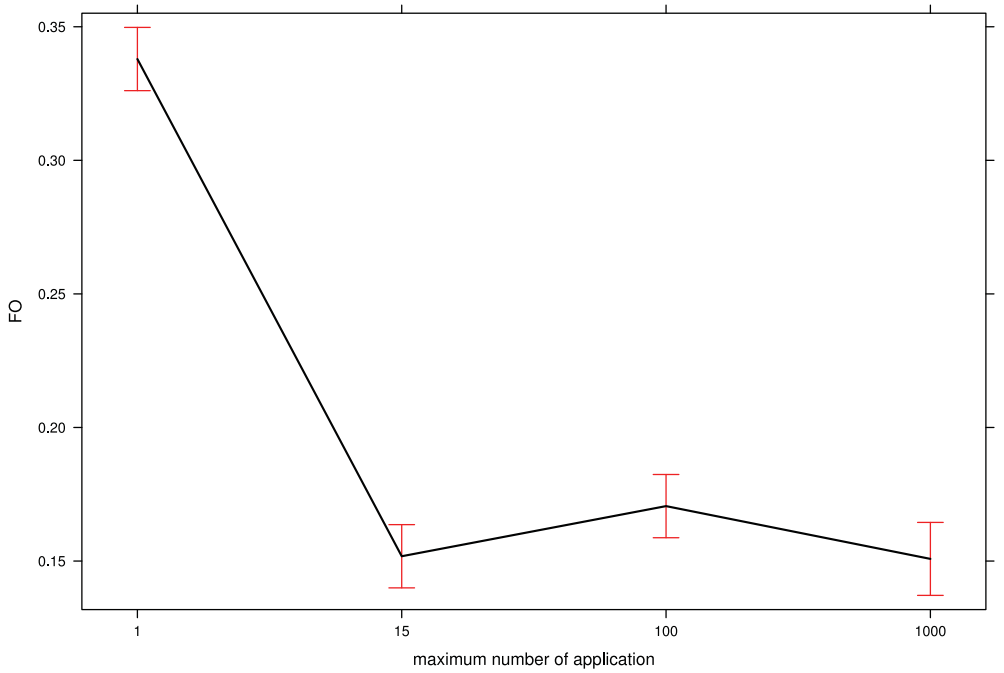


Figure 3: Effect of the maximum number of application nap_k for each NS.

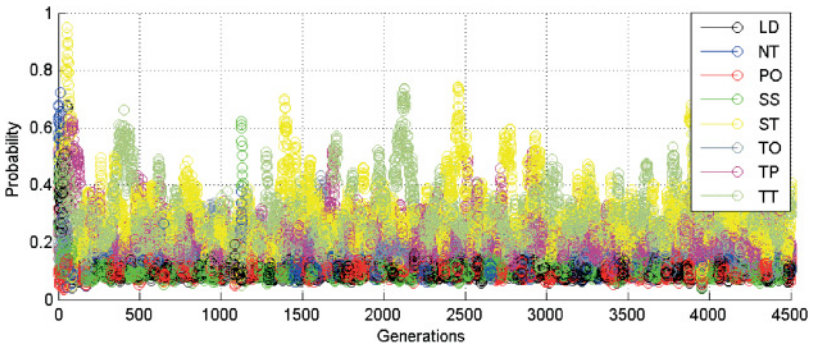
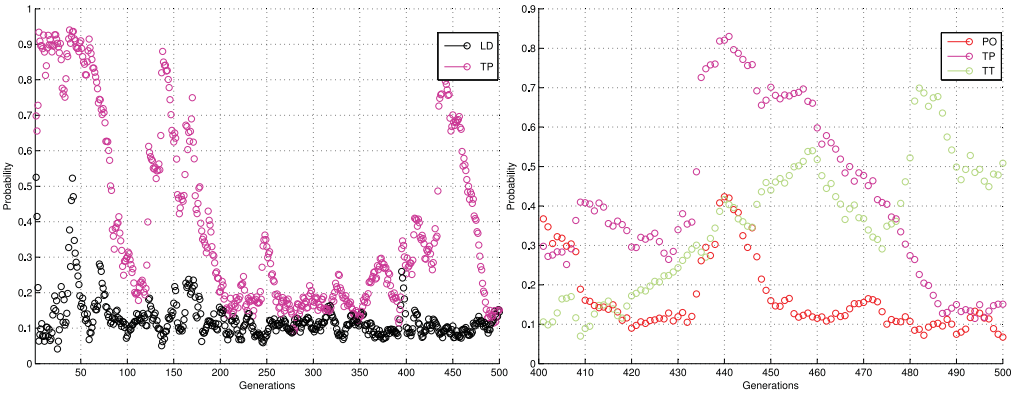


Figure 4: Mutation operators evolution—OPMOP.

A special relationship was detected between some NS; thus, two other plots (see Figures 5a and 5b), were generated focusing only on the specific ability of combining NS. As described in Section 3.3, the neighborhood $NS^{LD}(s)$ is able to swap shovels from different active pits. Due to the different ore composition among pits, it is interesting to reallocate trips for improving the final quality of the mixture after a swap done by $NS^{LD}(s)$. The GES was able to increase the number of moves from the neighborhood $NS^{TP}(s)$ at the same time that the probability of applying swaps from $NS^{LD}(s)$ increased. This was an interesting fact, since we verified that improvements on the trip allocation had to be performed and $NS^{TP}(s)$ was able to attend to these changes respecting the other pits, shovels, and trucks.



(a) Operators NS^{LD} and NS^{TP} (b) Operators NS^{PO}, NS^{TP} and NS^{TT}

Figure 5: Mutation operators evolution.

Another interesting behavior of the mutation operators was detected between the neighborhood $NS^{PO}(s)$, able to remove a pit from operation, and the neighborhoods that deal with trucks' trips $NS^{TP}(s)$ and $NS^{TT}(s)$. Again, GES was able to take profit of free trips from the trucks now free due to the higher probability of using $NS^{PO}(s)$ moves.

3.4.3 Benchmark Results

Finally, in this section, the GES performance is verified against two algorithms already applied for the OPMOP solution. A batch of 30 executions was done and objective functions values were calculated for comparing the performance of the proposed model over each of the eight test problems. Both algorithms from the literature, used for comparison, are based on VNS methods. The GGVNS algorithm is a trajectory search–based algorithm and its best results and average objective function in 30 executions were also previously reported. The second one is a multi-objective algorithm, so-called G2PPLS-VNS, for which only the best results were reported.

In Table 4, the column Instance indicates the test problems used. The column imp^d refers to the relative average percentage values gain of the GES algorithm against the GGVNS; that is:

$$imp_i^d = \frac{\bar{f}_i^{GGVNS} - \bar{f}_i^{GES}}{\bar{f}_i^{GGVNS}}. \tag{6}$$

The column imp^b exhibits the percentage of improvement afforded by the GES algorithm regarding the value of best solution found by the GGVNS algorithm.

$$imp_i^b = \frac{f_i^{\star GGVNS} - f_i^{\star GES}}{f_i^{\star GGVNS}}. \tag{7}$$

According to Table 4, we could verify that the proposed GES algorithm was able to generate good quality solutions with low variability, being competitive with the GGVNS algorithm. It was noticed that GES was able to slightly improve the quality of final solutions up to 0.87% and reduce the variability of these solutions up to 0.70%. Furthermore, the GES algorithm was able to find a better solution than the best one known by the literature, for instance, opm5. A similar behavior happened in the batch of

Table 4: Experimental results: GES applied in the OPMOP.

	GGVNS		G2PPLS	GES			
Instance	Average	Best	Best	Average	Best	imp^b	imp^d
opm1	230.12	230.12	**228.12**	**228.50**	**228.12**	**0.87**	**0.70**
opm2	256.56	**256.37**	**256.37**	**256.43**	**256.37**	0.00	**0.05**
opm3	164064.68	164039.12	164046.32	**164044.68**	**164031.28**	0.00	**0.01**
opm4	164153.92	164099.66	164074.32	**164097.61**	**164057.04**	**0.03**	**0.03**
opm5	228.09	228.09	227.04	**227.21**	**226.66**	**0.63**	**0.39**
opm6	237.97	**236.58**	236.35	**237.07**	**236.58**	0.00	**0.38**
opm7	164021.89	164021.38	164018.81	**164020.24**	**164018.22**	0.00	0.00
opm8	164027.29	164023.73	164022.63	**164022.38**	**164020.26**	0.00	0.00

experiments performed in Section 5.3.1. Thus, even though the proposal uses the same NS of both algorithms in analysis, individuals reached, guided by GES self-adaptive strategy, different solutions result in the search space.

4 Short-Term Load Forecasting Problem

The importance of load forecasting has been increasing lately and improving the use of energy resources remains a great challenge to the emerging Smart Grid (SG) systems. Energy consumption forecasting in the context of economic development of a country was highlighted by Lee and Tong (2011). SG are considered the future of power grids, able to manage production, transmission, and electricity distribution. The task of optimizing the SGs has been done mainly by using an Artificial Intelligence (AI) technique (Raza and Khosravi, 2015; Olivares et al., 2014; Rigo-Mariani et al., 2014).

In this sense, improving the calibration of a forecasting model through the aid of an evolutionary algorithm seems reasonable. Here, we verify the ability of the GES in calibrating the Hybrid Forecasting Model (HFM) proposed by Coelho et al. (2014a; 2014b; 2016). In its previous version, the model's fuzzy rules were being calibrated by a trajectory search–based algorithm and a classic evolution strategy using a mutation matrix fulfilled with standard deviations.

We will use this problem as a didactic case of study that combines the use of nine different NS. An acceptable convergence of the model is checked and verified in Section 4.3.2.

4.1 Representation and Evaluation of a Solution

A solution to the HFM is represented as a matrix of continuous values indicating fuzzy rules intervals and respective weights. An example of a solution representation s with three columns can be seen in Figure 6. For the batch of experiments analyzing the mutation operators, we are going to use a solution with several different inputs and approximately 1,000 columns to be calibrated.

The evaluating process is simple; the solution depicted in Figure 6 is evaluated by its ability in forecasting a given validation historical load time series. The rules of the matrix are applied considering previously measured data. Results of the combination of all rules and its weights give the next point forecast. These are very similar to artificial neural network–based models (Drezga and Rahman, 1999) and fuzzy time series (Song

$$s = \left[\begin{array}{c|ccc} & z(K-1) & z(K-2) & \frac{z(K-1)+z(K-2)}{2} \\ A & 87 & 95 & 103 \\ V & 70 & 80 & 95 \\ B & 100 & 90 & 110 \\ W & 110 & 50 & 80 \end{array}\right]$$

Figure 6: Metaheuristic fuzzy model solution, adapted from Coelho et al. (2014b).

and Chissom, 1993). Errors between each forecast and the real measured point from the validation set are calculated using quality indicators (Goodwin and Lawton, 1999).

4.2 Neighborhood Structures

We designed a new didactic NS able to change each cell of the fuzzy matrix s with a disturbance X. The NS will be called $NS^{add_X}(s)$ and is described next:

Movement add *X*—$NS^{add_X}(s)$—This move increases or decreases the value of a random cell of the rules and weights matrix of a solution s.

From this proposed NS, we generated nine different structures with different disturbances parameters X : $NS^{add_{0.1}}(s)$, $NS^{add_1}(s)$, $NS^{add_{\frac{M}{15}}}(s)$, $NS^{add_{\frac{M}{6}}}(s)$, $NS^{add_{\frac{M}{2}}}(s)$, $NS^{add_M}(s)$, $NS^{add_{2M}}(s)$, $NS^{add_{5M}}(s)$, and $NS^{add_{BM}}(s)$. Some of them use the average values of the historical load time series, namely M, as disturbance value. The special character B in $NS^{add_{BM}}(s)$ indicates a big value multiplied by the average M.

4.3 Computational Experiments and Analysis

Computational experiments were carried out on an Intel Core i7-3537U CPU (2.00 GHz), with 4 GB of RAM, on the operating system Ubuntu 14.04. For simplicity, the configuration achieved after the OPMOP TTTplots calibration (Section 3.4.1) and confirmed in literature instances is used here.

The dataset used to check the mutation operators was kindly provided by Liu et al. (2014). It is composed of microgrid user data from a small residential area with maximum load of 273 KW. The dataset is composed of 1,368 hourly samples for training and 672 samples used as blind validation. Section 4.3.2 reports obtained Mean Absolute Percentage Error (MAPE) for the validation set, as a way of certifying the success of the GES in calibrating the metaheuristic fuzzy model.

4.3.1 Evolution Strategy Self-Adaptive Mechanism

A first batch of two-minutes training, typical for online microgrids load forecasting, was performed and the behavior of the mutation operators are discussed in Figures 7, 8a, and 8b.

Figure 7 plots the vector of probability multiplied by the current number of applications of each NS; average values for the population are presented for each generation. The ability of the GES mutation mechanism to regulate the probability of the NS regarding its power of disturbance is highlighted. Moves that slightly change the solution matrix s are the most likely to be applied, as is the case of $NS^{add_{0.1}}(s)$, $NS^{add_1}(s)$, and $NS^{add_{\frac{M}{15}}}(s)$. On the other hand, $NS^{add_{BM}}(s)$ was adapted and adjusted, through the generations, to be rarely applied.

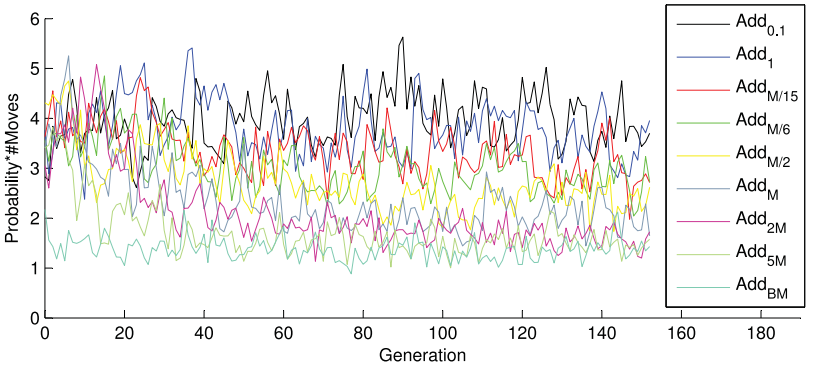
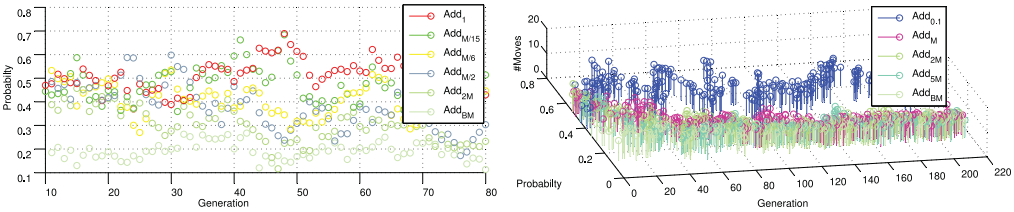


Figure 7: Mutation operators evolution $P \times A$—STLFP.



(a) Application probability of specific NS

(b) Stem plot showing NS groups

Figure 8: Analysis of NS in STLFP.

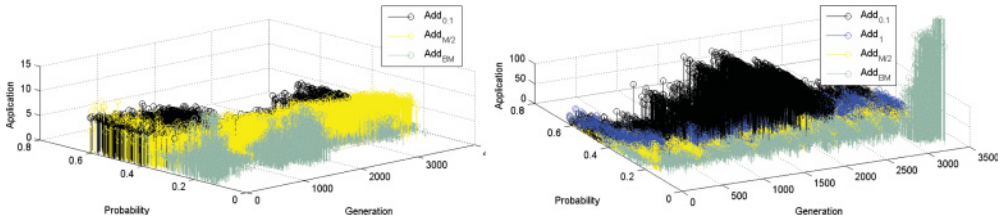
Figures 8b and 8a show groups of NS aggregated regarding the impact they have in mutating the solutions.

The stem plot (see Figure 8b) shows the higher number of moves that are applied for some NS and differences between application probabilities.

Another batch with two-hour training was performed. Figure 9a shows the average values with $nap_k = 15 \quad \forall \quad k = 1, \ldots, 8$ and Figure 9b presents the results for a larger nap limit of 1,000. We highlight that the operators seldom present slopes with higher probabilities and more moves applications. Specially, for the neighborhood $NS^{add_{BM}}(s)$ it happened in both cases, but a few generations later it converged to a steady state, returning the average values of the population to low values of application probability. Even after generation 3000, Figure 9b, we believe that the number of applications would follow the same decrease after some generations.

4.3.2 GES Convergence

A batch of 30 executions for the aforementioned dataset was executed and average MAPE errors of 9.5%, 8.5%, 9.8%, and 8.2% were obtained for the 1st, 2nd, 3rd, and 4th weeks, respectively. Thus, as described by Liu et al. (2014), it is known that when the MAPE is less than 10%, the applicability of the forecast model over microgrids becomes interesting and does not increase its cost sharply. The obtained results indicate, again, the ability of the GES in calibrating the matrix of weights and rules and achieving useful forecasting models.



(a) Training with lower limits for nap_k (b) Training with large limits for nap_k

Figure 9: Two-hour training results.

5 Unrelated Parallel Machine Scheduling Problem with Setup Times

The UPMSP-ST tackled here has as the main goal the makespan minimization. This problem has great practical and theoretical importance. It belongs to the $\mathcal{NP}$-hard class, since it can be seen as a generalization of Parallel Machine Scheduling Problem with Identical Machines and without Setup Times (Garey and Johnson, 1979). The UPMSP-ST is found in different industry sectors, such as textile, chemicals, painting, semiconductors, and paper production (Rabadi et al., 2006).

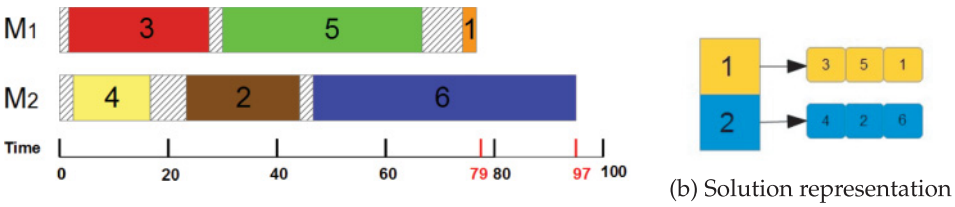
In the UPMSP-ST there is a set of N jobs and a set of M machines, with the following features: (i) each job must be allocated to only one machine; (ii) each job j has a different processing time p_{ij} for each machine $i \in M$; (iii) there is also a setup time S_{ijk} for calibrating the machine i after processing job j and before processing job k; (iv) there is a calibration time S_{i0j} for processing the first job of a given machine $i \in M$.

Different approaches were used for solving the UPMSP-ST; initially, in Al-Salem (2004), the authors developed a heuristic procedure called Partitioning Heuristic and introduced the UPMSP-ST. Rabadi et al. (2006) proposed a metaheuristic for random prioritized search and described a mathematical formulation for the problem. In Ying et al. (2012), a Simulated Annealing algorithm was implemented with a smart strategy for eliminating unpromising jobs. Vallada and Ruiz (2011) analyzed two genetic algorithms and published a benchmark set of instances at the SOA website: http://soa.iti.es/problem-instances. In the works of Haddad et al. (2014), Cota, Haddad, Souza, and Coelho (2014), Cota, Haddad, Souza, and Martins (2014), and Haddad et al. (2015), the authors proposed several trajectory search–based algorithms, named AIV, AIRP, AIA, and HIVP, respectively. These methods were designed based on the ILS and VND, which obtained better results than Vallada and Ruiz (2011), and among them, AIRP had the best performance. The AIRP combines a greedy constructive procedure with the ILS and RIV (described in Section 5.2.1) metaheuristics. Also, periodically, the search is intensified and diversified by a Path Relinking (Glover, 1996) procedure. In this sense, comparing the proposed evolutionary framework against the AIRP (see Section 5.3) is reasonable.

5.1 Representation and Evaluation of a Solution

A solution s for the UPMSP-ST is represented as a vector of integers with m positions, where each position represents one machine. Each active machine is associated with a list containing all jobs allocated to it.

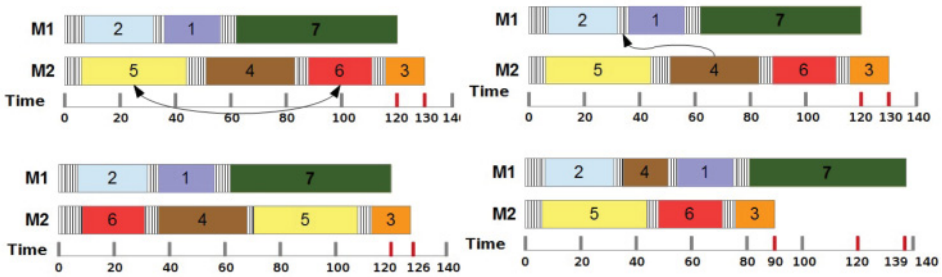
Figure 10a shows an example of a possible scheduling for an instance with two machines and six jobs. In this example, machine M_1 will process jobs 3, 5, and 1, in



(a) Example of a possible scheduling

(b) Solution representation

Figure 10: Example of solution representation.



(a) Swap in same machine

(b) Multiple Insertion

Figure 11: Example of operators (Haddad et al., 2015).

this order; and machine M_2, in turn, will process 4, 2, and 6, following this order. The conclusion time of machine M_1 is calculated by the expression $C_{M_1} = 1 + 28 + 3 + 38 + 8 + 1 = 79$, while machine M_2 conclusion time is given by $C_{M_2} = 2 + 17 + 7 + 21 + 2 + 48 = 97$.

Figure 10b illustrates the solution representation of the example given in Figure 10a. Jobs 3, 5, and 1 are scheduled to machine M_1 while the rest are scheduled to M_2.

In this single objective optimization, solution s is evaluated by the makespan, in other words, by the processing time of the last machine to finish its jobs.

5.2 Neighborhood Structures

Three well-known NS are tested in this application and they are described as follows:

Movement swap in the same machine—$NS^{SSM}(s)$: The movement of swapping jobs in the same machine originates an $N^{SSM}(s)$ neighborhood. It consists of changing the positions of two jobs that belong to the same machine.

Figure 11a illustrates the swap of jobs 5 and 6 in machine M_2.

Movement swap between different machines—$NS^{SMD}(.)$: Similar to the $NS^{SSM}(s)$, but swapping one job from one machine with another job that belongs to a different machine.

Multiple insertion—$N^{MI}(.)$: It consists of relocating one job from one machine to any position of all machines.

Figure 11b shows how $N^{MI}(.)$ works; job 4 from machine M_2 is transferred to machine M_1, just before job 1.

Table 5: GES proposed variants for the UPMSP-ST.

Acronym	μ	λ	Selection	RVI
GES_1	50	150	$(\mu + \lambda)$	$\surd$
GES_2	20	60	$(\mu + \lambda)$	$\surd$
GES_3	50	150	(μ, λ)	$\surd$
GES_4	20	60	(μ, λ)	$\surd$
GES_5	100	600	$(\mu + \lambda)$	$\surd$
GES_6	100	600	(μ, λ)	$\surd$
GES_7	100	600	$(\mu + \lambda)$	
GES_8	100	600	(μ, λ)	

5.2.1 RIV as an Optional Local Search Intensification Phase

The optional local search phase of the GES, Line 17 of the Algorithm 1, activates the proposal of Cota, Haddad, Souza, and Coelho (2014), namely RIV. The latter is inspired by the ILS with a Random VND.

The RIV uses the neighborhoods $N^{MI}(.)$ and $NS^{SSM}(s)$ with first and best improvements strategies, respectively, and explores, sporadically, the $NS^{SMD}(.)$ neighborhood for perturbing the solutions.

5.3 Computational Experiments and Analysis

The proposed algorithm for the UPMSP-ST was implemented in JAVA using the Netbeans 8.0.2 IDE. Computational experiments were carried out on a Core i7 (1.9 GHz) with 6 GB of RAM and Windows 7.

The method was re-implemented, following OptFrame first version in C++, and the ideas introduced in Section 2. This seems to be a good opportunity for analyzing and ensuring the convergence of the proposal in a different environment, considering new codes programmed in a similar language.

5.3.1 Benchmark Results

Eight different configurations of the GES algorithm were designed, as described in Table 5. Six, among these variants, were allowed to activate RIV local search in $\kappa = 3$ individuals from the offspring population, picked at random (see Section 3.4.1). The other two variants, with the same population size as the ones used for the previous case studies were analyzed without the RIV intensification phase.

The stopping criterion adopted was the processing time, given by $executionTime = n \times (m/2) \times t$ milliseconds, with n the total number of jobs, m the total number of machines, and three different values for t (10, 30, and 50). These values adopted for t were the same ones by Cota, Haddad, Souza, and Coelho (2014).

A batch of 30 executions was performed using 36 SOA instances. These instances involve combinations of 50, 100, and 150 jobs with 10, 15, and 20 machines. The average values of each instance were calculated and the deviations between the best results for each instance were measured. Due to the stochastic character of the search, the different GES configurations were executed 30 times for each instance and for each t value, as has been done already for the AIRP.

Figure 12 shows a box plot graph with average objective function values.

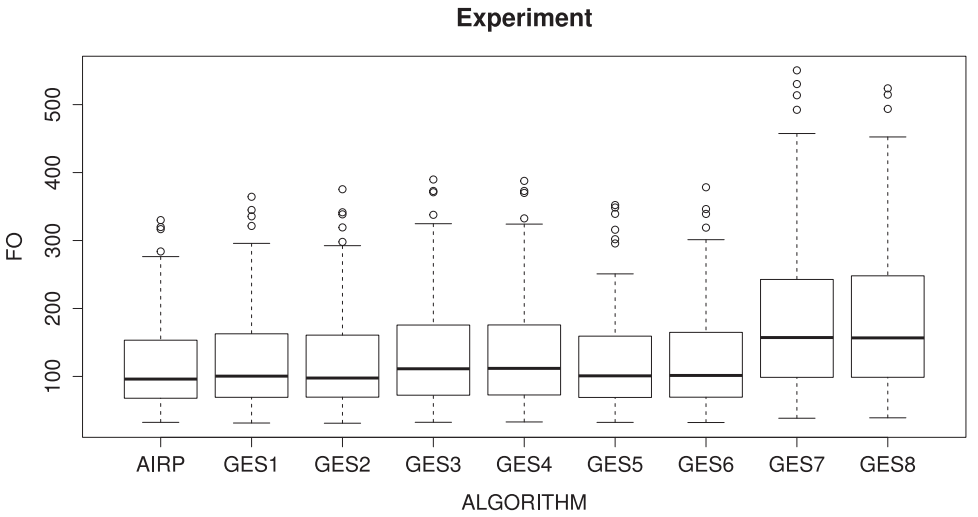


Figure 12: Box plot of the algorithms AIRP, GES_1, GES_2, GES_3, GES_4, GES_5, GES_6, GES_7, and GES_8.

It is noteworthy that the selection strategy $(\mu + \lambda)$ was again able to direct the evolutionary process to better performance. We were able to obtain better solutions (10% better) than the ones found by the AIRP. This fact shows the potential of the proposal to combine NS and to find new solutions in the search space.

It should be noticed that the two variants with the intensification procedure RIV achieved better average objective function values. This fact induces that the RIV procedure was well designed by Cota, Haddad, Souza, and Coelho (2014), being able to collaborate within the search as an intensification phase. Furthermore, it gives a brief motivation for using other smart strategies, based on metaheuristic procedures, in partnership with the proposed GES.

5.3.2 Evolution Strategy Self-Adaptive Mechanism

Three different sizes of instances (large, medium, and small size) were used for analyzing the mutation operators. Parameters $\mu = 100$ and $\lambda = 600$ were kept for this analysis. We aim now at two variants, verifying the convergence with and without the RIV intensification phase. For the sake of clarity, the best results among the two independent runs (using different seeds) are depicted in Figures 13, 14, and 15.

In these figures, the lower bounds (some are optimum values) were plotted for enhancing bounds comprehension of the mutation operators' behavior. The evaluation of the best known solution can be followed across the generations, exhibited with blue lines. Values were normalized for each of the three instances by dividing them by the maximum makespan found in the first generation.

The convergence of the variant using RVI was close to the lower bound in all executions. Figure 13 shows how operators keep trying to escape from local optimum, considerably improving the best known solution until generation 150.

Figure 14 depicts an execution applied for solving a medium size instance, in which the population was guided only by the ARVNS. Analyzing Figure 14b, it can be verified that the mutation operators suffered an intensive change after generation 50. This fact

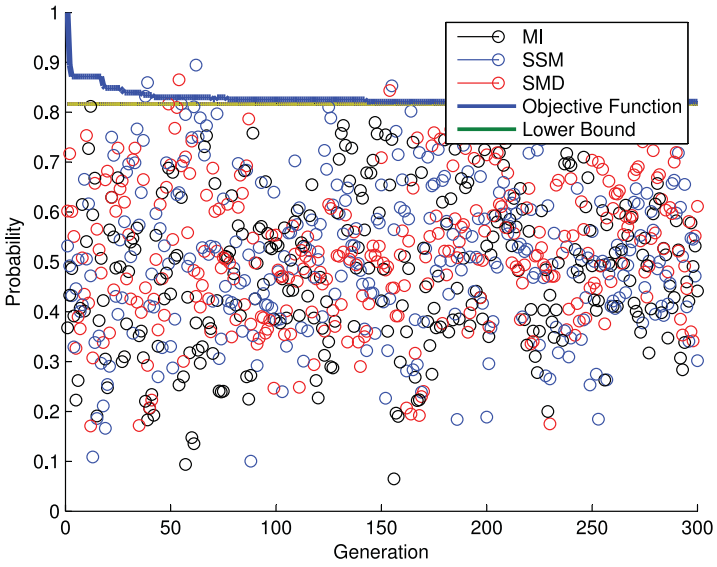
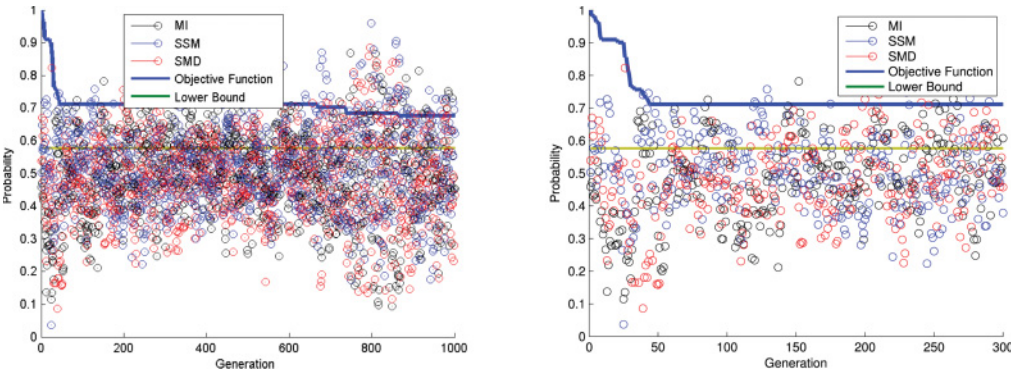


Figure 13: Average mutation operators' values evolution in a Large instance of the UPMSP-ST using $\kappa = 3$.



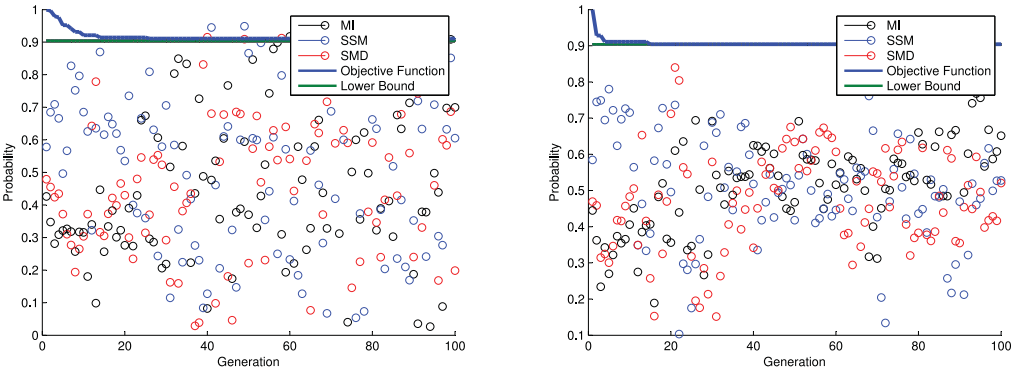
(a) From generations 1 to 1000

(b) From generations 1 to 300

Figure 14: Average mutation operators' values evolution on a Medium instance of the UPMSP-ST without using RVI.

might have happened because of a local trap, which the population was able to escape near generation 600 (see Figure 14a). Four other improvements were also achieved from generation 600 to 1,000.

Beyond a shadow of a doubt, it can be noted that the mutation operators became more randomized after finding local traps. For the cases shown in Figures 15a and 15b, both variants were able to reach the global optimum after generations 30 and 20, respectively.



(a) Without RVI

(b) With RVI

Figure 15: Average mutation operators' values evolution on a Small instance of the UPMSP-ST.

6 Conclusions and Extensions

A new hybrid self-adaptive algorithm based on the concepts of evolution strategies was introduced in this article. Three different combinatorial optimization problems were used as cases of study. For each of them, algorithm convergence and mutation operators' behavior were analyzed.

The results have shown that the proposed evolutionary method is able to achieve competitive solutions. For the OPMOP, the proposed GES algorithm was compared with two other algorithms from the literature. Computational experiments emphasized the competitiveness of the GES algorithm, since it was able to improve the solutions' quality and reduce their variability.

Even though the average performance of the GES, applied to the UPMSP-ST, was not considerably better than the literature, it was able to enhance the quality of the solutions in 10% of the analyzed problems.

The ability of adapting the probabilities of application was also verified in each of the three analyzed problems. The operators handled different NS and its application across different phases of the evolution process. The flexibility of the proposal makes it suitable for a wide area of practical applications, such as in mining, scheduling, and electric load forecasting.

As could be verified in this article, the self-adaptive ES was able to adapt the mutation operators in such a way that there is a balance between exploration and exploitation throughout the generations of the evolutionary process, being able to escape from local optima attraction basins. This ability is related to the issue of performing small changes in the solution for finding local optima, and, on the other hand, enhancing the probability and strength of the mutation operators reflected in increasing the shaking and hence escaping from local traps. This fact promoted a self-adaptive balance between exploiting an attraction basin and jumping out of it.

Further experiments should focus on how close the solutions are from local optima. Thus, future work might investigate the fitness landscape of the problems discussed here. Improving and designing novel self-adaptive mechanisms is another possible extension, as well as parameter calibration. Finally, we suggest implementing a parallel

version of the GES algorithm in order to take advantage of the multicore technology available in current devices.

Acknowledgments

The authors are indebted to the anonymous reviewers for their constructive suggestions that have helped us improve the original manuscript. The authors acknowledge CNPq (grants 306694/2013-1 and 312276/2013-3) and FAPEMIG (grants CEX PPM 772/15 and APQ-02449-14) for supporting the development of this research. Vitor N. Coelho would like to thank his brother, Bruno, for all his motivation and friendship.

References

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373.

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2007). TTTplots: A Perl program to create time-to-target plots. *Optimization Letters*, 1:355–366.

Al-Salem, A. (2004). Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times. *Engineering Journal of the University of Qatar*, 17(1):177–187.

Aler, R., Galván, I. M., and Valls, J. M. (2012). Applying evolution strategies to preprocessing EEG signals for brain-computer interfaces. *Information Sciences*, 215:53–66.

Andersen, S. B., and Santos, I. F. (2012). Evolution strategies and multi-objective optimization of permanent magnet motor. *Applied Soft Computing*, 12(2):778–792.

Beyer, H. G., and Schwefel, H. P. (2002). Evolution strategies—A comprehensive introduction. *Natural Computing*, 1:3–52.

Cai, J., and Thierauf, G. (1996). Evolution strategies for solving discrete optimization problems. *Advances in Engineering Software*, 25(2-3):177–183.

Chaquet, J. M., and Carmona, E. J. (2012). Solving differential equations with Fourier series and evolution strategies. *Applied Soft Computing*, 12(9):3051–3062.

Chen, T., and Chen, H. (2009). Mixed-discrete structural optimization using a rank-niche evolution strategy. *Engineering Optimization*, 41(1):39–58.

Chuang, Y.-C., Chen, C.-T., and Hwang, C. (2015). A real-coded genetic algorithm with a direction-based crossover operator. *Information Sciences*, 305:320–348.

Coelho, I. M., Munhoz, P. L. A., Haddad, M. N., Coelho, V. N., Silva, M. M., Souza, M. J. F., and Ochi, L. S. (2011). A computational framework for combinatorial optimization problems. In *VII ALIO/EURO Workshop on Applied Combinatorial Optimization*, pp. 51–54.

Coelho, V. N., Coelho, I. M., Coelho, B. N., Reis, A. J., Enayatifar, R., Souza, M. J., and Guimarães, F. G. (2016). A self-adaptive evolutionary fuzzy model for load forecasting problems on smart grid environment. *Applied Energy*, 169:567–584.

Coelho, V. N., Guimarães, F. G., Reis, A. J., Coelho, B. N., Coelho, I. M., and Souza, M. J. (2014a). A general variable neighborhood search heuristic for short term load forecasting in smart grids environment. In *Power Systems Conference (PSC), 2014 Clemson University*, pp. 1–8.

Coelho, V. N., Guimarães, F. G., Reis, A. J. R., Coelho, I. M., Coelho, B. N., and Souza, M. J. F. (2014b). A heuristic fuzzy algorithm bio-inspired by evolution strategies for energy forecasting problems. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 338–345.

Coelho, V. N., Souza, M. J. F., Coelho, I., Guimarães, F. G., Lust, T., and Cruz, R. C. (2012). Multi-objective approaches for the open-pit-mining operational planning problem. *Electronic Notes in Discrete Mathematics*, 39:233–240.

Coelho, V. N., Souza, M. J. F., Coelho, I. M., Guimarães, F. G., and Coelho, B. N. (2011). Evolution strategies applied to a mixed integer programming problem. In *Anais do X Congresso Brasileiro de Inteligência Computacional (CBIC)*, volume 1, pp. 1–8.

Costa, F. P. (2005). Applications of optimization techniques to open-pit-mining problems [In Portuguese]. Master's dissertation, Programa de Pós-Graduação em Engenharia Mineral, Escola de Minas, UFOP, Ouro Preto, Brazil.

Costa, F. P., Souza, M. J. F., and Pinto, L. R. (2004). A model of dynamic allocation of trucks [In Portuguese]. *Revista Brasil Mineral*, 231:26–31.

Costa, L., and Oliveira, P. (2001). Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, 25(10):257–266.

Cota, L. P., Haddad, M. N., Souza, M. J. F., and Coelho, V. N. (2014). AIRP: A heuristic algorithm for solving the unrelated parallel machine scheduling problem. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, pp. 1855–1862.

Cota, L. P., Haddad, M. N., Souza, M. J. F., and Martins, A. X. (2014). A heuristic algorithm for solving the unrelated parallel machine scheduling problem with sequence-dependent setup times [In Portuguese]. In *Proceedings of the 35th Congresso Nacional de Matemática Aplicada e Computacional*.

de Freitas, A. R. R., Guimarães, F. G., Pedrosa Silva, R. C., and Souza, M. J. F. (2014). Memetic self-adaptive evolution strategies applied to the maximum diversity problem. *Optimization Letters*, 8(2):705–714.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Dong, X., Nowak, M., Chen, P., and Lin, Y. (2015). Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem. *Computers & Industrial Engineering*, 87:176–185.

Drezga, I., and Rahman, S. (1999). Short-term load forecasting with local ann predictors. *IEEE Transactions on Power Systems*, 14(3):844–850.

Feo, T. A., Resende, M. G. C., and Smith, S. H. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Freitas, A. R., and Guimarães, F. G. (2011). Originality and diversity in the artificial evolution of melodies. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*.

Garey, M., and Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: W.H. Freeman & Co.

Geiger, M. J. (2004). Randomised variable neighbourhood search for multi objective optimisation. In *Proceedings of the 4th EUME Workshop Design and Evaluation of Advanced Hybrid MetaHeuristics*, pp. 34–42.

Glover, F. (1989). Tabu search—Part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1996). Tabu search and adaptive memory programming—Advances, applications and challenges. In R. R. Barr, R. Helgason, and J. Kennington (Eds.), *Interfaces in computer sciences and operations research*, pp. 1–75. Boston: Kluwer Academic Publishers.

Goodwin, P., and Lawton, R. (1999). On the asymmetry of the symmetric {MAPE}. *International Journal of Forecasting*, 15(4):405–408.

Guimarães, F. G., Campelo, F., Igarashi, H., Lowther, D. A., and Ramirez, J. A. (2007). Optimization of cost functions using evolutionary algorithms with local learning and local search. *IEEE Transactions on Magnetics*, 43(4):1641–1644.

Guimarães, I. F., Pantuza, G., and Souza, M. J. F. (2007). Computational simulation model for solutions validation of the open-pit-mining with dynamic trucks allocation [In Portuguese]. In *Proceedings of the XIV Simpósio de Engenharia de Produção*, volume 1, pp. 1–11.

Haddad, M. N., Cota, L. P., Souza, M. J. F., and Maculan, N. (2014). AIV: A heuristic algorithm based on iterated local search and variable neighborhood descent for solving the unrelated parallel machine scheduling problem with setup times. In *Proceedings of the 16th International Conference on Enterprise Information Systems*, pp. 376–383.

Haddad, M. N., Cota, L. P., Souza, M. J. F., and Maculan, N. (2015). Solving the unrelated parallel machine scheduling problem with setup times by efficient algorithms based on iterated local search. *Lecture Notes in Enterprise Information Systems*, 227:131–148.

Hansen, P., Brimberg, J., Urošević, D., and Mladenović, N. (2009). Solving large p-median clustering problems by primal–dual variable neighborhood search. *Data Mining and Knowledge Discovery*, 19(3):351–375.

Hansen, P., and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467.

Hansen, P., Mladenovic, N., and Pérez, J. A. M. (2008). Variable neighborhood search. *European Journal of Operational Research*, 191:593–595.

Hasancebi, O. (2007). Discrete approaches in evolution strategies based optimum design of steel frames. *Structural Engineering and Mechanics*, 26(2):191–210.

Herdy, M. (1992). Reproductive isolation as strategy parameter in hierarchically organized evolution strategies. In *Proceedings of the 2nd Conference of Parallel Problem Solving from Nature*, volume 2, pp. 207–217.

Hosny, M. I., and Mumford, C. L. (2010). Solving the one-commodity pickup and delivery problem using an adaptive hybrid VNS/SA approach. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part II*, pp. 189–198.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100.

Kashan, A. H., Akbari, A. A., and Ostadi, B. (2015). Grouping evolution strategies: An effective approach for grouping problems. *Applied Mathematical Modelling*, 39(9):2703–2720.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

Lee, Y.-S., and Tong, L.-I. (2011). Forecasting energy consumption using a grey model improved by incorporating genetic programming. *Energy Conversion and Management*, 52(1):147–152.

Li, J., Pardalos, P. M., Sun, H., Pei, J., and Zhang, Y. (2015). Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. *Expert Systems with Applications*, 42(7):3551–3561.

Li, R., Emmerich, M. T. M., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., and Reiber, J. H. C. (2013). Mixed integer evolution strategies for parameter optimization. *Evolutionary Computation*, 21(1):29–64.

Liu, N., Tang, Q., Zhang, J., Fan, W., and Liu, J. (2014). A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids. *Applied Energy*, 129:336–345.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In F. Glover and G. Kochenberger (Eds.), *Handbook of metaheuristics*. Boston: Kluwer Academic Publishers.

Lust, T., and Teghem, J. (2010). Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16:475–510.

Lust, T., Teghem, J., and Tuyttens, D. (2011). Very large-scale neighborhood search for solving multiobjective combinatorial optimization problems. In R. Takahashi, K. Deb, E. Wanner, and S. Greco (Eds.), *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pp. 254–268.

Meyer-Nieberg, S., and Beyer, H.-G. (2007). Self-adaptation in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pp. 47–75. Berlin, Heidelberg: Springer.

Mladenović, N., and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100.

Moscato, P., and Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pp. 105–144. New York: Springer.

Olivares, D., Mehrizi-Sani, A., Etemadi, A., Canizares, C., Iravani, R., Kazerani, M., Hajimiragha, A., Gomis-Bellmunt, O., Saeedifard, M., Palma-Behnke, R., Jimenez-Estevez, G., and Hatziargyriou, N. (2014). Trends in microgrid control. *IEEE Transactions on Smart Grid*, 5(4):1905–1919.

Pisinger, D., and Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, pp. 399–419. New York: Springer.

Prado, R. S., Silva, R. C. P., Neto, O. M., Guimarães, F. G., Sanches, D. S., London, J. B. A., Jr., and Delbem, A. C. B. (2014). Differential evolution using ancestor tree for service restoration in power distribution systems. *Applied Soft Computing*, 23:498–508.

Qaurooni, D., and Akbarzadeh-T, M.-R. (2013). Course timetabling using evolutionary operators. *Applied Soft Computing*, 13(5):2504–2514.

Rabadi, G., Moraga, R. J., and Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1):85–97.

Rajasekaran, S. (2006). Optimal mix for high performance concrete by evolution strategies combined with neural networks. *Indian Journal of Engineering and Materials Sciences*, 13(11):7–17.

Raza, M. Q., and Khosravi, A. (2015). A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50:1352–1372.

Resende, M. G. C., and Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau and J. Potvin (Eds.), *Handbook of metaheuristics*, pp. 283–319. New York: Springer.

Ribeiro, C. C., and Resende, M. G. C. (2011). Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, pp. 1–22.

Ribeiro, C. C., and Rosseti, I. (2009). Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. In *Proceedings of the VIII Metaheuristics International Conference*, pp. 1–6.

Rigo-Mariani, R., Sareni, B., Roboam, X., and Turpin, C. (2014). Optimal power dispatching strategies in smart-microgrids with storage. *Renewable and Sustainable Energy Reviews*, 40:649–658.

Schneider, M., Stenger, A., and Hof, J. (2014). An adaptive vns algorithm for vehicle routing problems with intermediate stops. *OR Spectrum*, 37(2):353–387.

Shapiro, S. S., and Wilk, M. B. (1964). An analysis of variance test for normality(complete samples). PhD thesis, JSTOR.

Smith, J. E. (2008). Self-adaptation in evolutionary algorithms for combinatorial optimisation. In *Adaptive and multilevel metaheuristics*, pp. 31–57. Berlin, Heidelberg: Springer.

Song, Q., and Chissom, B. S. (1993). Fuzzy time series and its models. *Fuzzy Sets and Systems*, 54(3):269–277.

Souza, M. J. F., Coelho, I. M., Ribas, S., Santos, H. G., and Merschmann, L. H. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051.

Vallada, E., and Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622.

Wanner, E. F., Guimarães, F. G., Takahashi, R. H. C., and Fleming, P. J. (2008). Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria. *Evolutionary Computation*, 16(2):185–224.

Xiao, Y., Kaku, I., Zhao, Q., and Zhang, R. (2011). A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research*, 214(2):223–231.

Yao, L., Zhang, A.-L., and Yang, H.-J. (2011). Discretion variable optimization design of prestressed steel truss with stress constrains. *Beijing Gongye Daxue Xuebao/Journal of Beijing University of Technology*, 37(3):368–374.

Ying, K.-C., Lee, Z.-J., and Lin, S.-W. (2012). Makespan minimisation for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, 23(5):1795–1803.