Diversity-Driven Selection Operator for Combinatorial Optimization*

Eduardo G. Carrano[1][0000−0003−3368−4245],
Felipe Campelo[1,2][0000−0001−8432−4325], and
Ricardo H. C. Takahashi[3][0000−0003−0814−6314]

[1] Department of Electrical Engineering, Universidade Federal de Minas Gerais - Brazil. egcarrano@ufmg.br
[2] School of Engineering and Applied Science, Aston University - UK
f.campelo@aston.ac.uk
[3] Department of Mathematics, Universidade Federal de Minas Gerais - Brazil.
taka@.mat.ufmg.br

Abstract. A new selection operator for genetic algorithms dedicated to combinatorial optimization, the *Diversity Driven* selection operator, is proposed. The proposed operator treats the population diversity as a second objective, in a multiobjectivization framework. The Diversity Driven operator is parameterless, and features low computational complexity. Numerical experiments were performed considering four different algorithms in 24 instances of seven combinatorial optimization problems, showing that it outperforms five classical selection schemes with regard to solution quality and convergence speed. Besides, the Diversity Driven selection operator delivers good and considerably different solutions in the final population, which can be useful as design alternatives.

Keywords: multiobjectivization · combinatorial optimization · genetic algorithms · selection operator · diversity preservation

1 Introduction

Genetic algorithms (GAs) are known to face the exploitation-exploration dilemma: too much focus on exploitation can result in premature convergence, whereas too much effort in exploration can negatively affect both convergence speed and solution quality. Different approaches have been proposed to address this issue, such as: changes in the population structure [1]; adaptation of crossover or mutation operators [18] or of operator parameters [18]; ranking-based selection schemes [2]; and niche-sharing/crowding-based selection schemes [14]. In those procedures, the algorithm can be adjusted in order to become either more exploitation-oriented or more exploration-oriented, and the choice of a specific setting should be performed specifically for each class of problem. An important drawback of those schemes is that the issue of diversity has been often considered as a merit factor by itself. However, diversity by itself does not provide a

* This work was supported by the Brazilian agencies CNPq, CAPES and FAPEMIG.

considerable contribution for the quality of the final solutions delivered; instead, diversity is valuable when the diverse solutions cover promising areas of search space [18, 31]. Those references have not employed a search mechanism based on multiobjective optimization principles for dealing with those two objectives.

The formal setting of *multiobjective optimization* for handling the objectives of fitness and diversity was firstly introduced by [36]. Within this setting, it becomes possible to properly define which solutions should be found and stored, and to establish suitable search strategies for finding them. Other works followed the same approach of defining a diversity measure as an auxiliary objective in a multiobjective setting within evolutionary algorithms [23, 8, 17, 32, 37], but many rely on diversity metrics that depend on the computation of the distance from all individuals to all other ones, which causes a computational complexity of $\mathcal{O}(n^2)$ [36, 23, 17, 32, 37]. More recent works have focused on the use of multiobjective indicators to optimize diversity [28] or in developing frameworks for the development of algorithms aimed at optimising in the quality-diversity landscape [15], which indicates an ongoing interest into the development of evolutionary methods capable of generating both diverse and high-quality final populations.

This paper proposes a further development on the use of multiobjective optimization principles for achieving *useful diversity* in single-objective combinatorial optimization. A selection operator based on the selection mechanism of NSGA-II [16] is proposed: the *Diversity-Driven* (DiD) selection operator. The DiD operator is parameterless, and can be used as an off-the-shelf operator for building problem-specific GAs, keeping compatibility with almost any crossover, mutation, and local search operators. As a main feature, instead of using a raw distance measure as a new objective, the DiD operator builds an auxiliary objective function that combines the solution ordering induced by the original objective function with the distance from solution to solution, in the genotype space, between solutions that are consecutive in this ordering.To define distance metrics that adhere to combinatorial problems, the DiD operator uses geometric descriptions of combinatorial spaces [27, 12, 25, 10].

The remainder of this paper is structured as follows: Section 2 presents some formal definitions and states the diversity-related objective function used for problem multiobjectivization. Section 3 describes the proposed diversity-driven selection operator (DiD operator). Distance metrics in discrete variable spaces are discussed in section 3.1. The test framework, including problems, instances, algorithms and comparison procedures is presented in section 4. The results achieved by the DiD operator and the other five ones employed for benchmark are discussed in section 5. Some concluding remarks are drawn in section 6.

2 Multiobjective Selection Preference

The optimization problem of interest is defined as stated in (1), where $\mathcal{X}$ represents the space of discrete decision variables, $\mathbf{x} \in \mathcal{X}$ is a point in this space denoted as a *candidate solution* (an *individual*); and $f(\cdot) : \mathcal{X} \mapsto \mathbb{R}$ is an objective function to be minimized.

$$\begin{aligned} &\text{Find:} && \mathbf{x}^* = \arg\min_{\mathbf{x}} \ f(\mathbf{x}) \\ &\text{Subject to:} && \mathbf{x} \in \mathcal{X} \end{aligned} \tag{1}$$

Let $\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ denote a *population* of current candidate solutions. Assume that a *total order* $[\mathbf{x}_1 \leftarrow \mathbf{x}_2 \leftarrow \ldots \leftarrow \mathbf{x}_p]$ is induced on this set by the comparison operator $\leq$ over the objective function values, such that the set is ordered in increasing order of objective function values ($f(\mathbf{x}_i) \leq f(\mathbf{x}_{i+1}) \forall i \leq p-1$). Suppose, *w.l.g.*, that when there are replicas of the same point $\mathbf{x}_i$, those replicas are always ordered with consecutive indices. Although the aim of a single-objective optimization algorithm is to find the minimal element of this order, $\mathbf{x}^*$, it should be noticed that the direct usage of the raw ordering $\mathbf{x}_1 \leftarrow \mathbf{x}_2 \leftarrow \ldots \leftarrow \mathbf{x}_p$ inside the search engine may cause the loss of diversity because nothing prevents the selection of very similar individuals, or even several replicas of the same individual.

The formal statement of orderings that are oriented by more than one objective should consider the dominance relation operator ($\prec$):

$$\mathbf{a} \prec \mathbf{b} \Leftrightarrow \{a_i \leq b_i \ \forall i = 1, \ldots, m \text{ and } \exists j \in \{1, \ldots, m\} \text{ such that } a_j < b_j\} \tag{2}$$

It is possible that neither $\mathbf{a} \prec \mathbf{b}$ nor $\mathbf{b} \prec \mathbf{a}$ is true; in such a case $\mathbf{a}$ and $\mathbf{b}$ are said to be *non-comparable* vectors. Considering a set $\mathcal{S}$ of vectors, in which the $\prec$ operation can be applied, a *minimal element* $\mathbf{a}^*$ within $\mathcal{S}$ is an element for which the following relation holds:

$$\nexists \ \mathbf{a} \in \mathcal{S} \text{ such that } \mathbf{a} \prec \mathbf{a}^* \tag{3}$$

As a consequence of the existence of non-comparable solutions, there may exist several different minimal elements in $\mathcal{S}$. The solution of a multiobjective optimization problem is defined as the set of minimal elements of this partial order in the problem domain, the *Pareto-optimal set* of the problem.

The idea in this paper is to employ, in the selection operator, the ordering established by the dominance relation operator ($\prec$) on the population, considering the objective function and an auxiliary objective stated as a particular distance to a neighbor, as follows. Let the *left neighbor* of an element $\mathbf{x}_i$ be the element $\mathbf{x}_{i-1}$. A function f_α is defined as:

$$f_\alpha(\mathbf{x}_i) = -\|(\mathbf{x}_i) - (\mathbf{x}_{i-1})\| \tag{4}$$

in which $\|\cdot\|$ denotes a norm defined in $\mathcal{X}$, and $f_\alpha(\mathbf{x}_1) = -\infty$ by convention. In situations where multiple copies of the same individual are present in $\mathcal{P}$, the relation $\mathbf{x}_i = \mathbf{x}_{i+1}$ will hold for some pairs of individuals. In this case, $f(\mathbf{x}_i) = f(\mathbf{x}_{i+1})$ and $f_\alpha(\mathbf{x}_{i+1}) = 0$ hold. Now, define f_d as:

$$f_d(\mathbf{x}_i) = \begin{cases} f_\alpha(\mathbf{x}_i), & \text{if} \quad f_\alpha(\mathbf{x}_i) < 0 \quad \text{or} \quad f_\alpha(\mathbf{x}_{i-1}) < 0 \\ f_d(\mathbf{x}_{i-1}) + 1, & \text{otherwise} \end{cases} \tag{5}$$

Function f_d will return smaller values for individuals at a greater distance from their left neighbors, which can be interpreted as a measure of diversity. The following properties of f_d are of interest:

(i) The best individual, $\mathbf{x}_1$, has its smallest possible value: $f_d(\mathbf{x}_1) = -\infty$;
(ii) For any two individuals $\mathbf{x}_i \neq \mathbf{x}_{i+1}$ the condition $f_d(\mathbf{x}_{i+1}) < 0$ holds, even if $f(\mathbf{x}_i) = f(\mathbf{x}_{i+1})$;
(iii) For any set of $j+1$ copies of the same individual, $\mathbf{x}_i = \mathbf{x}_{i+1} = \ldots = \mathbf{x}_{i+j}$ the conditions $f_d(\mathbf{x}_i) < 0$, $f_d(\mathbf{x}_{i+1}) = 0$ and $f_d(\mathbf{x}_{k+i+1}) = f_d(\mathbf{x}_{k+i}) + 1$ for $k = 1, \ldots, j-1$ hold.

The idea of the proposed selection method (the DiD operator) is to apply a non-dominated sorting (NDS) selection procedure [16] to $\mathcal{P}$, using functions $f(\mathbf{x})$ and $f_d(\mathbf{x})$ as the selection criteria.

3 Diversity-Driven Selection Operator

The non-dominated sorting procedure [16] relies on the assignment of a front number to each individual in $\mathcal{P}$, such that the non-dominated individuals are put in front $\mathcal{F}_0$, the individuals that are dominated only by individuals in the front $\mathcal{F}_0$ are put in $\mathcal{F}_1$, the individuals that are dominated only by individuals in the fronts $\mathcal{F}_0$ and $\mathcal{F}_1$ are placed in $\mathcal{F}_2$, and so forth.

By proceeding in this way with functions $f(\mathbf{x})$ and $f_d(\mathbf{x})$, the resulting fronts $\mathcal{F}_i$ are such that the best individual is always the only one in the front $\mathcal{F}_0$, since it dominates all other individuals in $\mathcal{P}$. This ensures that the best individual is always selected. The second front, $\mathcal{F}_1$, will contain the second copy of the best individual (if any), the first copy of the second best individual, and also the first copy of some other individuals (only one copy per individual). The following front $\mathcal{F}_2$ will contain the third copy of the best individual (if any), the second copy of the second best individual (if any), and other individuals. This pattern is repeated in the other fronts until all individuals are placed in some front.

Consider two individuals $\mathbf{x}_b$ and $\mathbf{x}_a$ such that $f(\mathbf{x}_b) > f(\mathbf{x}_a)$. These individuals will be placed in the same front only if $f_d(\mathbf{x}_b) < f_d(\mathbf{x}_a)$, which may happen if: (i) $\mathbf{x}_b$ is more distant from its left neighbor than $\mathbf{x}_a$, or (ii) the previous fronts already selected include more copies of $\mathbf{x}_a$ than copies of $\mathbf{x}_b$. In both cases, the situation can be interpreted as $\mathbf{x}_b$ being more important for diversity than $\mathbf{x}_a$. Otherwise, $\mathbf{x}_b$ will be placed in a front with higher index than $\mathbf{x}_a$.

After finding the k fronts $\{\mathcal{F}_0, \ldots, \mathcal{F}_{k-1}\}$, the individuals within each front $\mathcal{F}_i$ are ordered according to the one-point contribution metric [7]. For two-objective problems, such as the one considered here, the one-point contribution of a given point $\mathbf{x}_a$ is defined as the area A shown in Fig. 1.

The set $\mathcal{S}$ of the selected individuals resulting from DiD selection operator is filled with the first n_s individuals, according to the total ordering that is established considering first the increasing ordering of fronts, breaking the ties by the decreasing ordering of one-point contributions within any front. Both population diversity and objective function values are considered in the selection by the fitness function $f(\cdot)$ and the diversity function $f_d(\cdot)$.

The computational complexity of the DiD operator is found by analyzing the operations within it. Population sorting can be accomplished in $\mathcal{O}(n \log n)$ time. Evaluation of f_d requires $\mathcal{O}(n)$ distance calculations. Efficient implementations

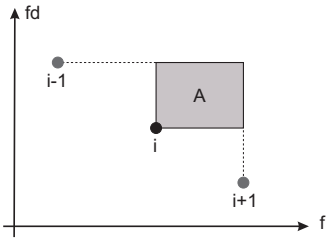

Fig. 1. Evaluation of the one-point contribution for the individual $\mathbf{x}_j$.

of NDS [38] have complexity of $\mathcal{O}(m \cdot n \log n)$, which reduces to $\mathcal{O}(n \log n)$ since $m = 2$. One-point contribution evaluation requires, in the worst case, $n-2$ multiplication operations. The complexity of the DiD selection procedure is therefore determined by population sorting and NDS: $\mathcal{O}(n \log n)$ in the worst case.

3.1 Distance Metrics in Combinatorial Vector Spaces

The proposed selection operator relies on the assumption of availability of a distance between the individuals in the combinatorial problem. A brief explanation of the formalism for defining distances in combinatorial spaces is presented next.

The geometric structure of the space of a combinatorial problem may be stated using the *edit move*, defined as the smallest change of variables that transforms one solution into another one [27]. A *path* $\mathbf{P}$ from $\mathbf{x}^a$ to $\mathbf{x}^b$ is a sequence of solutions that start in $\mathbf{x}^a$ and reaches $\mathbf{x}^b$ by edit move steps, and the length of a path is the number of edit moves that generate all solutions on the path. The *distance* from $\mathbf{x}^a$ to $\mathbf{x}^b$ is the length of the path of minimum length.

A more finely grained definition of a norm can be stated, the *weighted norm*, as proposed in [12] for tree networks. Such a definition starts with the definition of weights for the several components of the decision vector. Each component of each vector receives its weight, such that $x_i^a \mapsto w_i^a$. Those weights are defined such that the components whose change have a greater impact in the individual phenotype receive greater weight. The possibility of defining such weights depends on the problem. Then, the lengths of the paths between individuals become weighted. Let $\ell_w(\cdot)$ denote a weighted length of a path, and let $\mathbf{x}^b$ be generated from $\mathbf{x}^a$ by an edit move that changes only the components i and j. The weighted length of this edit move is defined as:

$$\ell_w(\mathbf{x}^a - \mathbf{x}^b) = w_i^a + w_i^b + w_j^a + w_j^b \tag{6}$$

The weighted length of a path $\mathbf{P} = \left\{\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^m\right\}$ in which $\mathbf{p}^1 = \mathbf{x}^a$ and $\mathbf{p}^m = \mathbf{x}^b$ becomes defined by the sum of the weighted lengths of the edit moves that compose the path:

$$\ell_w(\mathbf{x}^a - \mathbf{x}^b) = \ell_w(\mathbf{p}^m - \mathbf{p}^{m-1}) + \ldots + \ell_w(\mathbf{p}^2 - \mathbf{p}^1) \tag{7}$$

From this definition of weighted path length, comes the weighted distance between individuals.

4 Test Framework

4.1 Test problems

Seven combinatorial problems are considered: (i) degree-constrained minimum spanning tree; (ii) quadratic minimum spanning tree; (iii) optimal communication spanning tree; (iv) linear ordering; (v) common due-date single machine scheduling; (vi) makespan in unrelated parallel machine scheduling, and; (vii) generalized assignment problem. These problems can be divided in three classes: network problems, permutation problems, and assignment problems.

Degree-constrained minimum spanning tree problem (DCMST) is defined as the search for the minimum cost spanning tree with constraints on the degrees of its vertices. In [22], the authors propose a class of instances known as SHRD (structured hard), which are harder than the Euclidean ones. This class is used in the experiments here. Two instances, with 30 (DCMST30) and 50 vertices (DCMST50), have been generated, following the procedure described in [22].

Quadratic minimum spanning tree problem (QMST) involves searching for a spanning tree which minimizes a quadratic cost function that depends on the cost of edges and on the interaction between each pair of edges [33]. Instances with 25 (QMST25) and 50 nodes (QMST50) [12] are considered in this study.

Optimal communication spanning tree problem (OCST) which aims at obtaining the minimum cost spanning tree, with the cost considering the communication requirements between each pair of nodes in the system [33]. Two instances, OCST25 and OCST50 [12], are used in this paper.

Linear ordering (LinOrder) problem [21], in which there is a set of n objects that are to be ordered in a linear sequence. For every pair (i, j) of objects there are coefficients $c_{i,j}$ that define the preference for having i before j. The goal is to find a linear sequence that maximizes the sum of the coefficients that are compatible with the preferences. Six instances from the LOLIB-library [30] are used in this study: be75eec, be75np, be75oi, stabu1, stabu2 and stabu3. The first three have 50 objects and the others have 60 objects.

Common due date single machine scheduling (Scheduling) which consists of the search for the optimal processing sequence for a given set of tasks, in such a way that the weighted sum of earliness and tardiness is minimized. Instances sch50 (50 tasks, $K = 1$ and $h = 0.60$), sch100 (100 tasks, $K = 5$ and $h = 0.60$) and sch200 (200 tasks, $K = 1$ and $h = 0.60$) of the OR-Library [5] are used.

Makespan in unrelated parallel machine scheduling (Makespan) consists in assigning set of tasks to a set of machines, while guaranteeing that each task is assigned to a single machine [29]. In this work the *makespan* is the function to be minimized. Three random instances have been generated: $Mk25 \times 5$ (25 tasks, 5 machines), $Mk50 \times 10$ (50 tasks, 10 machines), $Mk100 \times 10$ (100 tasks, 10 machines). The processing times have been generated as integer numbers in the interval $1 \leq p_{i,j} \leq 10$, following a uniform distribution.

Generalized assignment problem (GAP), which consists in assigning n tasks to m agents. The objective function is the total cost required for performing the

tasks [4]. Six instances from the OR-Library [5] are considered in this study, and coded as "$Am \times n$": $A5 \times 100$ (5 agents, 100 tasks), $A5 \times 200$, $A10 \times 100$, $A10 \times 200$, $A20 \times 100$ and $A20 \times 200$.

4.2 Algorithms

Four different genetic algorithms are used for solving the problems described above. These algorithms vary on the crossover, mutation, and local search operators employed, which are chosen according to the specific problem under consideration, but have exactly the same generational structure. The operators employed by each algorithm are briefly described below:

VETreeOpt (DCMST, QMST and OCST) is an algorithm employed for network problems. It has been built with the same crossover (binary) and mutation (unary) operators as proposed in [12]. These operators have been chosen because they have achieved very good performance on this class of problems;

PermutationGA (LinOrder and Scheduling) is an algorithm used for permutation problems, in which each individual is encoded as a permutation. Partially Matched Crossover (PMX) [19] and Swap Mutation [19] are employed to perform crossover and mutation. In the local search operator, all first order swap operations are tested, one by one, until the first improvement is reached.

AssignmentGA (Makespan) encodes the individuals as chains of integer numbers, in which the integer lying in position i (x_i) identifies the machine that should perform task i. Crossover and mutation are performed using Uniform Crossover [35] and Flip Mutation [13]. The local search operator evaluates the tasks (one by one) on every machine, until it detects the first improvement.

GAPGA (GAP) is very similar to the AssignmentGA, since Makespan and GAP have similar structures. The only difference of GAPGA is the mutation operation, which is performed using the technique proposed in [34]. This operator has been designed specifically for GAP, and cannot be employed on unrelated parallel machine scheduling.

Each one of those algorithms is tested with six selection operators:

- DiD: the Diversity-Driven operator proposed in this paper;
- SW: stochastic wheel without ranking [20];
- SWLR: stochastic wheel with linear ranking [2];
- SUS: stochastic universal sampling without ranking [24];
- SUSLR: stochastic universal sampling with linear ranking [3];
- ST: stochastic binary tournament [20, 24].

Therefore, six different versions of each algorithm (one for each selection operator) are employed for solving each instance. All algorithms are executed considering the same parameters: population size of 50 individuals; stop criterion: $1000n$ function evaluations (FEs)[4]; crossover: 0.80 per pair; mutation: 0.40 per

[4] n is the problem size: *number of vertices* for DCMST, QMST and OCST; *number of objects* for LinOrder. *number of tasks* for Scheduling, Makespan and GAP.

individual; interval between local searches: 50 generations; maximum number of FEs used in local search: 500; algorithm runs per instance: 50. These settings are in agreement with what is commonly employed in the literature. In addition, preliminary testing of the algorithms showed that they are generally robust to small variations on these parameters.

4.3 Performance assessment

Two merit criteria are employed for comparing the selection operators:

Convergence Quality Criterion (CQC): the quality of the outcomes delivered by the algorithm is estimated using the objective function value of the best individual obtained at the end of each algorithm run. For each algorithm and for each run, the value of the objective function of the final best individual is stored. After r runs, the CQC performance of an algorithm on a given instance is determined from the r objective function values stored.

Convergence Speed Criterion (CSC), which estimates how fast an algorithm converges to a reasonable solution. For each algorithm and for each run, the best individual at the end of each generation and the number of function evaluations required for reaching that solution are stored. The worst final solution amongst all runs of all algorithms is found. Then, for each algorithm and for each run, the number of function evaluations that has been required to reach the first solution that is better than or equal to that worst one is found. The CSC metric is determined from this information.

The CQC and CSC values achieved by the algorithms in each individual instance are analyzed using the procedure described below:

1. For CQC and CSC criteria:
   (a) For each Instance k:
      i. For each pair of operators Op_i, Op_j, with $i < j$, estimate the p-value for the comparison of means of the algorithm equipped with operators Op_i and Op_j ($pv_{i,j}$) based on Welch's t-test [26].
      ii. Using the p-values obtained on step 1(a)i, build a partial order for the methods using the Benjamini-Hochberg Multiple Comparison procedure [6], considering a global significance $q^* = 0.05$.
   (b) Based on the partial orders achieved for CQC and CSC on step 1(a)ii, use the Pareto dominance principle to identify the set of efficient algorithms.

The output of this procedure is a set of algorithms which are efficient with regard to CQC and CSC. An algorithm i is efficient if no other algorithm j is better than i in one criterion without being worse in the other one. This comparison approach is similar to the one proposed in [11]. The only difference of the newer procedure is the replacement of the ANOVA and Tukey tests by the Welch's t-test and the Benjamini-Hochberg Multiple Comparison procedure, which makes the process more robust to heteroscedasticity [9].

This procedure makes it possible to establish a ranking of the algorithms: (i) algorithms that are not significantly outperformed by any others are ranked

in the first position; (*ii*) the algorithms that lose only to rank 1 methods are placed on rank 2, and so forth. In the specific case of this work, the statistical comparisons generate two orders for the algorithms, one for CQC and other for CSC. These orders are considered in a Pareto dominance analysis, in order to find the set of efficient operators in each instance.

5 Results

A summary of the results achieved in all instances is shown in Table 1. The DiD operator was not outperformed by any other selection operator, staying in the first position for all instances in both criteria, CQC and CSC. In addition, it was the only operator that was efficient in all 24 instances. Amongst the other five operators, SUSLR was efficient in 8 instances, SWLR and ST were efficient in 7, and SW and SUS were Pareto-optimal in only a single instance each.

Table 1. Operator positions in each criterion for each instance

	CQC						CSC					
Instance	DiD	ST	SUS	SUSLR	SW	SWLR	DiD	ST	SUS	SUSLR	SW	SWLR
DCMST30	1	2	3	2	3	2	1	2	3	2	3	2
DCMST50	1	2	3	2	3	2	1	2	3	2	3	2
QMST25	1	1	1	1	1	1	1	1	2	1	2	1
QMST50	1	1	1	1	1	1	1	2	4	2	4	3
OCST25	1	1	2	1	2	1	1	1	1	1	1	1
OCST50	1	1	1	1	1	1	1	2	2	2	2	2
be75eec	1	1	1	1	1	1	1	1	2	1	2	1
be75np	1	2	2	2	2	2	1	1	2	1	2	1
be75oi	1	2	2	2	2	2	1	1	2	1	2	1
stabu1	1	1	1	1	1	1	1	1	2	1	3	1
stabu2	1	1	1	1	1	1	1	1	2	1	3	1
stabu3	1	2	2	2	2	2	1	1	2	1	3	1
sch50	1	1	1	1	1	1	1	2	3	1	3	2
sch100	1	1	2	1	2	2	1	2	4	2	4	3
sch200	1	1	2	1	2	1	1	1	2	1	2	1
Mk25×5	1	2	1	2	2	2	1	2	2	2	2	2
Mk50×10	1	4	2	6	3	5	1	2	2	2	2	2
Mk100×10	1	3	2	5	2	4	1	1	1	1	1	1
A5×100	1	1	1	1	1	1	1	2	2	2	2	2
A10×100	1	1	1	1	1	1	1	3	2	3	3	3
A20×100	1	1	1	1	1	1	1	1	1	1	1	1
A5×200	1	1	1	1	1	1	1	3	2	3	3	3
A10×200	1	2	2	2	2	2	1	3	2	2	3	3
A20×200	1	2	2	2	2	2	1	2	2	2	2	2

The genetic algorithm using the DiD operator was also able to determine new "best known solutions" for two scheduling instances, namely sch50 and

sch100, when compared to the current best reported in the OR-Library (see `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo.html`):

- **sch50** (k = 1, h = 0.60), former best: 17990, new best: **17976**
- **sch100** (k = 5, h = 0.60), former best: 55291, new best: **55286**

The genetic algorithms using other selection operators were not able to achieve those new best objective function values.

6 Conclusions

This paper proposed a multiobjectivization approach for dealing with the exploration/exploitation dilemma in genetic algorithms for combinatorial problems. The trade-off between population fitness and diversity is processed inside a single parameter-less operator, the *Diversity-Driven (DiD) selection operator*, which selects the individuals according to the objective function and a function that is built over a specific measure of distance to one neighbor. The specific neighborhood structure induced by the solution ordering scheme employed within DiD allows the computation of distances with $\mathcal{O}(n \log n)$ worst-case complexity. The DiD operator was conceived as an off-the-shelf operator that can be used within problem-specific GA's while keeping compatibility with almost any crossover, mutation and local search operators.

Numerical experiments were performed on 24 instances of seven different combinatorial problems, for which some specific GA's were built and equipped with either the DiD operator or several other existing selection methods. The comparisons were performed with regards to convergence speed and solution quality. The results obtained indicate that the proposed DiD operator was able to significantly and consistently outperform the competing selection approaches in both merit criteria.

Additional studies are needed in order to assess the behavior of DiD operator on combinatorial problems with different structures, for instance in the case of satisfiability problems (maximum satisfiability, constraint satisfaction). It is expected that the development of distance metrics specially adapted for different problem structures will allow the application of the DiD operator in a greater variety of situations.

Other extensions of the DiD operators may be interesting to explore. These include the incorporation of preference information into the operator, to allow differential weighting of the dual objectives of solution quality or diversity; and the exploration of this operator in other problem contexts, such as continuous optimization.

References

1. Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook on Evolutionary Computation. Oxford University Press (1997)

2. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proc. 1st Int. Conf. Genetic Algorithms. pp. 101–111 (1985)
3. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proc. 2nd Int. Conf. Genetic Algorithms Appl. pp. 14–21 (1987)
4. Balachandran, V.: An integer generalized transportation model for optimal job assignment in computer networks. Oper. Res. **24**, 742–759 (1976)
5. Beasley, J.E.: OR-Library. http://people.brunel.ac.uk/ mastjjb/jeb/orlib/schinfo.html, last accessed at 2020-Sep-20
6. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. Royal Statistical Soc. – Series B **57**(1), 289–300 (1995)
7. Brandstreet, L., While, L., Barone, L.: A fast incremental hypervolume algorithm. IEEE Trans. Evol. Comp. **12**(6), 714–723 (2008)
8. Bui, L.T., Abbass, H.A., Branke, J.: Multiobjective optimization for dynamic environments. In: Proc. IEEE Cong. Evol. Comp. (CEC 2005). vol. 3, pp. 2349–2356. Edinburgh, UK (2005)
9. Campelo, F., Takahashi, F.: Sample size estimation for power and accuracy in the experimental comparison of algorithms. Journal of Heuristics **25**(2), 305–338 (2019)
10. Carrano, E.G., Ribeiro, G., Cardoso, E., Takahashi, R.H.C.: Subpermutation based evolutionary multiobjective algorithm for load restoration in power distribution networks. IEEE Trans. Evol. Comp. **20**, 546–562 (2016)
11. Carrano, E.G., Wanner, E.F., Takahashi, R.H.C.: A multicriteria statistical based comparison methodology for evaluating evolutionary algorithms. IEEE Trans. Evol. Comp. **15**(6), 848–870 (2011)
12. Carrano, E.G., Takahashi, R.H.C., Fonseca, C.M., Neto, O.M.: Nonlinear network optimization – an embedding vector space approach. IEEE Trans. Evol. Comp. **14**(2), 206–226 (2010)
13. Chicano, F., Alba, E.: Exact computation of the expectation curves of the bit-flip mutation using landscapes theory. In: Proc. Genetic Evol. Comp. Conf. (GECCO 2011). pp. 2027–2034. Dublin, Ireland (2011)
14. Cioppa, A., De Stefano, C., Marcelli, A.: Where are the niches? Dynamic fitness sharing. IEEE Trans. Evol. Comp. **11**(4), 453–465 (2007)
15. Cully, A., Demiris, Y.: Quality and diversity optimization: A unifying modular framework. IEEE Transactions on Evolutionary Computation **22**(2), 245–259 (2018)
16. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comp. **6**, 182–197 (2002)
17. Garcia-Najera, A., Bullinaria, J.A.: An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comp. & Oper. Res. **38**, 287–300 (2011)
18. Ginley, B.M., Maher, J., O'Riordan', C., Morgan, F.: Maintaining healthy population diversity using adaptive crossover, mutation, and selection. IEEE Trans. Evol. Comp. **15**(5), 692–714 (2011)
19. Goldberg, D.E., Lingle, R.: Alleles, loci, and the traveling salesman problem. In: Proc. Conf. Genetic Algorithms Appl. pp. 154–159 (1985)
20. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1st edn. (1989)
21. Grötschel, M., Jünger, M., Reinelt, G.: Optimal triangulation of large real world input-output matrices. Statistische Hefte **25**, 261–295 (1984)

22. Krishnamoorthy, M., Ernst, A.T., Sharaiha, Y.M.: Comparison of algorithms for the degree constrained minimum spanning tree. J. Heuristics **7**, 587–611 (2001)
23. Landa Silva, J.D., Burke, E.K.: Using diversity to guide the search in multi-objective optimization, chap. 30, pp. 727–751. World Scientific (2004)
24. Luke, S.: Essentials of Metaheuristics. Lulu, https://cs.gmu.edu/ sean/book/metaheuristics/Essentials.pdf (2016)
25. Martins, F.V.C., Carrano, E.G., Wanner, E.F., Takahashi, R.H.C., Mateus, G.R., Nakamura, F.G.: On a vector space representation in genetic algorithms for sensor scheduling in wireless sensor networks. Evol. Comp. **22**, 361–403 (2014)
26. Montgomery, D., Runger, G.: Applied Statistics and Probability for Engineers. John Wiley and Sons (2003)
27. Moraglio, A.: Towards a geometric unification of evolutionary algorithms. Ph.D. thesis, University of Essex (2007)
28. Neumann, A., Gao, W., Wagner, M., Neumann, F.: Evolutionary diversity optimization using multi-objective indicators. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM (2019)
29. Pfund, M., Fowler, J.W., Gupta, J.N.D.: A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. J. Chinese Inst. Industrial Engineers **21**, 230–241 (2004)
30. Reinelt, G.: LOLIB. http://comopt.ifi.uni-heidelberg.de/software/LOLIB/, last accessed at 2020-Sep-20
31. Segura, C., Coello Coello, C.A., Miranda, G., Leon, C.: Using multi-objective evolutionary algorithms for single-objective optimization. 4OR **11**(3), 201–228 (2013)
32. Segura, C., Coello Coello, C.A., Segredo, E., Miranda, G., Leon, C.: Improving the diversity preservation of multi-objective approaches used for single-objective optimization. In: Proc. IEEE Cong. Evol. Comp. (CEC 2013). pp. 3198–3205. Cancun, Mexico (2013)
33. Soak, S., Corne, D.W., Ahn, B.: The edge-window-decoder representation for tree-based problems. IEEE Trans. Evol. Comp. **10**, 124–144 (2006)
34. Subtil, R.F., Carrano, E.G., Souza, M.J., Takahashi, R.H.C.: Using an enhanced integer NSGA-II for solving the multiobjective generalized assignment problem. In: Proc. IEEE World Cong. Comput. Intell. Barcelona, Spain (2010)
35. Syswerda, G.: Uniform crossover in genetic algorithms. In: Proc. 3rd Int. Conf. Genetic Algorithms. pp. 2–9 (1989)
36. Toffolo, A., Benini, E.: Genetic diversity as an objective in multi-objective evolutionary algorithms. Evol. Comp. **11**(2), 151–167 (2003)
37. Wessing, S., Preuss, M., Rudolph, G.: Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In: Proc. IEEE Cong. Evol. Comp. (CEC 2013). pp. 103–110. Cancun, Mexico (2013)
38. Zhang, X., Tian, Y., Cheng, R., Jin, Y.: An efficient approach to nondominated sorting for evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation **19**(2), 201–213 (2015)