MOW-P: A simple yet efficient partial neighborhood walk for multiobjective optimization

Matthieu Basseur, Arnaud Liefooghe, Sara Tari

MOW-P: A Simple yet Efficient Partial Neighborhood Walk for Multiobjective Optimization

Matthieu Basseur, Arnaud Liefooghe, Sara Tari
LISIC, Université du Littoral Côte d'Opale, Calais, France
matthieu.basseur@univ-littoral.fr, arnaud.liefooghe@univ-littoral.fr, sara.tari@univ-littoral.fr

***Abstract*—We present a simple neighborhood search that aims at efficiently approximating the Pareto set of particularly difficult multiobjective combinatorial optimization problems. Unlike (elitist) local search which exclusively accepts improving neighbors, the proposed walk explores a subset of λ neighbors only, among which the best one is accepted, whether it improves the current solution or not. This resembles the comma-selection from evolutionary computation, which is to contrast with the plus-selection. This principle was shown to better escape basins of attraction in the context of single-objective optimization. Non-elitism has also been recently revised in the context of multiobjective optimization. However, we emphasize that we do not generate random offspring here. Instead, we create a whole set of λ neighbors among which one is carefully selected, thus providing better control of the selection pressure. In order to extend the partial neighborhood walk principles to multiobjective search, we rely on decomposition: multiple scalarizing sub-problems are uniformly defined and optimized (independently or cooperatively) in order to form a whole approximation set. Based on a benchmark of difficult NK-landscapes with two and three objectives, we show that even independent sub-problem solving results in a clear improvement over more advanced multiobjective decomposition approaches such as MOEA/D. We further report an in-depth analysis of the neighborhood sample size, the number of sub-problems, and the cooperation among them.**

I. Introduction

This work proposes a multiobjective walk based on the partial exploration of the neighborhood (MOW-P). This algorithm finds its inspiration in sampled walk (SW) [1], a straightforward single-objective local search, where a subset of λ neighbors from the current solution is evaluated, among which the best one is selected, whether it results in an improvement or not. This echoes the comma-selection strategy from evolutionary computation. Our aim is to investigate an approach that is particularly simple and easy to implement and configure, but also efficient in obtaining good-quality solutions for difficult multiobjective combinatorial optimization problems. The adaptation of SW to multiple objectives comes quite naturally, given that it already meets these criteria for a single objective. On top of that, it was found to perform a well-balanced search on different classes of combinatorial optimization problems [2]. Meeting these criteria for multiobjective optimization could be even more valuable, as additional challenges arise when optimizing several objectives simultaneously and a whole set of optimal trade-offs among the objectives is to be identified. The definition of a simple algorithm helps reduce the scope of optimization-related issues and focus on multiobjective issues such as selection or archiving.

MOW-P builds upon the concept of *decomposition*, which provides a natural framework for extending single-objective approaches to multi-objective optimization. The most popular decomposition-based method is certainly MOEA/D [3], which has become one of the most widely used multiobjective algorithms from the literature. The MOW-P principles are as follows: a set of scalarizing sub-problems are defined by means of uniformly-generated weight vectors. For each sub-problem, a single-objective walk SW is performed, targeting a given region of the Pareto front. In its basic variant, each walk is performed independently and an archive keeps track of non-dominated solutions. As such, in contrast to MOEA/D, there is *no* cooperation between sub-problems — although we do study the impact of injecting cooperation on algorithm performance. We emphasize that this is therefore a particularly straightforward approach that requires only a couple of parameters: the number of sub-problems and the number of neighbors sampled at each iteration. Despite its simplicity, our experiments on two- and three-objective ρmnk-landscape benchmark instances show the superiority of MOW-P over a more sophisticated algorithm like MOEA/D, especially for difficult problems in terms of ruggedness and multimodality

Unlike the vast majority of multi-objective evolutionary and local search, the MOW-P selection is *non-elitist*. Indeed, current solutions are not necessarily the best found so far and additional mechanisms such as archiving are necessary to avoid the loss of good solutions. Apart from some early work on the topic [4], we remark that non-elitism was recently revisited in the multi-objective optimization literature [5], [6], often leading to an improvement in performance compared against their elitist counterparts. However, we emphasize that MOW-P does not simply perform a generational replacement of the population with respect to random offspring. It indeed selects the best among λ neighbors, thus providing better control of the selection pressure. We further note that decomposition-based multiobjective neighborhood search has been explored previously [7], [8]. However, those attempts follow an elitist strategy, accepting improving neighbors only. This is to contrast with the non-elitist strategy from MOW-P.

The paper is organized as follows. Section II provides the background on multiobjective optimization and SW. Section III presents MOW-P in detail. Section IV is dedicated to the experimental setup while Section V provides experimental results and discussion. The last section concludes the paper and discusses possible ways forward.

II. Background

This section gives definitions for multiobjective optimization and presents the single-objective sampled walk algorithm.

A. Multiobjective Optimization

A multi-objective optimization problem aims to maximize an m-dimensional objective function vector $f\colon \mathcal{X} \to \mathcal{Z}$ such that each solution from $x \in \mathcal{X}$ maps to a vector in the objective space $z \in \mathcal{Z}$, with $z = f(x)$ and $\mathcal{Z} \subseteq \mathbb{R}^m$. Given two objective vectors $z, z' \in \mathcal{Z}$, z is dominated by z' if there is a $j \in \{1, \dots, m\}$ such that $z_j < z'_j$, and $z_i \leqslant z'_i$ for all $i \in \{1, \dots, m\}$. A solution $x \in \mathcal{X}$ is dominated by $x' \in \mathcal{X}$ if $f(x)$ is dominated by $f(x')$. An objective vector $z^\star \in \mathcal{Z}$ is non-dominated if there is no $z \in \mathcal{Z}$ such that $z^\star$ is dominated by z. A Pareto optimal solution $x^\star \in \mathcal{X}$ is a solution such that $f(x^\star)$ is non-dominated. The Pareto set is the set of Pareto optimal solutions, and the Pareto front is its mapping in the objective space. We here aim to approximate the Pareto set.

B. Single-objective Sampled Walk

Sampled walk (SW) is a straightforward local search algorithm for single-objective optimization [1]. As shown in Algorithm 1, it requires a single parameter λ and works as follows: given a solution space $\mathcal{X}$ and a neighborhood relation $\mathcal{N}\colon \mathcal{X} \mapsto 2^{\mathcal{X}}$, SW modifies the current solution by selecting the best among λ randomly-generated neighbors. As SW is straightforward and easy to implement, it can be reminiscent of hill-climbing algorithms. However, it is important to notice that the selected neighbor can *deteriorate* the current solution, allowing the search to bypass local optima and find better solutions. Note that SW is quite similar to `IDbest` [9]. The major difference lies in the fact that the fitness value of the current solution does not influence the selection process of SW.

Despite its high simplicity, SW was shown to perform well compared to well-known generic local search algorithms such as tabu search and iterated local search on a number of single-objective combinatorial optimization problems. Its parameter setting and search trajectories highlight that λ directly controls the balance between exploration and exploitation [10]. The optimal setting for λ, i.e. the proportion of neighbors to evaluate at each step, increases slightly with the size of the search space and its ruggedness, which refers to the number of local optima and their distribution in the landscape. When a proper λ-value is chosen, the search trajectory reveals the ability of SW to find diverse and promising solutions.

We argue that an algorithm based on similar principles to SW for multiobjective optimization could potentially deliver a substantial benefit, provided that simplicity and efficiency can be maintained. Multiobjective optimization induces many additional challenges such as diversity, archiving, or even the pairwise comparison of solutions for selection. Consequently, an approach requiring few design choices while being versatile enough could help maintain focus on these challenging issues.

Algorithm 1 Sampled Walk (SW)

1: Choose $x \in \mathcal{X}$ (initialization)
2: **while** stop criterion not reached **do**
3: $\quad \mathcal{N}_\lambda \leftarrow$ subset of λ random solutions of $\mathcal{N}(x)$
4: $\quad x \leftarrow \arg\max_{y \in \mathcal{N}_\lambda} f(y)$
5: **end while**
6: **return** best encountered solution

III. Multiobjective Sampled Walk

We rely on the decomposition paradigm in order to propose a multi-objective sampled walk based on the partial exploration of the neighborhood, together with a number of archiving techniques to integrate into it. This is developed below.

A. Decomposition in Multiobjective Optimization

The multi-objective optimization problem to be solved can be *decomposed* into a number of single-objective sub-problems that target different regions of the Pareto front. MOEA/D has become one of the most popular decomposition-based multi-objective approach [3]. Each sub-problem is defined by a particular weight vector for the considered scalarizing function. Different scalarizing functions can be used. A popular example is the weighted Chebyshev function:

$$g(x \mid w) = \max_{i \in \{1,\dots,m\}} w_i \cdot \left| z_i^\star - f_i(x) \right| \tag{1}$$

where $x \in \mathcal{X}$ is a solution, $w = (w_1, \dots, w_m)$ is a weighting coefficient vector with $w_i \geqslant 0$ for all $i \in \{1, \dots, m\}$, and $z^\star = (z_1^\star, \dots, z_m^\star)$ is a reference point.

A set of uniformly-generated weight vectors $W = (w^1, \dots, w^\mu)$ can be used to define the scalar sub-problems, for which one solution is maintained and evolved in the population. More particularly, given a scalarizing function $g : \mathcal{X} \mapsto \mathbb{R}$, MOEA/D seeks a solution $x \in \mathcal{X}$ with the best scalarizing function value $g(x \mid w^i)$ for each sub-problem $i \in \{1, \dots, \mu\}$. To this end, it maintains a population $P = (x^1, \dots, x^\mu)$ such that each individual is the current solution for the corresponding sub-problem. Therefore, the population size exactly matches the number of weight vectors $\mu \in \mathbb{N}^+$. Moreover, for each sub-problem $i \in \{1, \dots, \mu\}$, a set of *cooperating sub-problems* $B(i)$ is defined with the t closest weight vectors. The population evolves such that sub-problems are optimized iteratively and cooperatively. At a given iteration, and for a given sub-problem $i \in \{1, \dots, \mu\}$, some solutions are selected from $B(i)$ and an offspring y is created by means of variation operators (crossover and mutation). Next, for every sub-problem $j \in B(i)$, the offspring y replaces the current solution x^j if there is an improvement with respect to the scalarizing function, i.e. if $g(y \mid w^j)$ is better than $g(x^j \mid w^j)$. The algorithm loops over sub-problems, attempting to improve them one after the other until a stopping condition is satisfied. An external archive is optionally maintained to keep track of all non-dominated solutions found so far.

B. Baseline MOW-P Algorithm

We present MOW-P, the proposed multiobjective walk based on partial neighborhood exploration. Similar to MOEA/D, the approach is based on a set of μ weight vectors defining the

Algorithm 2 MOW-P

Define a set W of μ weight vectors
Initialize $\{x^1, .., x^\mu\}$, x^i randomly taken from $\mathcal{X}$
$A \leftarrow \emptyset$
`update_archive`$(A, \{x^1, .., x^\mu\})$
while stopping criterion not reached **do**
    randomly select $i \in \{1, \ldots, \mu\}$ (round-robin)
    $N_\lambda \leftarrow$ subset of λ random neighbors from $\mathcal{N}(x^i)$
    `update_archive` (A, N_λ)
    $x^i \leftarrow \arg\min_{y \in N_\lambda \setminus \{x^i\}} g(y \mid w^i)$
end while
return A

scalar sub-problems. In its default setting, MOW-P simply consists of performing μ *independent* sampled walks — one per sub-problem. The pseudo-code is given in Algorithm 2. Based on a set of weight vectors, MOW-P starts by initializing a population with one random solution per walk and an archive of non-dominated solutions. At each iteration, a sub-problem $i \in \{1, \ldots, \mu\}$ is randomly selected in a round-robin fashion, such that x^i is the current solution for this sub-problem. A subset of λ neighbors from x^i are randomly generated, evaluated, and archived. Among them, the one with the best scalar value for the current sub-problem is selected. The chosen neighbor then replaces x^i, regardless of whether it results in an improvement. The algorithm iterates until a stopping condition is met ; e.g., a maximum evaluation budget.

An important difference with the single-objective SW algorithm concerns the `update_archive` routine. In SW, the archive somehow contains a single solution that corresponds to the best solution encountered since the beginning of the search. In a multiobjective setting, we aim at returning a whole set of mutually non-dominated solutions. We discuss different ways of implementing `update_archive` hereafter.

C. Archiving Strategies

We present three definitions of `update_archive(A,S)`, which aims at updating an archive A w.r.t a set S of candidate solutions. Note that these different implementations do not affect the search process since the only purpose of archived solutions is to be returned at the end of the search process.

1) Single Global Archive: This is the standard method used in most multi-objective approaches. An archive A of mutually non-dominated solutions is maintained and updated after the evaluation of any solution. The archive goes through a filtering process: a solution $x \in S$ is added to the archive if it is not dominated by any solution from A, and solutions from A that happen to be dominated by x are discarded.

Notice, however, that this procedure can be time consuming. Indeed, the archive may contain a large number of solutions, and each new solution must be compared to the whole archive in the worst case. Although several efficient archiving algorithms exist in the literature [11], this remains a recurring issue in multiobjective optimization.

2) One Archived Solution per Sub-Problem: A possibility to reduce the cost of archiving is to store only one solution per walk. This can be seen as an inherent way of bounding the size of the archive [12]. In this case, Pareto dominance is not used, only the best solution found for each scalar sub-problem is kept. When the search process ends, non-dominated solutions among these μ are filtered and returned.

Since the archiving procedure has no impact on the search process, the resulting archive is a subset of the one obtained with the previous approach. A benefit is that the archiving cost is often significantly reduced.

3) One Sub-Archive per Sub-Problem: With this last approach, μ sub-archives $\{A_1, \ldots, A_\mu\}$ are maintained, one per sub-problem. Each archive A_i is being updated with the values successively taken by $N_\lambda \in \mathcal{N}(x^i)$ only. When the search process ends, the sub-archives are then merged and filtered to return a single set of mutually non-dominated solutions.Since each sub-archive focuses on a specific region of the objective space, the size of each one of them tends to be much smaller than the global archive. As such, although μ archives are maintained, we expect each update to be much faster.

Despite the use of multiple sub-archives, the final solutions returned by this strategy exactly matches the ones from the first (global) archiving strategy. Indeed, as there are only external archives in MOW-P, they do not take part in the selection process. The algorithm behavior remains unchanged even if solutions are archived differently. The corresponding MOW-P versions differ only in their computational cost.

IV. Experimental setup

This section describes the experimental setup of our analysis, covering the considered problems and related parameters.

A. Benchmark Problems: ρmnk-Landscapes

We experiment MOW-P on pseudo-Boolean multiobjective multimodal problems with correlated objectives: ρmnk-landscapes [13]. They are described by four parameters: the length of binary strings (solutions) n, the number of objectives to optimize m, the rate of variable interdependency k, and the objective correlation ρ. The objective function vector $f = (f_1, \ldots, f_i, \ldots, f_m)$, to be maximized, is defined as $f\colon \{0,1\}^n \rightarrow [0,1]^m$. For a solution $x = (x_1, \ldots, x_j, \ldots, x_n)$, its objective value $f_i(x)$ is an average value of the individual contributions associated with each variable x_j. The contribution of a variable x_j is determined by its own value and the values of k other variables in interaction with x_j. For each variable x_j, the k variables influencing its contribution are chosen uniformly at random among the $(n-1)$ variables other than x_j [14]. The value of k directly influences the ruggedness of landscapes: $k = 0$ corresponds to a completely smooth landscape while $k = n-1$ corresponds to an extremely rugged landscape. In ρmnk-landscapes, the contribution values follow a multivariate uniform distribution such that $\rho > \frac{-1}{m-1}$ defines the correlation among the objectives [13]. The positive (resp. negative) correlation ρ decreases (resp. increases) the degree of conflict between the objective

TABLE I
INSTANCE PARAMETERS.

description	values
number of variables	$n \in \{128, 512\}$
number of interactions	$k \in \{2, 4, 6, 8, 10\}$
number of objectives	$m \in \{2, 3\}$
objective correlation	$\rho \in \{-0.4, 0.0, 0.4\}$

values. ρmnk-landscapes exhibit different characteristics and degrees of difficulty for multiobjective algorithms [15]. The source code of the benchmark generator is available at the following URL: https://gitlab.com/aliefooghe/mocobench.

B. Parameter Setting

1) Instance Parameters: The benchmark is composed of 48 ρmnk-landscapes generated following the parameters listed in Table I. All parameter combinations have been considered, except $k \in \{4, 8\}$ for $n = 512$. Experiments have been conducted on 30 medium-size instances ($n = 128$) and 18 large-size instances ($n = 512$). This allows us to investigate the behavior of MOW-P on relatively smooth to relatively rugged landscapes, with two and three objectives, and with conflicting, uncorrelated or correlated objectives.

2) Algorithm Parameters: We compare MOW-P against MOEA/D. Both algorithms use the same set of parameter values (when applicable):

- Variation: 1-flip (MOEA/D, MOW-P), uniform crossover and standard bit-flip with rate $1/n$ (MOEA/D only).
- Population size: $\mu \in \{21, 51, 101, 201\}$ for $m = 2$; $\mu \in \{66, 231, 861\}$ for $m = 3$. Values of μ were chosen to ensure weights made of decimal values.
- Number of collaborating sub-problems $t = 0$ (no communication), m, $5 \cdot m$, $\mu - 1$ (full communication).
- Scalarizing function: Chebyshev.
- Sample size: $\lambda \in \{8, 16, 32, 64, 128\}$ (MOW-P only).
- Archiving: (single) global archive (for both algorithms).
- Stopping condition: up to 10^7 evaluations.

We experiment using all possible combinations of these parameter values. This corresponds to 80 (resp. 60) MOW-P and 32 (resp. 24) MOEA/D versions for $m = 2$ (resp. $m = 3$). We perform 20 independent runs for each algorithm and instance. In total, 97 440 runs were performed.

3) Performance Assessment: We report the hypervolume [16] relative deviation `hvrd` covered by the final archive obtained by each algorithm with respect to the best-known Pareto front for the considered instance. The lower the value the better. Thus, `hvrd` $= 0$ means the best-known Pareto front was actually found by the algorithm. The hypervolume reference point is set to the origin. The algorithms were implemented using `Paradiseo` [17], executed on the CALCULCO computing platform, hypervolume computed with the `eaf` package [18], and figures produced with `ggplot` [19].

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we report and discuss a number of experimental results with two goals in mind. The first one is to highlight the efficiency of the simple MOW-P approach. To do so, we compare it against the state-of-the-art MOEA/D. We pay a particular attention to the neighborhood sample size λ, a key feature of the proposed method. The second purpose is to show the straightforwardness of MOW-P. To this aim we analyze the impact of other parameters on its performance.

TABLE II
DEFAULT MOW-P AND MOEA/D CONFIGURATIONS.

algorithm	nb. obj.	neighborhood / mutation	pop. size	nb. sub-problems
MOW-P	$m = 2$	1-flip	$\mu = 21$	–
	$m = 3$		$\mu = 66$	–
MOEA/D `same`	$m = 2$	1-flip	$\mu = 21$	$t = 0$
	$m = 3$		$\mu = 66$	$t = 0$
MOEA/D `default`	$m = 2$	standard $1/n$	$\mu = 101$	$t = 10$
	$m = 3$		$\mu = 231$	$t = 15$
MOEA/D `tuned`	$m = 2$	standard $1/n$	$\mu = 201$	$t = 2$
	$m = 3$	1-flip	$\mu = 861$	$t = 15$

A. Comparison against MOEA/D

In order to ease the readability of results, we start by focusing on the following eight settings:

- Five MOW-P variants corresponding to five λ-values: 8, 16, 32, 64, and 128. The population size is set to $\mu = 21$ for $m = 2$ and $\mu = 66$ for $m = 3$.
- Three MOEA/D variants : `same` refers to the same setting used by MOW-P, `default` refers to MOEA/D's default setting [3], and `tuned` refers to the best overall setting for MOEA/D, one per number of objectives.

These settings are summarized in Table II.

The first set of results is reported in Table III. Given that eight algorithms are compared, each one of them is assigned a rank from 0 to 7, based on pairwise statistical testing. The lower the rank, the better. The rank corresponds to the number of competing algorithms that significantly outperform the one under consideration in terms of hypervolume. Several observations can be drawn from the table. First, there is always at least one MOW-P variant that is not statistically outperformed by any other considered algorithm. Moreover, all MOEA/D variants are statistically outperformed by at least two MOW-P configurations. Noticeably, MOW-P with $\lambda = 32$ is never statistically outperformed by any MOEA/D variant.

Comparing the different instances, the best λ-value slightly increases with the problem size (n) and the interdependency level (k). Interestingly, its does not seem to be affected by the objective correlation (ρ) nor the number of objectives (m). This corroborates results from SW on single-objective nk-landscapes [2]: the λ-value that lead to better solutions also increase with k and n. This goes along with the low impact of the number of objectives on which λ-value to chose.

Fig. 1 reports the empirical attainment function (EAF) differences [18] between MOW-P with $\lambda = 32$ and the tuned version of MOEA/D. The EAF provides the empirical probability distribution that an objective vector is (weakly)

TABLE III

COMPARISON OF MOW-P AND MOEA/D VARIANTS WITH RESPECT TO THE HYPERVOLUME RELATIVE DEVIATION FOR A BUDGET OF 10^7 CALLS TO THE EVALUATION FUNCTION. FOR EACH INSTANCE, THE FIRST VALUE GIVES THE NUMBER OF COMPETING ALGORITHMS THAT SIGNIFICANTLY OUTPERFORM THE ONE UNDER CONSIDERATION ACCORDING TO A (NON-PARAMETRIC) WILCOXON SIGNED RANK TEST AT A SIGNIFICANCE LEVEL OF 0.05 WITH BONFERRONI CORRECTION (LOWER IS BETTER). RANKS IN BOLD CORRESPOND TO APPROACHES THAT ARE NOT SIGNIFICANTLY OUTPERFORMED BY ANY OTHER, THE NUMBER IN BRACKETS IS THE AVERAGE HYPERVOLUME RELATIVE DEVIATION ROUNDED TO THE HIGHEST VALUE (LOWER IS BETTER).

instance				MOW-P $\lambda=8$	MOW-P $\lambda=16$	MOW-P $\lambda=32$	MOW-P $\lambda=64$	MOW-P $\lambda=128$	MOEA/D same	MOEA/D default	MOEA/D tuned
$m=2$	$n=128$	$\rho=-0.4$	$k=2$	1 (0.02)	**0** (0.01)	1 (0.02)	4 (0.05)	6 (0.07)	7 (0.11)	4 (0.04)	2 (0.03)
			$k=4$	2 (0.04)	**0** (0.02)	1 (0.03)	3 (0.06)	6 (0.11)	7 (0.16)	5 (0.08)	3 (0.06)
			$k=6$	3 (0.09)	**0** (0.04)	**0** (0.04)	2 (0.07)	4 (0.11)	7 (0.19)	4 (0.11)	3 (0.10)
			$k=8$	5 (0.13)	1 (0.06)	**0** (0.04)	1 (0.07)	3 (0.10)	7 (0.19)	4 (0.11)	3 (0.11)
			$k=10$	6 (0.14)	2 (0.08)	**0** (0.04)	1 (0.05)	3 (0.09)	7 (0.19)	4 (0.11)	4 (0.12)
		$\rho=0.0$	$k=2$	1 (0.02)	**0** (0.01)	1 (0.02)	3 (0.04)	6 (0.07)	7 (0.11)	4 (0.05)	3 (0.03)
			$k=4$	2 (0.04)	**0** (0.01)	1 (0.03)	3 (0.06)	5 (0.09)	7 (0.16)	4 (0.08)	4 (0.08)
			$k=6$	3 (0.09)	**0** (0.03)	1 (0.04)	2 (0.07)	4 (0.10)	7 (0.19)	4 (0.12)	4 (0.11)
			$k=8$	4 (0.12)	1 (0.05)	**0** (0.04)	2 (0.06)	3 (0.10)	7 (0.19)	4 (0.13)	4 (0.11)
			$k=10$	6 (0.14)	2 (0.08)	**0** (0.04)	1 (0.06)	3 (0.09)	7 (0.19)	4 (0.12)	4 (0.12)
		$\rho=0.4$	$k=2$	1 (0.02)	**0** (0.01)	1 (0.02)	3 (0.03)	6 (0.05)	7 (0.09)	3 (0.03)	3 (0.03)
			$k=4$	2 (0.04)	**0** (0.01)	1 (0.03)	3 (0.05)	5 (0.08)	7 (0.15)	4 (0.07)	4 (0.07)
			$k=6$	3 (0.08)	**0** (0.03)	1 (0.03)	2 (0.06)	4 (0.10)	7 (0.18)	4 (0.11)	4 (0.11)
			$k=8$	4 (0.12)	1 (0.05)	**0** (0.03)	1 (0.06)	3 (0.10)	7 (0.18)	4 (0.12)	5 (0.13)
			$k=10$	5 (0.13)	2 (0.07)	**0** (0.03)	1 (0.05)	3 (0.09)	7 (0.18)	4 (0.13)	4 (0.11)
	$n=512$	$\rho=-0.4$	$k=2$	2 (0.04)	**0** (0.02)	1 (0.03)	4 (0.06)	5 (0.08)	7 (0.10)	5 (0.07)	3 (0.05)
			$k=6$	5 (0.13)	2 (0.05)	**0** (0.03)	1 (0.04)	3 (0.07)	7 (0.15)	5 (0.13)	4 (0.10)
			$k=10$	7 (0.17)	3 (0.10)	1 (0.04)	**0** (0.03)	2 (0.05)	6 (0.14)	5 (0.12)	3 (0.10)
		$\rho=0.0$	$k=2$	2 (0.04)	**0** (0.02)	1 (0.03)	3 (0.05)	5 (0.07)	7 (0.10)	5 (0.07)	3 (0.05)
			$k=6$	5 (0.13)	1 (0.05)	**0** (0.03)	1 (0.05)	3 (0.07)	7 (0.16)	5 (0.13)	4 (0.11)
			$k=10$	7 (0.17)	3 (0.10)	**0** (0.04)	**0** (0.03)	2 (0.05)	6 (0.14)	5 (0.11)	3 (0.10)
		$\rho=0.4$	$k=2$	2 (0.03)	**0** (0.01)	1 (0.03)	3 (0.05)	6 (0.07)	7 (0.10)	4 (0.05)	3 (0.04)
			$k=6$	6 (0.12)	1 (0.04)	**0** (0.02)	1 (0.04)	3 (0.07)	7 (0.15)	4 (0.10)	4 (0.10)
			$k=10$	7 (0.17)	3 (0.09)	1 (0.03)	**0** (0.02)	2 (0.05)	6 (0.14)	3 (0.09)	3 (0.09)
$m=3$	$n=128$	$\rho=-0.4$	$k=2$	1 (0.04)	**0** (0.03)	2 (0.05)	5 (0.08)	6 (0.11)	7 (0.16)	3 (0.07)	3 (0.06)
			$k=4$	2 (0.08)	**0** (0.05)	1 (0.07)	3 (0.11)	6 (0.15)	7 (0.22)	5 (0.13)	3 (0.10)
			$k=6$	4 (0.14)	**0** (0.08)	**0** (0.08)	2 (0.11)	5 (0.16)	7 (0.24)	5 (0.16)	3 (0.13)
			$k=8$	6 (0.18)	1 (0.10)	**0** (0.07)	1 (0.10)	4 (0.15)	7 (0.25)	4 (0.16)	3 (0.13)
			$k=10$	6 (0.19)	2 (0.12)	**0** (0.08)	1 (0.09)	2 (0.13)	7 (0.24)	5 (0.15)	2 (0.13)
		$\rho=0.0$	$k=2$	1 (0.03)	**0** (0.02)	1 (0.03)	5 (0.07)	6 (0.10)	7 (0.16)	3 (0.05)	3 (0.05)
			$k=4$	2 (0.07)	**0** (0.04)	1 (0.06)	3 (0.10)	6 (0.15)	7 (0.23)	4 (0.12)	3 (0.10)
			$k=6$	4 (0.14)	**0** (0.06)	1 (0.07)	2 (0.11)	5 (0.17)	7 (0.26)	4 (0.15)	2 (0.13)
			$k=8$	6 (0.18)	1 (0.09)	**0** (0.06)	2 (0.10)	4 (0.15)	7 (0.26)	3 (0.15)	3 (0.13)
			$k=10$	6 (0.20)	2 (0.11)	**0** (0.06)	1 (0.08)	3 (0.14)	7 (0.25)	4 (0.14)	2 (0.12)
		$\rho=0.4$	$k=2$	1 (0.02)	**0** (0.02)	2 (0.04)	5 (0.06)	6 (0.09)	7 (0.13)	2 (0.04)	4 (0.05)
			$k=4$	2 (0.06)	**0** (0.03)	1 (0.05)	3 (0.09)	6 (0.14)	7 (0.21)	3 (0.10)	4 (0.11)
			$k=6$	3 (0.13)	**0** (0.05)	1 (0.06)	2 (0.10)	5 (0.15)	7 (0.25)	3 (0.14)	3 (0.13)
			$k=8$	6 (0.17)	1 (0.08)	**0** (0.05)	2 (0.09)	3 (0.14)	7 (0.24)	3 (0.15)	3 (0.14)
			$k=10$	6 (0.19)	2 (0.10)	**0** (0.05)	1 (0.08)	3 (0.12)	7 (0.23)	4 (0.15)	4 (0.14)
	$n=512$	$\rho=-0.4$	$k=2$	2 (0.07)	**0** (0.04)	1 (0.06)	3 (0.09)	5 (0.10)	6 (0.14)	7 (0.17)	3 (0.08)
			$k=6$	5 (0.16)	2 (0.07)	**0** (0.05)	1 (0.07)	3 (0.10)	6 (0.18)	7 (0.23)	4 (0.12)
			$k=10$	6 (0.21)	4 (0.12)	1 (0.05)	**0** (0.05)	2 (0.07)	5 (0.17)	6 (0.21)	3 (0.11)
		$\rho=0.0$	$k=2$	2 (0.06)	**0** (0.03)	1 (0.05)	4 (0.08)	5 (0.10)	7 (0.14)	6 (0.13)	3 (0.07)
			$k=6$	5 (0.17)	1 (0.07)	**0** (0.05)	1 (0.07)	3 (0.11)	6 (0.19)	7 (0.21)	4 (0.12)
			$k=10$	7 (0.24)	4 (0.14)	1 (0.06)	**0** (0.05)	2 (0.09)	5 (0.18)	5 (0.19)	3 (0.12)
		$\rho=0.4$	$k=2$	2 (0.05)	**0** (0.03)	1 (0.04)	3 (0.08)	5 (0.10)	7 (0.14)	5 (0.10)	3 (0.07)
			$k=6$	6 (0.17)	1 (0.06)	**0** (0.05)	1 (0.07)	3 (0.11)	7 (0.20)	5 (0.16)	3 (0.11)
			$k=10$	7 (0.24)	4 (0.13)	**0** (0.05)	**0** (0.05)	2 (0.08)	6 (0.18)	5 (0.15)	3 (0.09)

dominated by a solution obtained by an algorithm. The EAF differences show regions where one algorithm outperforms the other. The magnitude of the difference in favor of one algorithm is plotted in a colored scale. One can observe that MOW-P performs significantly better than MOEA/D, and often finds solutions not likely to be found by MOEA/D, especially on the lexicographically optimal regions of the Pareto front — the "extremes". This observation increases with the ruggedness (k). On relatively smooth landscapes, MOEA/D appears to perform slightly better in the central area of the Pareto front. This is only observed when $n = 512$, yet as ruggedness is determined by the rate between k and n, this instance is the smoothest among the ones considered here.

The comments so far are for the largest budget of 10^7 evaluations. Fig. 2 now reports the anytime performance of the three most efficient versions of MOW-P and the two most efficient versions of MOEA/D observed above. A subset of instances was selected due to space restrictions, but the trends are similar on other instances. Note the log-scale on both axes. A lower hypervolume relative deviation means a better performance. Except for particularly small computational budgets, MOW-P consistently outperforms MOEA/D. For two objectives, the superiority of MOW-P occurs for a smaller budget than for three objectives. The increase in objectives also impacts the gap between the two MOEA/D variants. Note that the relative efficiency of MOW-P variants may change with the budget: lower λ-values are more recommended for small budgets.

Overall, MOW-P outperforms MOEA/D on this benchmark, provided that an appropriate setting of λ is chosen, with the exception of very low computational budgets. While affected

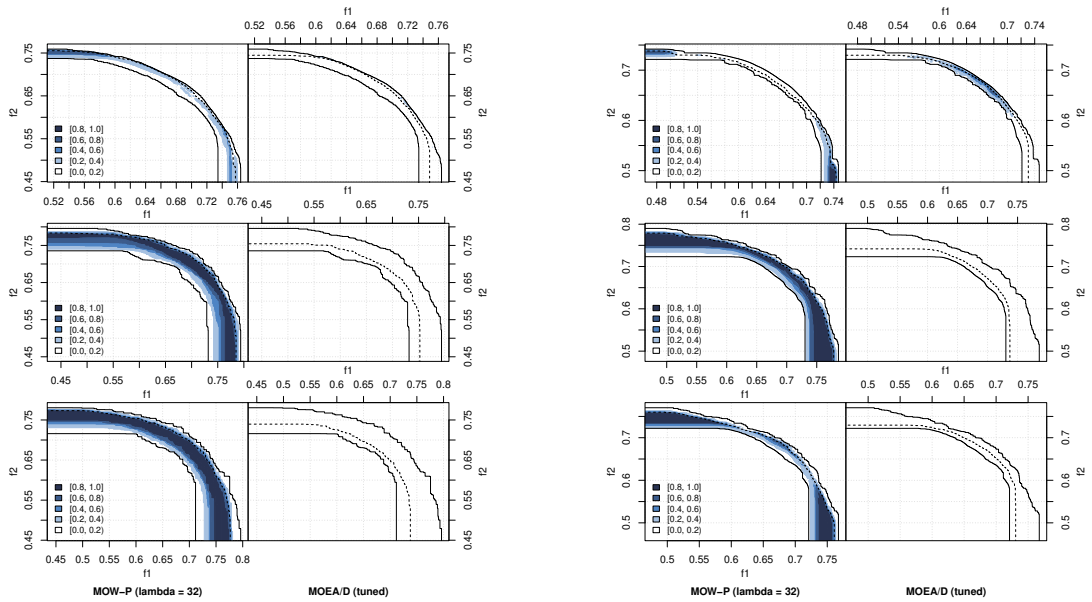

Fig. 1. Empirical attainment function differences between MOW-P and MOEA/D for instances with $m = 2$, $\rho = 0$, $n = 128$ (left) and $n = 512$ (right), $k = 2$ (top), $k = 6$ (middle) and $k = 10$ (bottom).

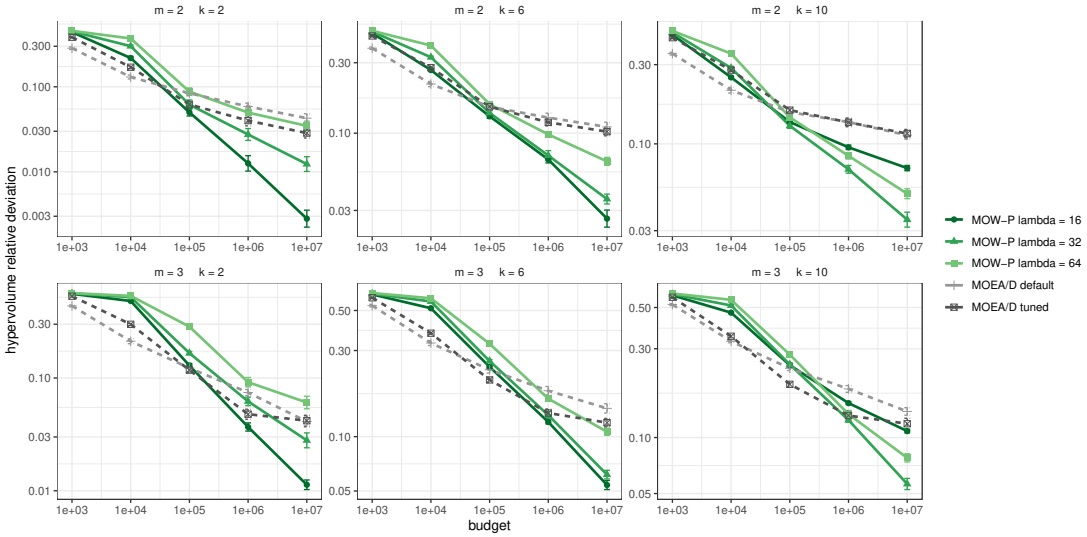

Fig. 2. Impact of the neighborhood sample size λ ($n = 128, \rho = 0$).

by n and k, the appropriate λ-value is rather stable and $\lambda = 32$ always outperforms any MOEA/D variant. MOW-P provides a good coverage of the Pareto front, especially at the extremes. A notable observation is that the MOW-P version that achieves such results turns out to be particularly straightforward.

B. Investigating Other MOW-P Settings

After λ, we investigate other settings of MOW-P below.

1) Population Size (μ): We recall that μ corresponds to the number of weight vectors of the decomposition, and therefore to the number of walks performed during MOW-P's execution. Fig. 3 reports the anytime behavior of MOW-P for different population sizes when $\lambda = 32$. For two objectives, the lower population size generally leads to a better hypervolume for different budgets. The only exception is for $k = 2$ and a budget of 10^7 evaluations, where there is no significant difference. Note that $k = 2$ means a low variable interaction and leads to smooth landscapes that are easier to tackle with local search. For three objectives, the lowest μ-value is consistently better, whatever the budget. The gap in performance induced by the population size is larger for 3 objectives and tend to increase with k. We note that this difference may be due to a slower convergence with this higher number of objectives. The bottom line of this results is that too many sub-problems does not lead to the best possible results for MOW-P: To ensure good results under a given total budget, not too many walks should be performed in parallel but still in sufficient numbers for providing a good coverage of the Pareto front.

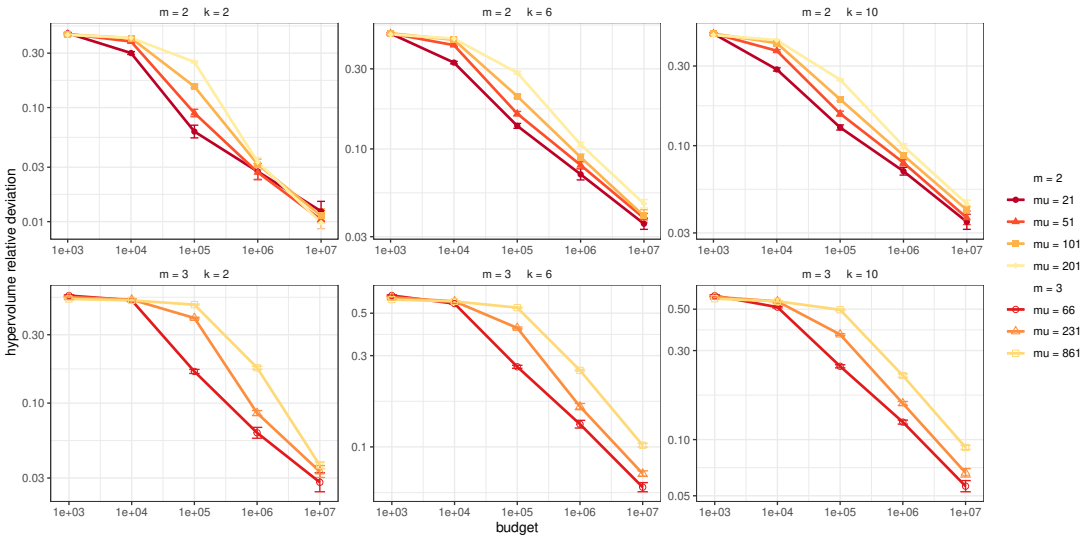

Fig. 3. MOW-P – Impact of the population size μ for $\lambda = 32$ ($n = 128$, $\rho = 0$).

2) Cooperation among Sub-problems (t): Although we did *not* consider cooperation between sub-problems in the baseline version of MOW-P introduced in Section III, we here study the possible benefits of cooperative MOW-P's walks. To do so, we follow a similar approach to MOEA/D [3]: For each sub-problem $i \in \{1, \ldots, \mu\}$, we define a set of *cooperating sub-problems* $B(i)$ with its t closest weight vectors. However, unlike MOEA/D, we must also define which of the λ evaluated neighbors (N_λ) is to be selected at a given iteration. We follow the strategy of selecting the neighbor with the smallest (normalized) deviation among cooperating sub-problems:

$$\underset{y \in N_\lambda,\, j \in B(i) \cup \{i\}}{\arg\min} \frac{g(x^j \mid w^j) - g(y \mid w^j)}{g(x^j \mid w^j)} \tag{2}$$

The selected solution replaces the current solution of the corresponding sub-problem j. As in the baseline MOW-P, the selected neighbor may or may not result in an improvement.

Fig. 4 reports the anytime behavior of MOW-P with different numbers of cooperating sub-problems t. While cooperation does help MOW-P to achieve better results for smaller budgets, it usually performs significantly better without any cooperation between the walks ($t = 0$). Although the results achieved for the three MOW-P versions with cooperation are particularly close, the gap in efficiency without communication significantly increases with the budget. This observation is similar regardless of the number of objectives or the k-value, except for the instance with $m = 3$ and $k = 2$.

VI. Conclusion and Further Considerations

In this paper, we proposed MOW-P, a multi-objective algorithm that demonstrated compelling results on difficult problems despite its high simplicity. Its principles are based on SW, a straightforward single-objective local search that selects the best among λ random neighbors. The main difference is that MOW-P applies SW to a set of scalar sub-problems obtained by decomposition of the target multi-objective problem.

The uncommon characteristics of MOW-P compared to existing approaches stem from its simplicity. For example, it operates on a single solution at each step, the selection process never relies on the archive nor the fitness of the current solution, and it never explicitly aims at converging towards local optima. A key feature that differentiates MOW-P from the single-objective setting is its (external) archive. However, archiving has no influence on the search process, yet it ensures that a good-quality set of mutually non-dominated solutions is returned. We highlighted that MOW-P could use multi-archiving, in which each walk manages its own archive of solutions that are all merged only once, at the end of the search process. This allows MOW-P to handle multiple, smaller sub-archives, while still managing to cover a reasonable portion of the Pareto front. On top of that, this gives the potential of solving the sub-problems in parallel, each one independently of the others. This is illustrated in Fig. 5. We argue that multi-archiving leads to the same final archive as the standard single global archive in a significantly reduced computational time.

The simplicity of MOW-P provides a great advantage, given that some issues that are difficult to handle with most algorithms can be managed with minor tweaks. In particular, the following refinements could be considered:

- an adaptive setting of λ, as proposed, e.g., in [9];
- an anytime version whose convergence would speed up by starting with a small number of sub-problems that would gradually increase during the search process;
- a preference-based version in which the decision maker would provide their preferences by means of a single weight vector that could be updated at runtime;
- hybridizations with other types of algorithms, even just executing a simple local search following MOW-P in order to ensure that the final solutions are local optima;
- a parallel version in which each thread performs a walk, which could be easily implemented with no communication, apart from filtering once the final archive;

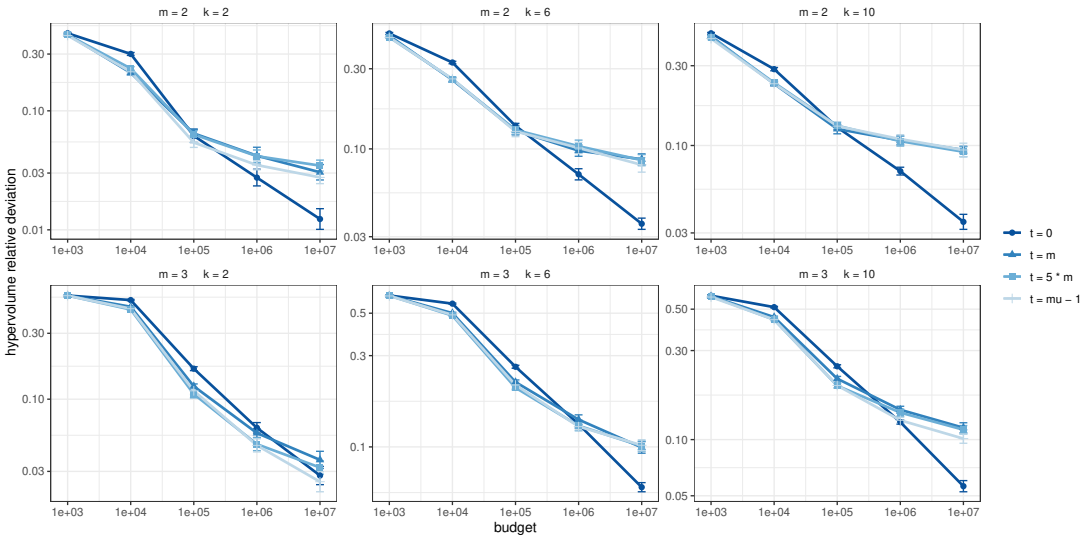

Fig. 4. MOW-P – Impact of the number of cooperating sub-problems t for $\lambda = 32$ ($n = 128$, $\rho = 0$).

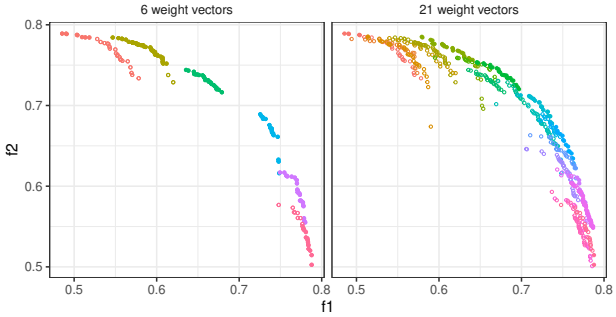

Fig. 5. Illustration of the multiple archiving process for $\mu = 6$ (left) and $\mu = 21$ (right) sub-problems ($n = 128$, $k = 6$, $\rho = 0$). Each sub-archive appears in a different color.

- covering a wider range of problems, including various solution representations and neighborhood structures;
- comparison against other classes of multi-objective algorithms, including multi-objective local search.

References

[1] S. Tari, M. Basseur, and A. Goëffon, "Sampled walk and binary fitness landscapes exploration," in *Artificial Evolution*. Cham: Springer International Publishing, 2018, pp. 47–57.

[2] S. Tari, M. Basseur, and A. Goëffon, "Partial neighborhood local searches," *International Transactions in Operational Research*, vol. 29, no. 5, pp. 2761–2788, 2022.

[3] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[4] J. D. Knowles, "Local-search and hybrid evolutionary algorithms for Pareto optimization," Ph.D. dissertation, University of Reading, 2002.

[5] Z. Liang, M. Li, and P. K. Lehre, "Non-elitist evolutionary multi-objective optimisation: Proof-of-principle results," in *Genetic and Evolutionary Computation Conference (GECCO 2023)*. Lisbon, Portugal: ACM Press, 2023, p. 383–386.

[6] R. Tanabe and H. Ishibuchi, "Non-elitist evolutionary multi-objective optimizers revisited," in *Genetic and Evolutionary Computation Conference (GECCO 2019)*. Prague, Czech Republic: ACM Press, 2019, pp. 612—619.

[7] B. Derbel, A. Liefooghe, Q. Zhang, H. E. Aguirre, and K. Tanaka, "Multi-objective local search based on decomposition," in *Parallel Problem Solving from Nature (PPSN XIV)*, ser. Lecture Notes in Computer Science, vol. 9921. Edinburgh, UK: Springer, 2016, pp. 431–441.

[8] J. Shi, Q. Zhang, B. Derbel, A. Liefooghe, and S. Verel, "Using parallel strategies to speed up Pareto local search," in *Simulated Evolution and Learning (SEAL 2017)*, ser. Lecture Notes in Computer Science, vol. 10593. Shenzhen, China: Springer, 2017, pp. 62–74.

[9] B. Neveu, G. Trombettoni, and F. Glover, "Id walk: A candidate list strategy with a simple diversification device," in *Principles and Practice of Constraint Programming (CP 2004)*. Springer, 2004, pp. 423–437.

[10] S. Tari, M. Basseur, and A. Goëffon, "On the use of $(1, \lambda)$-evolution strategy as efficient local search mechanism for discrete optimization: a behavioral analysis," *Natural Computing*, vol. 20, pp. 345–361, 2021.

[11] J. E. Fieldsend, "Data structures for non-dominated sets: implementations and empirical assessment of two decades of advances," in *Genetic and Evolutionary Computation Conference (GECCO 2020)*. Cancún, Mexico: ACM Press, 2020, pp. 489–497.

[12] J. D. Knowles and D. Corne, "Bounded Pareto archiving: Theory and practice," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems. Berlin, Germany: Springer, 2004, vol. 535, pp. 39–64.

[13] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens, "On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives," *European Journal of Operational Research*, vol. 227, no. 2, pp. 331–342, 2013.

[14] S. A. Kauffman, *The Origins of Order*. Oxford University Press, 1993.

[15] A. Liefooghe, F. Daolio, B. Derbel, S. Verel, H. E. Aguirre, and K. Tanaka, "Landscape-aware performance prediction for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 6, pp. 1063–1077, 2020.

[16] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[17] J. Dréo, A. Liefooghe, S. Verel, M. Schoenauer, J.-J. Merelo, A. Quemy, B. Bouvier, and J. Gmys, "Paradiseo: from a modular framework for evolutionary computation to the automated design of metaheuristics," in *Genetic and Evolutionary Computation Conference (GECCO 2020)*. Lille, France: ACM Press, 2021.

[18] M. López-Ibáñez, L. Paquete, and T. Stützle, "Exploratory analysis of stochastic local search algorithms in biobjective optimization," in *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, Germany, 2010, pp. 209–222.

[19] H. Wickham, *ggplot2: elegant graphics for data analysis*. Springer, 2009.