An experimental evaluation of the Algebraic Differential Evolution algorithm on the Single Row Facility Layout Problem

Gabriele Di Bari
University of Perugia
gabriele.dibari@unifi.it

Marco Baioletti
University of Perugia
marco.baioletti@unipg.it

Valentino Santucci
University for Foreigners of Perugia
valentino.santucci@unistrapg.it

ABSTRACT

The Algebraic Differential Evolution for Permutations (ADEP) has been recently proposed as an effective evolutionary algorithm for permutation-based optimization problems. ADEP is built upon a framework that exploits the rich algebraic structure of the permutations search space. In this paper we further explore the abilities of ADEP by presenting an implementation for the Single Row Facility Layout Problem (SRFLP): a permutation problem with interesting real-world applications ranging from designing the layouts of machines in certain manufacturing systems to optimally arranging rooms in hospitals. An experimental investigation was conducted on a set of commonly adopted benchmarks and different settings ADEP were compared among them and with respect to the other methods in the literature. Interestingly, the experimental results confirm the validity of ADEP by showing its competitiveness with respect to the state-of-the-art results for the SRFLP.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; **Discrete space search**; **Randomized search**.

KEYWORDS

Differential Evolution, Combinatorial optimization, Algebraic Differential Evolution

ACM Reference Format:
Gabriele Di Bari, Marco Baioletti, and Valentino Santucci. 2020. An experimental evaluation of the Algebraic Differential Evolution algorithm on the Single Row Facility Layout Problem. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3377929.3398130

1 INTRODUCTION

Differential Evolution (DE) is a powerful meta-heuristic algorithm originally proposed for continuous optimization problems [32]. Nevertheless, there are also many applications of DE to combinatorial problems. In most of them, DE was adapted to work into discrete spaces by means of transformation techniques that decode a numerical vector (genotype) into a corresponding discrete solution (phenotype) during the evaluation step. See for instance [10, 13, 21].

However, a single discrete solution could be represented by a potentially infinite number of continuous individuals, hence the continuous-to-discrete transformation schemes introduce a one-to-many mapping from the phenotypic to the genotypic space. Thereby, large plateaus are very likely to be introduced in the search landscape.

In order to overcome these issues, we have recently proposed a discrete realization of DE which is based on an algebraic framework for combinatorial optimization based on strong mathematical foundations. The proposed algebraic DE directly works on discrete search spaces by simulating the behavior of its numerical counterpart [32]. In particular, the discrete differential mutation operator is directly defined on the solution space of the discrete problem at hand.

Actually, the algebraic framework, thus the algebraic DE, tackle all the optimization problems whose search space can be represented by means of a *finitely generated group*. This requirement is met by most of the combinatorial search spaces commonly considered like, for instance, the space of binary strings [26] and that of permutations [6].

In the case of permutations, the algebraic differential mutation has been firstly implemented using a generating set based on the adjacent swap moves [27, 29] and the corresponding Algebraic Differential Evolution for Permutations (ADEP) has been applied to the flowshop scheduling problems [29, 30] and to the linear ordering problem [4, 28] where, respectively, state-of-the-art and competitive results have been obtained. In [5, 7, 8] ADEP has been extended by introducing other generating sets based on exchange, insertion and reversal moves. These new variants of ADEP have been applied to the linear ordering problem with cumulative costs [7] and to the traveling salesman problem [8].

In this paper we further explore the abilities of ADEP as a solver for permutation-based optimization problems by tackling the Single Row Facility Layout Problem (SRFLP). The SRFLP is an NP-hard problem widely studied in the literature [3, 31] where the goal is to find a linear arrangement, i.e., a permutation, of a set of facilities in order to minimize the sum of their pairwise distances in the linear order. The SRFLP has interesting real-world applications ranging from designing the layouts of machines in certain manufacturing systems [17] to arranging rooms in hospitals [31].

An experimental investigation was conducted on a set of commonly adopted benchmarks for the SRFLP. Different settings of ADEP were experimentally compared and the results obtained by the best setting were also compared with the state-of-the-art results from the literature.

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07…$15.00
https://doi.org/10.1145/3377929.3398130

The rest of the paper is organized as follows. Section 2 describes the SRFLP by also reporting the most recent works in the literature. Sections 3 recalls the working scheme of the algebraic DE, while Section 4 describes the implementation of ADEP used in this work. Experimental findings are described in Section 5, while Section 6 concludes the paper by also depicting future lines of research.

2 THE SINGLE ROW FACILITY LAYOUT PROBLEM

2.1 Problem definition

The Single Row Facility Layout Problem (SRFLP), also known as one-dimensional space allocation problem [31], aims to find an optimal linear arrangement of a set of rectangular facilities of equal height but different lengths. In particular, the objective is to minimize a weighted sum of the distances between the centers of all pairs of facilities in the linear arrangement.

Formally, a SRFLP instance is given as a set of n facilities $F = \{1, 2, \dots, n\}$ and any facility $i \in F$ has length $l_i > 0$. Moreover, a transmission cost $c_{i,j} \geq 0$ is given for any pair of facilities $i, j \in F$ with $i \neq j$. Also note that the transmission costs are symmetric, i.e., $c_{i,j} = c_{j,i}$ for all $i, j \in F$. An SRFLP solution is thus a permutation $\pi = \langle \pi(1), \pi(2), \dots, \pi(n) \rangle$ of the facilities in F, whose cost $C(\pi)$ is

$$C(\pi) = \sum_{q=1}^{n-1} \sum_{r=q+1}^{n} c_{\pi(q),\pi(r)} d\left(\pi(q), \pi(r)\right), \tag{1}$$

where $d\left(\pi(q), \pi(r)\right)$ is the distance between the centers of the facilities $\pi(q)$ and $\pi(r)$, which is calculated as

$$d\left(\pi(q), \pi(r)\right) = \frac{l_{\pi(q)}}{2} + \frac{l_{\pi(r)}}{2} + \sum_{s=q+1}^{r-1} l_{\pi(s)}. \tag{2}$$

Therefore, the goal is to find a permutation π^* of the facilities in F that minimizes the objective function given in equation (1). Formally, π^* is a permutation from the set $\mathcal{S}_n$ of all the permutations of n items, such that

$$\pi^* = \operatorname{argmin}_{\pi \in \mathcal{S}_n} C(\pi). \tag{3}$$

The SRFLP has interesting real-world applications such as: arranging rooms in hospitals or offices in buildings [31], designing layouts of machines in flexible manufacturing systems [17], assigning information to disk cylinders in computer storage [18], or warehouse design [23].

2.2 Related Work

SRFLP has been proven to be an NP-Hard problem in [11]. For this reason, exact approaches – like, for instance, branch and bound and integer linear programming [16] – allow to solve only relatively small instances of the problem.

Therefore, many (meta)heuristic methods for the SFRLP have been proposed in the literature. Two genetic algorithms have been presented in [15] and [1]. A particle swarm optimization algorithm (PSO) for the SRFLP has been introduced in [25], while another swarm intelligence scheme equipped with a local exchange heuristic has been devised in [35]. Local search approaches are the tabu search and variable neighborhood search proposed in, respectively, [33] and [22]. Other interesting meta-heuristics are GENALGO, a genetic algorithm introduced in [18], and SS-P2, a scatter search approach reported in [19]. Finally, a greedy adaptive search for the SRFLP has been proposed in [14, 24], which also use a path relinking operator in the context of the SRFLP.

One of the most recent and prominent algorithms for the SRFLP is the population-based improvement heuristic with local search [3], which is later referred to as PBIH. This algorithm is among the state-of-the-art methods together with the local search based approaches, i.e.: the variable neighborhood search [22], the tabu search [33], and the location exchange heuristic [35].

3 ALGEBRAIC DIFFERENTIAL EVOLUTION

3.1 Abstract definition

As described in [7, 29, 34], the Algebraic Differential Evolution (ADE) simulates the behaviour of the classical DE in a discrete search space representable by a *finitely generated group*. The population $\{x_1, \dots, x_N\}$ of N candidate solutions is iteratively evolved by the three operators of differential mutation, crossover and selection. As widely recognized in the literature [32], the key operator of DE is the differential mutation (DM), therefore hereinafter we focus on DM that, in its most common variant, called rand/1, generates a mutant v according to

$$v \leftarrow x_{r_0} \oplus F \odot (x_{r_1} \ominus x_{r_2}), \tag{4}$$

where $x_{r_0}, x_{r_1}, x_{r_2}$ are three randomly selected population individuals, and $F \in (0, 1]$ is the *scale factor parameter*.

In numerical DE, the operators $\oplus, \ominus, \odot$ are the usual vector operations of $\mathbb{R}^n$, while, in ADE, their definitions are formally derived using the underlying algebraic structure of the search space [9].

The triple $(X, \circ, G)$ is a finitely generated group representing a combinatorial search space when:

- X is the discrete set of solutions in the space;
- $\circ$ is a binary operation on X satisfying the group properties, i.e., closure, associativity, existence of a neutral element (or identity e), and invertibility (x^{-1}); and
- $G \subseteq X$ is a finite generating set of the group, i.e., any $x \in X$ has a (not necessarily unique) minimal-length decomposition $\langle g_1, \dots, g_l \rangle$, with $g_i \in G$ for all $i \in \{1, \dots, l\}$, and whose evaluation is x, i.e., $x = g_1 \circ \cdots \circ g_l$.

For the sake of clarity, the length of a minimal decomposition of x is denoted by $|x|$.

Using $(X, \circ, G)$ we can provide the formal definitions of the operators $\oplus, \ominus, \odot$. Let $x, y \in X$ and $\langle g_1, \dots, g_k, \dots, g_{|x|} \rangle$ be a minimal decomposition of x, then

$$x \oplus y := x \circ y, \tag{5}$$

$$x \ominus y := y^{-1} \circ x, \tag{6}$$

$$F \odot x := g_1 \circ \cdots \circ g_k, \text{ with } k = \lceil F \cdot |x| \rceil \text{ and } F \in [0, 1]. \tag{7}$$

The algebraic structure on the search space naturally defines neighborhood relations among the solutions. Indeed, it induces a colored digraph whose vertices are the solutions in X and two generic solutions $x, y \in X$ are linked by an arc with color $g \in G$ if and only if $y = x \circ g$. Therefore, a simple one-step move can be directly encoded by a generator, while a composite move can be synthesized as the evaluation of a sequence of generators (a path on the graph). In analogy with $\mathbb{R}^n$, the elements of X can

be interpreted both as solutions (vertices on the graph) and as displacements between solutions (colored paths on the graph). As detailed in [29], this provides a rational interpretation to the discrete DM of definition (4). The key idea is that the difference $x \ominus y$ is the evaluation of the generators in a shortest path from y to x.

Clearly, $\oplus$ and $\ominus$ do not depend on the generating set, thus they are uniquely defined. Conversely, $\odot$ relies on the chosen generating set and, also fixing it, a minimal decomposition is not unique in general. Therefore, $\odot$ is implemented as a stochastic operator, thus requiring a randomized decomposition algorithm for the finitely generated group at hand.

3.2 ADE for Permutations

The Algebraic Differential Evolution for Permutations (ADEP) [29] is an implementation of ADE for the search space of permutations. In fact, the permutations of $[n] = \{1, \ldots, n\}$ can be composed by the operator $\circ$ and form the so-called *Symmetric group* $\mathcal{S}_n$. Different choices for the generating set are possible and, more interestingly, each generating set can be interpreted as a class of moves between permutations. In this work we consider the following three generating sets:

- ASW_n, which is based on *adjacent swap moves* and it is formally defined as
$$ASW_n = \{\sigma_i : 1 \leq i < n\}, \tag{8}$$
where σ_i is the identity permutation with the items i and $i+1$ exchanged. For example, let $\pi = \langle 3, 5, 2, 4, 1\rangle$, then $\pi \circ \sigma_3 = \langle 3, 5, 2, 4, 1\rangle \circ \langle 1, 2, 4, 3, 5\rangle = \langle 3, 5, 4, 2, 1\rangle$.
- EXC_n, which is based on *exchange moves* and it is formally defined as
$$EXC_n = \{\epsilon_{ij} : 1 \leq i < j \leq n\}, \tag{9}$$
where ϵ_{ij} is the identity permutation with the items i and j exchanged. For example, let $\pi = \langle 3, 5, 2, 4, 1\rangle$, then $\pi \circ \epsilon_{2,5} = \langle 3, 5, 2, 4, 1\rangle \circ \langle 1, 5, 3, 4, 2\rangle = \langle 3, 1, 2, 4, 5\rangle$.
- INS_n, which is based on *insertion moves* and it is formally defined as
$$INS_n = \{\iota_{ij} : 1 \leq i, j \leq n\}, \tag{10}$$
where ι_{ij} is the identity permutation where the item i is shifted to position j. For example, let $\pi = \langle 3, 5, 2, 4, 1\rangle$, thus $\pi \circ \iota_{3,5} = \langle 3, 5, 2, 4, 1\rangle \circ \langle 1, 2, 4, 5, 3\rangle = \langle 3, 5, 4, 1, 2\rangle$.

A minimal decomposition of a permutation $\pi \in \mathcal{S}_n$ can be obtained by ordering the items in π using a sorting algorithm that only consider moves from the chosen set, i.e., adjacent swaps for *ASW*, generic exchanges for *EXC*, and insertions for *INS*.

Optimal randomized decomposers for *ASW*, *EXC*, and *INS* have been implemented by means of generalized and randomized variants of, respectively, the bubble-sort, selection-sort, and insertion-sort algorithms [5, 7, 29]. They have been called *RandBS*, *RandSS*, *RandIS* and exploit different combinatorial properties of the permutations.

Since (the ordered permutation) e is the only permutation with 0 inversions [29], *RandBS* iteratively decreases the number of inversions by swapping two randomly selected adjacent items π_i and π_{i+1} which form an inversion of π, i.e., $\pi_i > \pi_{i+1}$. The time complexity of *RandBS* is $\Theta(n^2)$.

RandSS exploits the fact that e is the only permutation with n cycles (of length 1) in its cycles representation, thus it iteratively increases the number of cycles of π by randomly breaking a randomly selected cycle. In fact, it turns out that any cycle can be split in two shorter cycles by exchanging any pair of items in the original cycle. The amortized time complexity of *RandSS* is $\Theta(n)$.

RandIS considers that e is the only permutation with a (unique) longest increasing subsequence (LIS) of length n, thus it iteratively extends the LIS length of π by shifting a suitable item in a suitable position. The time complexity of *RandIS* is $\Theta(n^2)$.

For further implementation details, proofs of correctness and complexity we refer the interested reader to [5, 7, 29].

4 ADEP FOR THE SRFLP

In this work we present an ADEP implementation purposely defined for the SRFLP. Its pseudo-code is outlined in Algorithm 1.

Algorithm 1 ADEP scheme for the SRFLP

```
Randomly initialize the population {π_1, ..., π_N}
for gen ← 1 to MaxGen do
    for i ← 1 to N do
        v_i ← DifferentialMutation(i, {π_1, ..., π_N}, F);
        v_i ← Crossover(π_i, v_i);
        Evaluate f(v_i)
    end for
    Select the population for the next iteration;
    if restart criterion is satisfied then
        Reinitialize the population;
        Apply local search to the best individual π_best;
    end if
end for
Apply local search to the best individual π_best;
```

ADEP directly evolves a population $\{\pi_1, \ldots, \pi_N\}$ of N permutations. For every population individual π_i, the differential mutation operator builds a mutant v_i by using the algebraic operators introduced in Section 3. Two different mutation scheme have been considered: the explorative rand/1 and the exploitative best/1. They are formally defined in the following two equations:

$$v_i \leftarrow \pi_{r_0} \oplus F \odot (\pi_{r_1} \ominus \pi_{r_2}), \tag{11}$$

$$v_i \leftarrow \pi_{\text{best}} \oplus F \odot (\pi_{r_1} \ominus \pi_{r_2}), \tag{12}$$

where $\pi_{r_0}, \pi_{r_1}.\pi_{r_2}$ are three population individuals different from each other and with respect to π_i, while π_{best} is the best individual so far in the population. Moreover, $F \in [0, 1]$ is the scale factor parameter of ADEP which, in this work, is self-adapted during the evolution by means of the popular jDE scheme [12].

The mutant v_i undergoes crossover together with the original population individual π_i in order to produce the trial solution v_i. Two selection operators are used in ADEP: *1to1* and *Crowd*. The *1to1* operator replaces π_i in the ADEP population with v_i if the latter is fitter than the former, while, in the *Crowd* operator, v_i competes with the closest individual in the current ADEP population. Clearly, the fitness f considered in this work is the SRFLP objective function depicted in equation (1).

After some preliminary experiments, five alternative crossover operators for the permutations space have been selected: the three order-based operators O, O2 and OB, the partially mapped crossover PMX and the maximal preservative MP. These are among the most used crossovers for permutation problems and their descriptions can be found, for instance, in [20].

Since the search space is finite, ADEP executions may converge quicker than its numerical counterpart, therefore a soft restart procedure was introduced. The restart is executed when the best individual so far π_{best} was not improved during the last T generations. T is set to 150 after some preliminary experiments. The soft restart consists in: (i) optionally performing a local search starting from π_{best}, and (ii) randomly reinitialize all the other individuals. In this way, the best individual is always maintained in the population, thus guaranteeing an elitist behaviour to ADEP.

The local search is performed by iteratively replacing the incumbent solution with its best neighbor if and only if the latter is fitter than the former (*best style* local search). The neighborhood based on insertion moves has been adopted since it is considered to be the most effective neighborhood for the SRFLP [24].

5 EXPERIMENTS

In order to evaluate the performance of ADEP in the SRFLP, we have conducted an experimental analysis using the *dept* instances, introduced in [2], and the *sko* instances based on the quadratic assignment problem. Both groups of instances are commonly used in the literature as benchmark [3].

The experiments are organized as follows. We have initially performed a tuning phase in order to find the best combination of the ADEP parameters. We have then analysed the impact of equipping the best ADEP configurations with a local search. We have compared the results of ADEP with the state-of-the-art results from the literature. Finally, we have also studied the convergence of the algorithm.

5.1 Experimental tuning of ADEP

The ADEP algorithm has six parameters to be set, the generating set $\mathcal{G}$, the mutation operator $\mathcal{M}$, the crossover operator $\mathcal{C}$, the type of selection $\mathcal{S}$, the population size N, and the number of generations $MaxGen$. We decided to choose the values of the last two parameters as $N = 100$ and $MaxGen = 10000$, while for the other ones we decided to perform a fully factorial tuning phase on the *sko* instances.

The parameters to be set have the following possible values:

- $\mathcal{G} \in \{ASW, EXC, INS\}$,
- $\mathcal{M} \in \{rand/1, best/1\}$,
- $\mathcal{C} \in \{O, O2, OB, MP, PMX\}$,
- $\mathcal{S} \in \{1to1, Crowd\}$,

hence the total number of configurations is 60.

For each configuration, the average fitness values over 10 runs per instance have been recorded. We have then computed the average rank for each setting and in Table 1, the top 10 settings are reported.

The best configuration found is $C_1 = (EXC, rand/1, OB, Crowd)$ whose average rank value is 2.69.

Configuration	Average Rank
(EXC, rand/1, OB, Crowd)	2.69
(EXC, best/1, OB, Crowd)	3.19
(EXC, rand/1, O, Crowd)	3.31
(INS, best/1, PMX, 1to1)	3.75
(EXC, best/1, MP, Crowd)	3.94
(EXC, rand/1, O, 1to1)	10.44
(ASW, rand/1, O2, 1to1)	11.50
(INS, rand/1, O, 1to1)	11.75
(INS, rand/1, MP, 1to1)	12.43
(EXC, rand/1, O2, 1to1)	13.06

Table 1: Configurations ranks

Because of these results, we decided to perform a deeper experimental study between the best configuration C_1 and the second best $C_2 = (EXC, best/1, OB, Crowd)$, which only differs from C_1 for the value of $\mathcal{M}$. Moreover, we have also analyzed the effect of the local search, comparing these configurations with and without the local search.

5.2 Analysis of the local search

The aim of this experimental comparison is to check if the use of the local search (performed when the population is restarted and at the end of the evolution) gives significant improvements on the algorithm performances. Therefore, the two best ADEP configurations

$$C_1 = (EXC, rand/1, OB, Crowd)$$

and

$$C_2 = (EXC, best/1, OB, Crowd)$$

were compared both with and without considering local search (LS). For this test, ADEP has been executed 10 times for each configuration. The results are provided in Table 2, where we reported the minimum and the average of the fitness function. The results in bold are the best objective functions obtained on a specific instance.

The configuration C_2 obtained the best results in the 30% of the *dept* instances, and in the 10% of the *sko* instances. The configuration C_1 never reached the best results but in the average case it is better than C_2 in the 45% of *dept* instances and 50% of *sko* ones. The configuration $C2$ + LS never reached the best results in terms of minimum and average fitness values if compared to the other configurations. It is likely that ADEP with $\mathcal{M} = best/1$ and the local search easily get stuck in a local minimum.

The configurations C_1 + LS produced the best results in terms of the minimum and the average fitness values. It reached the best solution in the 70% of the *dept* instances and the 90% on the *sko* instances. Also, in average it is the best configuration, precisely, in the 55% and 50% of the *dept* and *sko* instances.

We also performed three Wilcoxon signed-rank tests, among C_1+LS and the three other configurations C_2, C_1, and C_2+LS, whose p-values are, respectively, 0.56×10^{-5}, 1.32×10^{-5}, 3.22×10^{-5}. This means that the configuration C_1 + LS is significantly the best ADEP setting.

Since this results, we decided to perform an empirical comparison between ADEP with the configuration $(EXC, rand/1, OB, Crowd)$+ LS and the state-of-the-art algorithm PBIH [3].

5.3 Comparison with the state of the art

We compared the ADEP performances with the PBIH ones in terms of best results obtained by both algorithms in each instance of the group *dept* and *sko* and computed the relative percentage deviation of ADEP with respect to PBIH.

The results, presented in Table 3, show that our algorithm is competitive with PBIH. In fact, ADEP reached the same minima found by PBIH in the 55% of the case and its worst result was only 0.003%.

Statistical test reveals that there are no significant difference between the two algorithms on the *dept* instances, while there are very small differences on the *sko* ones.

Anyway, as a conclusion, it is possible to state that ADEP, a general purpose solver for permutation problems, can achieve results which are extremely close to the state-of-the-art SRFLP results obtained by algorithms which have been purposely designed for the problem at hand.

5.4 Convergence analysis

We have also studied the impact of using or not the local search in terms of the convergence curve on the best ADEP configuration, i.e. $C_1 = (EXC, rand/1, OB, Crowd)$.

In Figure 1a we present the most representative convergence graph obtained on the *dept* instances. It is possible to notice that C_1 with local search has a smoother descent with respect to the plain C_1. In fact, the latter search gets stuck in a local minimum after a few generations; it gets out from there at generation 5000 and then falls in another local minimum at generation 7000. Conversely, C_1+LS slowly converges to its final result at generation 7000, which is lower than the result obtained by C_1.

We can observe an analogous situation in Figure 1b, where the most representative behaviour on the *sko* instances is reported. In fact, as in the *dept* instances, C_1 without local search quickly reaches the final fitness value (found after 2000 generations). On the other hand, C_1 with local search has a slower convergence: initially produces worse results than C_1, but it then reaches better and better results and converges after 8000 generations.

6 CONCLUSION AND FUTURE WORK

In this paper we have presented an Algebraic Differential Evolution (ADEP) algorithm for the Single Row Facility Layout Problem. ADEP is based on the recently introduced algebraic framework for discrete evolutionary algorithms [29, 34] and, in this work, it has been adapted and enhanced for the problem at hand by implementing different crossovers, algebraic mutations, selection operators, a local search method, and a restart procedure. ADEP parameters have been tuned and the best configuration has been compared with the state-of-the-art algorithm PBIH, showing that its results are comparable with its competitor.

As a future line of research, we would like to enrich ADEP with other generating sets for the algebraic differential mutation or with other algorithmic components, such as some mechanism to enhance the population diversity, in order to improve its effectiveness.

We are also interested in trying to use ADEP with other similar permutation based or discrete optimization problems, like the Double Row Layout Problem.

ACKNOWLEDGMENTS

The research described in this work has been partially supported by the research projects: "Università per Stranieri di Perugia – Finanziamento per Progetti di Ricerca di Ateneo — PRA 2020", "Università per Stranieri di Perugia – Artificial intelligence for education, social and human sciences", and Università degli Studi di Perugia – Fondi per la ricerca di base 2017 - 2019 "Algoritmi evolutivi per l'ottimizzazione combinatoria e per il machine learning".

REFERENCES

[1] Giuseppe Aiello, Giada La Scalia, and Mario Enea. 2012. A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding. Expert Systems with Applications 39, 12 (2012), 10352–10358.
[2] Miguel F. Anjos, Andrew Kennings, and Anthony Vannelli. 2005. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. Discrete Optimization 2, 2 (2005), 113 – 122. https://doi.org/10.1016/j.disopt.2005.03.001
[3] Soumen Atta and Priya Ranjan Sinha Mahapatra. 2019. Population-based improvement heuristic with local search for single-row facility layout problem. Sādhanā 44, 11 (2019), 222.
[4] M. Baioletti, A. Milani, and V. Santucci. 2015. Linear Ordering Optimization with a Combinatorial Differential Evolution. In 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2135–2140. https://doi.org/10.1109/SMC.2015.373
[5] Marco Baioletti, Alfredo Milani, and Valentino Santucci. 2016. An Extension of Algebraic Differential Evolution for the Linear Ordering Problem with Cumulative Costs. In Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings. 123–133. https://doi.org/10.1007/978-3-319-45823-6_12
[6] Marco Baioletti, Alfredo Milani, and Valentino Santucci. 2017. A New Precedence-Based Ant Colony Optimization for Permutation Problems. In Simulated Evolution and Learning. Springer International Publishing, Cham, 960–971. https://doi.org/10.1007/978-3-319-68759-9_79
[7] M. Baioletti, A. Milani, and V. Santucci. 2020. Variable neighborhood algebraic Differential Evolution: An application to the Linear Ordering Problem with Cumulative Costs. Information Sciences 507 (2020), 37–52.
[8] Marco Baioletti, Alfredo Milani, Valentino Santucci, and Umberto Bartoccini. 2019. An Experimental Comparison of Algebraic Differential Evolution Using Different Generating Sets. In Proc. of the Genetic and Evolutionary Computation Conference Companion (GECCO '19). 1527–1534. https://doi.org/10.1145/3319619.3326854
[9] Marco Baioletti and Valentino Santucci. 2017. Fitness Landscape Analysis of the Permutation Flowshop Scheduling Problem with Total Flow Time Criterion. In Computational Science and Its Applications – ICCSA 2017. Springer International Publishing, Cham, 705–716. https://doi.org/10.1007/978-3-319-62392-4_51
[10] J. C. Bean. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. ORSA Journal on Computing 6, 2 (1994), 154–160. https://doi.org/10.1287/ijoc.6.2.154
[11] M Beghin-Picavet and P Hansen. 1982. Deux problèmes d'affectation non linéaires. RAIRO-Operations Research 16, 3 (1982), 263–276.
[12] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. 2006. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation 10, 6 (2006), 646–657. https://doi.org/10.1109/TEVC.2006.872133
[13] Erbao Cao, Mingyong Lai, and Hongming Yang. 2014. Open vehicle routing problem with demand uncertainty and its robust strategies. Expert Systems with Applications 41, 7 (2014), 3569–3575.
[14] Gildásio Lecchi Cravo and André RS Amaral. 2019. A GRASP algorithm for solving large-scale single row facility layout problems. Computers & Operations Research 106 (2019), 49–61.
[15] Dilip Datta, André R.S. Amaral, and José Rui Figueira. 2011. Single row facility layout problem using a permutation-based genetic algorithm. European Journal of Operational Research 213, 2 (2011), 388 – 394. https://doi.org/10.1016/j.ejor.2011.03.034
[16] SS Heragu. 2016. Facility Layout. CRC Press.

Instance	(EXC, best/1, OB, Crowd)		(EXC, rand/1, OB, Crowd)		(EXC, bast/1, OB, Crowd)+LS		(EXC, rand/1, OB, Crowd)+LS	
	min	mean	min	mean	min	mean	min	mean
60dept_01	**1477834.00**	1479029.30	1477834.00	1479260.80	1477834.00	1478860.70	1477834.00	**1478627.05**
60dept_02	841872.00	850724.90	841776.00	845613.80	841776.00	846336.60	**841776.00**	**844938.60**
60dept_03	649833.50	652107.50	648744.50	**650232.70**	648918.50	650795.30	**648543.50**	650497.55
60dept_04	398468.00	401442.40	398468.00	**400002.00**	398406.00	401775.70	**398406.00**	400738.05
60dept_05	318855.00	324183.30	318805.00	**320782.75**	318805.00	322755.50	**318805.00**	321006.20
70dept_01	1528604.00	1534864.50	1528537.00	**1533738.20**	1528537.00	1531140.20	**1528537.00**	1533849.00
70dept_02	1441515.00	1446417.10	1441028.00	**1442696.60**	1441028.00	1443132.40	**1441028.00**	1443300.40
70dept_03	**1518993.50**	1529772.90	1518993.50	1524888.15	1519547.50	1526259.50	1518993.50	**1521985.80**
70dept_04	**968796.00**	976947.20	968796.00	**972951.80**	968796.00	976925.30	968796.00	974376.85
70dept_05	4218451.50	4234015.20	4218031.50	4224563.20	4218002.50	4220638.40	**4218002.50**	**4219338.15**
75dept_01	2401463.50	2411741.10	2393483.50	2409890.10	2393483.50	2402484.20	**2393456.50**	**2403818.20**
75dept_02	4321355.00	4339721.30	4321257.00	**4328458.10**	4321190.00	4330479.90	**4321190.00**	4329344.20
75dept_03	1250438.00	1258160.10	1249149.00	1256427.20	1250010.00	1254129.20	**1248537.00**	**1252692.00**
75dept_04	3947635.50	3954943.60	3941816.50	**3947604.55**	3942013.50	3948392.70	**3941816.50**	3950038.80
75dept_05	1794998.00	1804000.10	1791408.00	1799429.50	1791478.00	1799799.10	**1791408.00**	**1795702.70**
80dept_01	2069192.50	2073128.40	2069097.50	**2072720.25**	2070076.50	2071885.40	**2069097.50**	2074103.10
80dept_02	**1921136.00**	1940301.00	1921232.00	1924442.60	1921212.00	1924296.50	1921136.00	**1926003.40**
80dept_03	**3251368.00**	3265852.50	3256927.00	3269509.00	3251413.00	3260389.70	3251368.00	**3262633.55**
80dept_04	3747798.00	3762632.30	3749755.00	3767575.60	3746515.00	3757872.60	**3746515.00**	**3755197.65**
80dept_05	**1588885.00**	1604975.50	1588920.00	1601249.00	1589356.00	1599131.50	1588885.00	**1596659.60**
sko64_01	96977.00	97985.90	96891.00	97457.95	96910.00	97800.30	**96889.00**	**97414.10**
sko64_02	634929.50	644324.20	634346.50	**637714.80**	634544.50	642684.10	**634332.50**	640087.50
sko64_03	414405.50	418550.40	415422.50	417537.40	414349.50	417895.20	**414349.50**	**417322.45**
sko64_04	297875.00	300019.90	297388.00	299793.50	298078.00	301070.30	**297335.00**	**298808.30**
sko64_05	505059.50	509480.50	504170.50	**506388.20**	505213.50	508958.80	**501922.50**	506702.25
sko72_01	139379.00	141221.40	139180.00	**139491.10**	139556.00	142375.60	**139166.00**	140912.95
sko72_02	712370.00	718129.80	711998.00	716275.00	714665.00	718717.00	**711998.00**	**714659.60**
sko72_03	1056695.50	1062816.00	1055400.50	**1060545.50**	1056947.50	1063248.80	**1054581.50**	1061616.00
sko72_04	**920125.50**	930638.10	920973.50	929585.50	920973.50	929041.80	920312.50	**926666.80**
sko72_05	429185.50	432416.30	428866.50	430875.45	429162.50	431420.40	**428228.50**	**430322.50**
sko81_01	206692.00	207835.80	205585.00	**206519.60**	206557.00	207693.50	**205213.00**	207155.90
sko81_02	524684.50	527896.80	521505.50	**524805.40**	521434.50	526862.60	**521434.50**	525544.25
sko81_03	976269.00	981625.70	971011.00	977852.90	973484.00	980412.60	**970931.00**	**975839.20**
sko81_04	2034949.00	2052480.00	2032056.00	**2046672.10**	2033155.00	2055354.60	**2031979.00**	2048413.45
sko81_05	1308068.00	1318523.20	1308675.00	1313771.50	1305014.00	1313411.80	**1305014.00**	**1312719.90**
sko100_01	380540.00	385578.00	379068.00	**383854.00**	379638.00	385746.00	**378895.00**	384114.85
sko100_02	2081864.50	2100275.30	2081303.50	**2092804.05**	2083931.50	2093061.10	**2076182.50**	2094979.60
sko100_03	16173074.50	16299995.70	16166516.50	**16243748.40**	16154828.50	16303677.30	**16154828.50**	16292969.95
sko100_04	3253852.00	3269996.90	3243753.00	3259588.80	3232522.00	3249239.10	**3232522.00**	**3249839.75**
sko100_05	**1033617.50**	1041683.30	1035858.50	1041559.15	1035858.50	1042802.00	1036178.50	**1041220.00**

Table 2: The results between four configurations of the ADEP algorithm.

[17] Sunderesh S Heragu and Andrew Kusiak. 1988. Machine layout problem in flexible manufacturing systems. Operations research 36, 2 (1988), 258–268.

[18] Ravi Kothari and Diptesh Ghosh. 2014. An efficient genetic algorithm for single row facility layout. Optimization Letters 8, 2 (2014), 679–690.

[19] Ravi Kothari and Diptesh Ghosh. 2014. A scatter search algorithm for the single row facility layout problem. Journal of Heuristics 20, 2 (2014), 125–142.

[20] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13, 2 (1999), 129–170.

[21] Xiangtao Li and Minghao Yin. 2013. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Advances in Engineering Software 55 (2013), 10–31.

[22] Liping Liu, Yizhong Ma, Miaomiao Yu, and Yiliu Tu. 2014. Economic and economic-statistic designs of an X control chart for two-unit series systems with condition-based maintenance. Quality control and applied statistics 59, 1 (2014), 29–32.

[23] Jean-Claude Picard and Maurice Queyranne. 1981. On the One-Dimensional Space Allocation Problem. Operations Research 29, 2 (1981), 371–391.

[24] Manuel Rubio-Sánchez, Micael Gallego, Francisco Gortázar, and Abraham Duarte. 2016. GRASP with path relinking for the single row facility layout problem. Knowledge-Based Systems 106 (2016), 1–13.

[25] Hamed Samarghandi, Pouria Taabayan, and Farzad Firouzi Jahantigh. 2010. A particle swarm optimization for the single row facility layout problem. Computers & Industrial Engineering 58, 4 (2010), 529–534.

[26] Valentino Santucci, Marco Baioletti, Gabriele Di Bari, and Alfredo Milani. 2019. A Binary Algebraic Differential Evolution for the MultiDimensional Two-Way Number Partitioning Problem. In Evolutionary Computation in Combinatorial Optimization. Springer International Publishing, Cham, 17–32. https://doi.org/10.1007/978-3-030-16711-0_2

Instance	ADEP	PBIH	RPD	Instance	ADEP	PBIH	RPD
60dept_01	1477834.00	1477834.00	**0**	sko64_01	96889.00	96881	$< 10^{-3}$
60dept_02	841776.00	841776.00	**0**	sko64_02	634332.50	634332.5	**0**
60dept_03	648543.50	648337.50	$< 10^{-3}$	sko64_03	414349.50	414323.5	$< 10^{-3}$
60dept_04	398406.00	398406.00	**0**	sko64_04	297335.00	297129	0.001
60dept_05	318805.00	318805.00	**0**	sko64_05	501922.50	501922.5	**0**
70dept_01	1528537.00	1528537.00	**0**	sko72_01	139166.00	139150	$< 10^{-3}$
70dept_02	1441028.00	1441028.00	**0**	sko72_02	711998.00	711998	**0**
70dept_03	1518993.50	1518993.50	**0**	sko72_03	1054581.50	1054110.5	$< 10^{-3}$
70dept_04	968796.00	968796.00	**0**	sko72_04	920312.50	919586.5	0.001
70dept_05	4218002.50	4218002.50	**0**	sko72_05	428228.50	428226.5	$< 10^{-3}$
75dept_01	2393456.50	2393456.50	**0**	sko81_01	205213.00	205106	0.001
75dept_02	4321190.00	4321190.00	**0**	sko81_02	521434.50	521391.5	$< 10^{-3}$
75dept_03	1248537.00	1248423.00	$< 10^{-3}$	sko81_03	970931.00	970796	$< 10^{-3}$
75dept_04	3941816.50	3941816.50	**0**	sko81_04	2031979.00	2031803	$< 10^{-3}$
75dept_05	1791408.00	1791408.00	**0**	sko81_05	1305014.00	1302711	0.002
80dept_01	2069097.50	2069097.50	**0**	sko100_01	378895.00	378234	0.002
80dept_02	1921136.00	1921136.00	**0**	sko100_02	2076182.50	2076008.5	$< 10^{-3}$
80dept_03	3251368.00	3251368.00	**0**	sko100_03	16154828.50	16145614.5	0.001
80dept_04	3746515.00	3746515.00	**0**	sko100_04	3232522.00	3232522	**0**
80dept_05	1588885.00	1588885.00	**0**	sko100_05	1036178.50	1033080.5	0.003

Table 3: The comparison between the (EXC, rand/1, OB)+LS configuration and the current State of Art on the *debt* and *sko* instances.

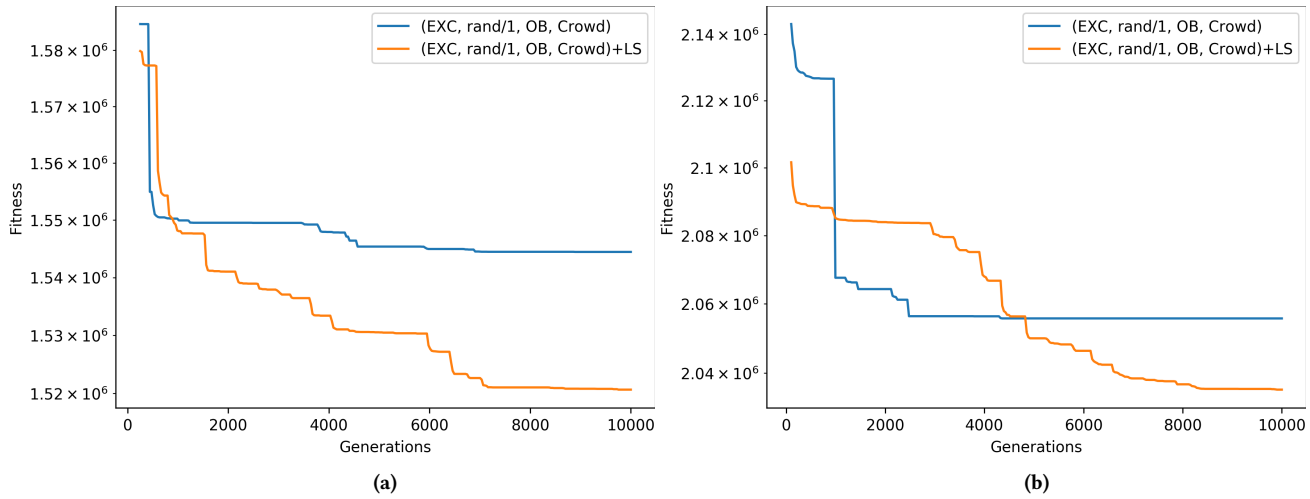

Figure 1: Convergence on *dept* and *sko* instances with respect to generations

[27] Valentino Santucci, Marco Baioletti, and Alfredo Milani. 2014. A Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem with Total Flow Time Criterion. In Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings. 161–170. https://doi.org/10.1007/978-3-319-10762-2_16

[28] Valentino Santucci, Marco Baioletti, and Alfredo Milani. 2015. An Algebraic Differential Evolution for the Linear Ordering Problem. In Proceedings of GECCO 2015. 1479–1480. https://doi.org/10.1145/2739482.2764693

[29] Valentino Santucci, Marco Baioletti, and Alfredo Milani. 2016. Algebraic Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem With Total Flowtime Criterion. IEEE Trans. Evolutionary Computation 20, 5 (2016), 682–694. https://doi.org/10.1109/TEVC.2015.2507785

[30] V. Santucci, M. Baioletti, and A. Milani. 2016. Solving Permutation Flowshop Scheduling Problems with a Discrete Differential Evolution Algorithm. AI Communications 29, 2 (2016), 269–286. https://doi.org/10.3233/AIC-150695

[31] Donald M Simmons. 1969. One-dimensional space allocation: an ordering algorithm. Operations Research 17, 5 (1969), 812–826.

[32] Rainer Storn and Kenneth Price. [n.d.]. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Jour. of Global Opt. 11, 4 ([n. d.]), 341–359. https://doi.org/10.1023/A:1008202821328

[33] Amalia Utamima et al. 2013. Hybrid Estimation of Distribution Algorithm for solving Single Row Facility Layout Problem. (2013).

[34] A. Milani V. Santucci, M. Baioletti. 2020. An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization. Swarm and Evolutionary Computation 55 (2020).

[35] Wei-Chang Yeh, Chyh-Ming Lai, Hsin-Yi Ting, Yunzhi Jiang, and Hsin-Ping Huang. 2017. Solving single row facility layout problem with simplified swarm optimization. In 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, 267–270.