# Self–organizing Migrating Algorithm for the Single Row Facility Layout Problem

Pavel Krömer
*Department of Computer Science*
*VSB - Technical University of Ostrava*
Ostrava, Czech Republic
pavel.kromer@vsb.cz

Jan Platoš
*Department of Computer Science*
*VSB - Technical University of Ostrava*
Ostrava, Czech Republic
jan.platos@vsb.cz

Václav Snášel
*Department of Computer Science*
*VSB - Technical University of Ostrava*
Ostrava, Czech Republic
vaclav.snasel@vsb.cz

***Abstract*—Single row facility layout problem is an important problem encountered in facility design, factory construction, production optimization, and other areas. At the same time, it is a challenging NP–hard combinatorial optimization problem that has been addressed by many advanced algorithms. In practical scenarios, real–world problems can be cast as single row facility location problem instances with different high–level properties and efficient algorithms that can solve them are sought. This work uses a variant of the self–organizing migration algorithm developed recently for permutation problems to tackle the single row facility layout problem and evaluates its accuracy and performance.**

***Index Terms*—single row facility layout problem, combinatorial optimization, self–organizing migrating algorithm, evolutionary computation**

## I. Introduction

Facility layout problems form a family of challenging tasks related to the design and planning of complex manufacturing facilities [1]. In general, they look for an optimal arrangement of elements (work and/or repair stations, production support facilities, equipment, machines, robots, etc.) to reduce total production costs.

Single row facility layout problem (SRFLP) is a linear placement problem that seeks optimal linear ordering of facility elements to minimize the sum of distances between each pair of them [1]. The SRFLP is an appealing problem with a number of practical applications in facility and building design, operations research, circuit design, physical data storage, and many other areas [2]. It is an NP–hard problem [2], [3] and, as a result, exact algorithms are able to solve small SRFLP instances only [4]. Various heuristic and metaheuristic methods have been employed to address larger SRFLP instances [2]. The evolutionary and swarm algorithms that have been used to solve SRFLP in the past include ant colony optimization, particle swarm optimization [5], [6], genetic algorithms [2], [7], [8], clonal selection algorithm, and bacterial foraging algorithm [9].

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic in project "Metaheuristics Framework for Multi-objective Combinatorial Optimization Problems (META MO-COP)", reg. no. LTAIN19176, and in part by the grants of the Student Grant System no. SP2020/108 and SP2020/161, VSB - Technical University of Ostrava, Czech Republic.

Self–organizing migrating algorithm (SOMA) [10] is a population–based stochastic evolutionary optimization algorithm that simulates the behaviour of a group of independent agents that explore a search space using a competitive–cooperative movement strategy. The agents (individuals) are represented by real–valued vectors that correspond to a particular location in the search space. Each position is evaluated by a multidimensional fitness function that assesses the quality of the individual within the context of the solved problem. The positions of the individuals are iteratively updated by the application of stochastic operators that simulate the competitive–cooperative behaviour. The real–valued nature of SOMA makes it a natural choice for high–dimensional continuous optimization problems. However, it has been recently used to address discrete [10], [11] and combinatorial optimization problems [12], too. In this work, random key SOMA, used previously to tackle two other prominent permutation problems, is used to address the SRFLP.

The rest of this paper is organized in the following way: the SRFLP is described in section II and the SOMA algorithm is outlined in section III. The random key SOMA algorithm for the SRFLP is introduced in section IV. Experimental evaluation of the random key SOMA on test SRFLP instances is detailed in section V. Finally, major conclusions are drawn and future work is outlined in section VI.

## II. Single Row Facility Layout Problem

The SRFLP is a linear placement problem that looks for an arrangement of a set of facilities with the same height (depth) on a straight line that would minimize the sum of weighted distances (transportation costs) between every two of them [2], [13]. Formally, the SRFLP is defined by a set of facility elements, $F = \{1, 2, \ldots, n\},\ n > 2$, a vector of element lengths, $\vec{L} = (l_1, l_2, \ldots l_n)$, and a symmetric transportation cost matrix, $C = \{c_{ij}\} \in \mathbb{R}^{n \times n}$. The goal of the SRFLP is to find an $n$–permutation, $\pi = (\pi_1, \pi_2, \ldots \pi_n)$, from the universe of all permutations of $n$ objects, $\mathcal{S}_n$, to minimize the total transportation costs associated with the arrangement it defines:

$$\min_{\pi \in \mathcal{S}_n} f_{\mathrm{SRFLP}}(\pi), \tag{1}$$

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

$$f_{\mathrm{SRFLP}}(\pi) = \sum_{l \le i < j \le n} \left[ c_{\pi_i \pi_j} \cdot d(\pi_i, \pi_j) \right], \quad (2)$$

$$d(\pi_i, \pi_j) = \frac{l_{\pi_i} + l_{\pi_j}}{2} + \sum_{i<k<j} l_{\pi_k}. \quad (3)$$

The SRFLP has been tackled by a variety of exact, heuristic, and metaheuristic algorithms. The exact methods look for an optimal solution to the problem. They include branch–and–bound methods [1], [7], graph–theoretic approaches [1], dynamic, semidefinite, and (mixed) integer linear programming [7], cutting, decomposition, and several improvement algorithms [1]. However, due to the hardness of the problem, exact methods are restricted to small SRFLP instances only. Problem instances up to the size of 42 facilities can be at present effectively solved by exact methods [4].

The heuristic methods for the SRFLP include constructive and improvement algorithms such as the modified spanning tree algorithm, the 2–opt and 3–opt algorithms [1], the greedy constructive method [7], the modified penalty algorithm [1], the population–based improvement heuristic with local search [14], and the greedy randomized adaptive search algorithm [15]. The metaheuristic methods for the SRFLP include tabu search [1], scatter search [2], simulated annealing, ant colony optimization, genetic algorithms [1], [7], [8], [16], multi–start simulated annealing [17], particle swarm optimization [6] and its variants [5], and the clonal selection and bacterial foraging algorithms [9].

The state–of–the–art SRFLP methods use local improvement methods and advanced concepts including greedy adaptive search [15], exhaustive 2–opt local search [13], [16], tabu–based local search [18], domain–specific evolutionary operators [7] and encoding [8], variable neighborhood search [19], location exchange heuristics [5], [6], multiobjective optimization [8], restarting [17], and various hybridization strategies [13], [18], [19].

Particle swarm optimization (PSO) is the sole nature–inspired optimization algorithm for continuous search spaces that was seemingly applied to the SRFLP in the past [5], [6]. The PSO by Samarghandi et al. [6] uses a factoradic permutation representation and modified particle velocity and location updating strategies. As a result, it represents particles by integers (uniquely mapped to $n$–permutations) and uses modified particle velocity and position update rules to simulate the behaviour of the swarm. Another PSO–like algorithm for the SRFLP, a method with simplistic particle updates by Yeh et al. [5], was in [5] not sufficiently detailed to analyze the solution encoding and the mapping between the continuous search space and the discrete feasible solution space. Besides, both algorithms use a form of local search based on facility location exchanges. In summary, it can be stated that no genuine continuous optimization method has been applied to the SRFLP and the ability of such methods to solve the instances of this problem is largely unknown.

This work contributes to this area and uses a traditional permutation representation for continuous spaces, the random key encoding, together with an efficient nature–inspired metaheuristic algorithm, SOMA, to search for SRFLP solutions. It adopts a pure–metaheuristic approach that does not use any form of local search to eliminate its effects and compares the SOMA algorithm with two other well–known nature–inspired optimization methods for continuous spaces, the particle swarm optimization and the differential evolution.

## III. Self–organizing migrating algorithm

Self–organizing migrating algorithm [10] is a stochastic optimization algorithm inspired by the competitive–cooperative behaviour of groups of individuals migrating through an environment in search for food. In a group of foragers, the individual with the best food source advertises its location and other group members move towards its location. On their way, they explore the environment and look for more attractive (i.e., better) food sources themselves. This simple nature–inspired principle is in SOMA implemented as an iterative population–based stochastic search process.

SOMA maintains a population of real–valued vectors (candidate solutions) that iteratively explore a multidimensional search space, $\mathbb{S}^n \subseteq \mathbb{R}^n$. Each individual is evaluated using a fitness function, $\mathcal{F} : \mathbb{S}^n \to \mathbb{R}$, that shows its goodness as a solution to the solved problem. In each iteration (migration loop), the best individual (leader) is selected and other population members move (migrate) towards its location. The migration consists of the exploration of the search space on a straight line between the moving individual and the leader. The moving individual performs a pre–defined number of fixed–length jumps on this line and the fitness of each position it visits is evaluated. At the end of the migration, it relocates to the best position it has visited.

To randomize the migration process and to increase the robustness of the algorithm, randomly selected coordinates of each moving individual are fixed during the migration. This is achieved by so–called perturbation vector, $\vec{prt}$, generated for every migration with respect to a fixed SOMA parameter, $PRT \in [0, 1]$, according to

$$\vec{prt}[j] = \begin{cases} 1 & \text{if } rand() < PRT \\ 0 & \text{if } rand() \ge PRT \end{cases}, \forall j \in \{1, \ldots, D\} \quad (4)$$

where $D$ is the dimension (number of coordinates) and $rand()$ is a pseudorandom number drawn from the range $[0, 1]$ for every coordinate of the $\vec{prt}$. The move, associated with the $k$–th migration jump of an individual, $\vec{x}_i$, is formally defined as

$$\vec{x}_i(k) = \vec{x}_i + \left[ (\vec{x}_L - \vec{x}_i) \odot \vec{prt}_i \right] \cdot (k \cdot step\_size), \quad (5)$$

where $\vec{x}_L$ is the position of the leader, $\vec{prt}_i$ is the perturbation vector, $step\_size$ is the fixed size of the jump, and $\odot$ is the elementwise multiplication operation.

SOMA is a successful optimization algorithm that has been applied to a number of theoretical as well as real–world optimization problems. It has been used to find optimum control parameters for laboratory chemical and plasma reactors, aircraft wing geometry optimization, synthesis of nonlinear

systems (deterministic chaos, periodic generators) and other applications [10]. A discrete version of SOMA was used to solve a variant of the flow–shop scheduling problem [11].

Nevertheless, SOMA is an optimization algorithm developed for continuous search spaces [10]. In order to use it in discrete search spaces (e.g., to solve combinatorial optimization problems), the real–valued individuals need to be translated to discrete problem solutions. This can be for some problems achieved easily by truncation, rounding, or similar elementary mathematical operations [10]. However, certain combinatorial optimization problems require solutions with a more sophisticated structure (combinations, permutations) that cannot be always guaranteed by naïve discretization methods. When infeasible solutions are generated in this way, they have to be detected and subsequently discarded or repaired [11]. It is apparent that such operations reduce the efficiency of the optimization process and represent an additional overhead.

## IV. Random key SOMA for the Single Row Facility Location Problem

SOMA is a population–based metaheuristic for optimization in continuous search spaces. The SRFLP is a discrete combinatorial optimization problem that requires well–structured solutions with stringent dependencies between elements (i.e., permutation elements). The application of continuous optimization methods to this problem requires sophisticated solution transformation or encoding to translate the real–valued candidate solutions to discrete spaces and to ensure that only feasible problem solutions are considered. The assessment of solution feasibility and their potential corrections introduce additional overhead to the optimization process. To avoid it, alternative representations and methods that transform arbitrary real–valued solution vectors to valid permutations have been proposed in the past. They include probability–based permutation recombination [20], position, precedence, and adjacency–based permutation representations [21], random key encoding [22], [23], and its variants [24]–[26].

The random key encoding belongs to permutation representations used by continuous optimization methods most often. The elements of real–valued candidate solutions are under the random key encoding ordered from smallest to largest and the resulting changes in positions are associated with permutation indices. The principle of the random key encoding is illustrated in eq. (6).

$$\begin{aligned} \vec{x} = &\begin{pmatrix} 0.2 & 0.3 & 0.1 & 0.5 & 0.4 \end{pmatrix} \rightarrow \\ &\begin{pmatrix} 0.2 & 0.3 & 0.1 & 0.5 & 0.4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \xrightarrow[\text{value}]{\text{sort by}} \\ &\begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix} \rightarrow \\ &\begin{pmatrix} 3 & 1 & 2 & 5 & 4 \end{pmatrix} = \pi \end{aligned} \tag{6}$$

The random key SOMA (rkSOMA) was developed to enable the application of SOMA to permutation problems [12]. In essence, it uses an arbitrary SOMA variant to evolve a population of real–valued candidate solutions inside an n–dimensional continuous search space, $\mathbb{S}^n$, and applies the random key encoding to transform them into valid $n$–permutations inside the feasible solution space, $\mathcal{S}_n$. The transformation can be formally expressed by the function,

$$\mathrm{rk} : \mathbb{R}^n \rightarrow \mathcal{S}_n. \tag{7}$$

The transformation, $\mathrm{rk}$, is evaluated according to the principles outlined in eq. (6) and implemented by an appropriate sorting algorithm such as quicksort or bubblesort. The SRFLP cost function, $f_{\mathrm{SRFLP}} : \mathcal{S}_n \rightarrow \mathbb{R}$, is then applied to the permutation, $\pi_i$, created from the individual, $x_i$,

$$\mathcal{F}(x_i) = f_{\mathrm{SRFLP}}(\mathrm{rk}(x_i)) \tag{8}$$

to evaluate its fitness.

## V. Experiments and results

In this work, rkSOMA is used to solve the single row facility location problem. Its ability to evolve SRFLP solutions is compared to two well–known evolutionary optimization algorithms for continuous search spaces: the PSO and the differential evolution (DE). The PSO and the DE were selected to conduct a wider comparison of the ability of three different types of metaheuristics for continuous optimization to solve the SRFLP. The three methods use different operations to evolve the population and to drive the evolutionary search process. That results in different high–level search strategies and different directions found by each one of them during the exploration of the search space associated with the SRFLP.

The methods were used in their basic variants with control parameters set to values recommended in the literature [27] (see table I for details). They also used the same random key encoding, the same fitness function defined by eq. (2), $f_{\mathrm{SRFLP}}$, and the maximum number of fitness function evaluations set to 1,000,000. All three algorithms were implemented in C++ and used to solve the same SRFLP instances.

The computational experiments were conducted on three groups of medium–sized SRFLP instances [28]–[30] that are extensively studied in the literature [14], [17]. The 'dept' instances with the number of facilities (i.e., problem dimension) ranging from 60 to 80 were introduced by Anjos et al. [29]. The 'sko' instances are based on the quadratic assignment problem and have dimensions between 64 and 100 [28]. Finally, the three 'SRFLP' instances all have the dimension 110 [30]. All test SRFLP instances were downloaded from the OPTSICOM project web page[1] and solved by DE, PSO, and rkSOMA, respectively. Because of the stochastic nature of the algorithms, all experiments were conducted 31 times independently.

The results of the experiments are summarized in table II. It shows the best (minimum) and the average (mean) fitness of the SRFLP solutions found by each algorithm for all problem instances during the 31 independent optimization runs. The table illustrates that rkSOMA delivered after 1,000,000 fitness function evaluations for all test instances better average

[1] http://grafo.etsii.urjc.es/optsicom/srflp/

TABLE I: Algorithms and settings.

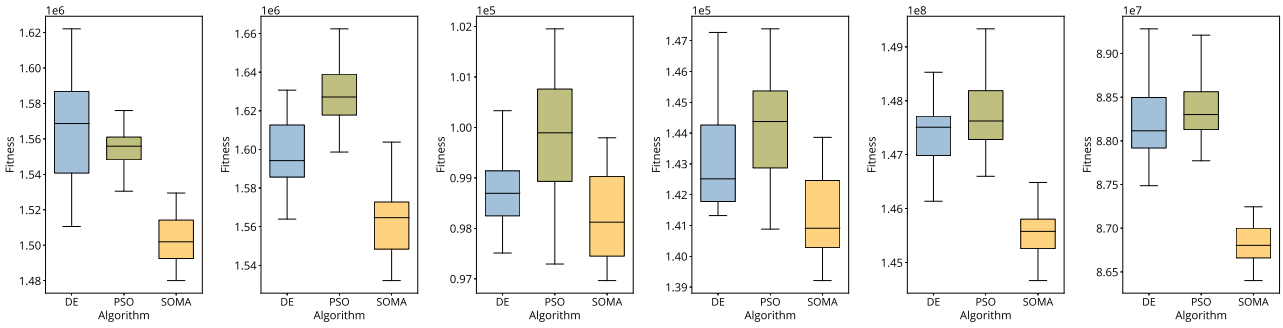
| DE | PSO | rkSOMA |
|---|---|---|
| /DE/rand/1 DE [27] with population size 100, scaling factor $F = 0.9$ and crossover probability $C = 0.9$. | AllToOne SOMA [10] with population size 100, perturbation probability $PRT = 0.02$, migration path length 3, and step size 0.21. | Global best PSO [27] with 100 particles, inertia weight, $w = 0.729$, cognitive component weight, $c_1 = 1.49445$, and social component weight, $c_2 = 1.49445$. |

TABLE II: Fitness of solutions found by rkSOMA, DE, and PSO for each test SRFLP instance after 1,000,000 fitness function evaluations.

| SRFLP | | DE | | PSO | | rkSOMA | |
|---|---|---|---|---|---|---|---|
| instance | n | min | mean | min | mean | min | mean |
| 60dept_01 | 60 | 1510576.0 | 1.567585e+06 | 1525112.0 | 1.554327e+06 | **1480068.0** | **1.504335e+06** |
| 60dept_02 | 60 | 853103.0 | 8.733891e+05 | 866048.0 | 8.858948e+05 | **842456.0** | **8.548742e+05** |
| 60dept_03 | 60 | 667281.5 | 6.800845e+05 | 670083.5 | 6.785872e+05 | **650065.5** | **6.578629e+05** |
| 60dept_04 | 60 | 404818.0 | 4.213691e+05 | 410793.0 | 4.241498e+05 | **399682.0** | **4.079065e+05** |
| 60dept_05 | 60 | 332897.0 | 3.522256e+05 | 334267.0 | 3.430993e+05 | **318922.0** | **3.291339e+05** |
| 70dept_01 | 70 | 1563909.0 | 1.598286e+06 | 1598640.0 | 1.633499e+06 | **1532073.0** | **1.562690e+06** |
| 70dept_02 | 70 | 1462873.0 | 1.501851e+06 | 1481073.0 | 1.525221e+06 | **1446300.0** | **1.467319e+06** |
| 70dept_03 | 70 | 1552350.5 | 1.590704e+06 | 1566302.5 | 1.607744e+06 | **1519029.5** | **1.545280e+06** |
| 70dept_04 | 70 | 992398.0 | 1.025611e+06 | 995740.0 | 1.032090e+06 | **970546.0** | **9.911380e+05** |
| 70dept_05 | 70 | 4344409.5 | 4.436033e+06 | 4304956.5 | 4.374208e+06 | **4246653.5** | **4.280043e+06** |
| 75dept_01 | 75 | 2452679.5 | 2.519129e+06 | 2463681.5 | 2.493391e+06 | **2419973.5** | **2.433152e+06** |
| 75dept_02 | 75 | 4494127.0 | 4.576587e+06 | 4411692.0 | 4.505680e+06 | **4334139.0** | **4.377426e+06** |
| 75dept_03 | 75 | 1282065.0 | 1.326362e+06 | 1289657.0 | 1.319964e+06 | **1254154.0** | **1.271260e+06** |
| 75dept_04 | 75 | 4045905.5 | 4.110310e+06 | 4052388.5 | 4.120730e+06 | **3962568.5** | **3.999076e+06** |
| 75dept_05 | 75 | 1827750.0 | 1.869228e+06 | 1834515.0 | 1.877394e+06 | **1794612.0** | **1.820219e+06** |
| 80dept_01 | 80 | 2117873.5 | 2.168170e+06 | 2146555.5 | 2.199808e+06 | **2078953.5** | **2.110553e+06** |
| 80dept_02 | 80 | 1975091.0 | 2.022697e+06 | 1986607.0 | 2.041766e+06 | **1921590.0** | **1.965766e+06** |
| 80dept_03 | 80 | 3385942.0 | 3.517317e+06 | 3357548.0 | 3.433013e+06 | **3275321.0** | **3.310982e+06** |
| 80dept_04 | 80 | 3901756.0 | 4.001769e+06 | 3851647.0 | 3.950943e+06 | **3791544.0** | **3.818050e+06** |
| 80dept_05 | 80 | 1614975.0 | 1.653284e+06 | 1631979.0 | 1.686725e+06 | **1593158.0** | **1.623936e+06** |
| sko64_01 | 64 | 97511.0 | 9.876106e+04 | 97293.0 | 9.976935e+04 | **96965.0** | **9.820226e+04** |
| sko64_02 | 64 | 638139.5 | 6.551250e+05 | 652971.5 | 6.630882e+05 | **634708.5** | **6.467020e+05** |
| †sko64_03 | 64 | **415562.5** | 4.227887e+05 | 419775.5 | 4.293972e+05 | 415814.5 | **4.198194e+05** |
| sko64_04 | 64 | 298973.0 | 3.030649e+05 | 300690.0 | 3.060607e+05 | **297735.0** | **3.005963e+05** |
| sko64_05 | 64 | 507794.5 | 5.184117e+05 | 512241.5 | 5.222276e+05 | **504378.5** | **5.108472e+05** |
| sko72_01 | 72 | 141322.0 | 1.431058e+05 | 140882.0 | 1.442254e+05 | **139211.0** | **1.412191e+05** |
| sko72_02 | 72 | 715647.0 | 7.279857e+05 | 720117.0 | 7.383161e+05 | **712739.0** | **7.213017e+05** |
| sko72_03 | 72 | 1071834.5 | 1.087603e+06 | 1084122.5 | 1.102282e+06 | **1057619.5** | **1.069878e+06** |
| sko72_04 | 72 | 939162.5 | 9.532920e+05 | 945274.5 | 9.641629e+05 | **928741.5** | **9.350713e+05** |
| sko72_05 | 72 | 433776.5 | 4.401781e+05 | 433532.5 | 4.440246e+05 | **429052.5** | **4.338203e+05** |
| sko81_01 | 81 | 208108.0 | 2.107385e+05 | 206948.0 | 2.107928e+05 | **205560.0** | **2.075386e+05** |
| sko81_02 | 81 | 529529.5 | 5.365674e+05 | 535190.5 | 5.435435e+05 | **523220.5** | **5.296382e+05** |
| sko81_03 | 81 | 982011.0 | 1.006650e+06 | 990541.0 | 1.010486e+06 | **972096.0** | **9.845067e+05** |
| sko81_04 | 81 | 2055801.0 | 2.091972e+06 | 2073011.0 | 2.117406e+06 | **2041526.0** | **2.065163e+06** |
| sko81_05 | 81 | 1327640.0 | 1.344975e+06 | 1328570.0 | 1.354944e+06 | **1307172.0** | **1.323458e+06** |
| sko100_01 | 100 | 382613.0 | 3.876576e+05 | 381975.0 | 3.897804e+05 | **380109.0** | **3.844677e+05** |
| sko100_02 | 100 | 2129085.5 | 2.163478e+06 | 2123073.5 | 2.167038e+06 | **2084031.5** | **2.108740e+06** |
| sko100_03 | 100 | 16466134.5 | 1.684543e+07 | 16715222.5 | 1.699491e+07 | **16185228.5** | **1.640159e+07** |
| sko100_04 | 100 | 3306661.0 | 3.371147e+06 | 3324778.0 | 3.393381e+06 | **3256219.0** | **3.281190e+06** |
| sko100_05 | 100 | 1051849.5 | 1.064591e+06 | 1051711.5 | 1.077137e+06 | **1037890.5** | **1.048614e+06** |
| SRFLP1 | 110 | 146135914.5 | 1.473977e+08 | 146598728.5 | 1.477776e+08 | **144663856.5** | **1.455824e+08** |
| SRFLP2 | 110 | 87004206.0 | 8.824770e+07 | 87772217.0 | 8.841421e+07 | **86400517.0** | **8.681574e+07** |
| SRFLP3 | 110 | 2267202.5 | 2.283647e+06 | 2279940.5 | 2.300523e+06 | **2247873.5** | **2.263356e+06** |

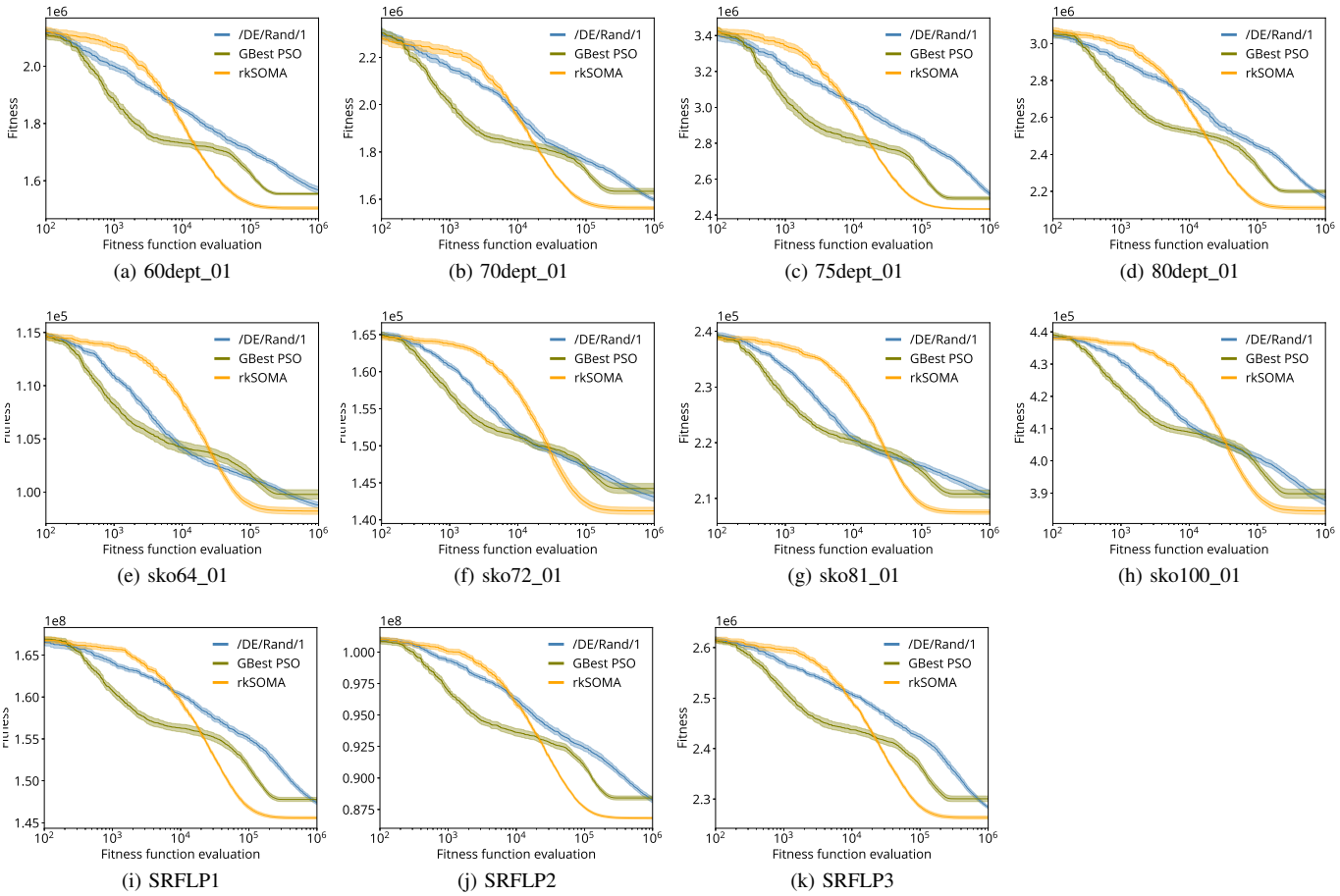
solutions than the DE and the PSO. It has also found better minimum solutions than both reference algorithms for all test instances except 'sko64_03', for which was the overall best solution delivered by the DE. However, the fitness of the average solution for 'sko64_03' evolved by the DE was still worse than the fitness of the average solution discovered for this SRFLP instance by rkSOMA. All the differences between the fitness of the SRFLP solutions found by rkSOMA and both the DE and the PSO were for all problem instances statistically significant at significance level $\alpha = 0.01$. The minimum solutions found by rkSOMA had fitness by between 0.41 ('sko_72_02') to 4.81% ('60dept_05') lower than the fitness of the minimum solutions found by the reference algorithms. The average solutions found by rkSOMA had fitness lower by between 0.57 ('sko_64_01') and 7.02% ('60dept_05') in comparison with the solutions found by the DE and the PSO. The differences between the final solutions evolved by rkSOMA, the DE, and the PSO are for selected test SRFLP instances illustrated by boxplots shown in fig. 1.

The progress of SRFLP solution evolution by rkSOMA, DE, and PSO was further analyzed in order to compare the ability of the algorithms to discover good problem solutions and to converge towards optimal (suboptimal) ones in time. The evolution of solutions to selected SRFLP instances with




(a) 60dept_01 (b) 70dept_01 (c) sko64_01 (d) sko72_01 (e) SRFLP1 (f) SRFLP2

Fig. 1: Fitness of the final solutions found by rkSOMA, DE, and PSO to selected SRFLP instances during the independent optimization runs.




(a) 60dept_01 (b) 70dept_01 (c) 75dept_01 (d) 80dept_01

(e) sko64_01 (f) sko72_01 (g) sko81_01 (h) sko100_01

(i) SRFLP1 (j) SRFLP2 (k) SRFLP3

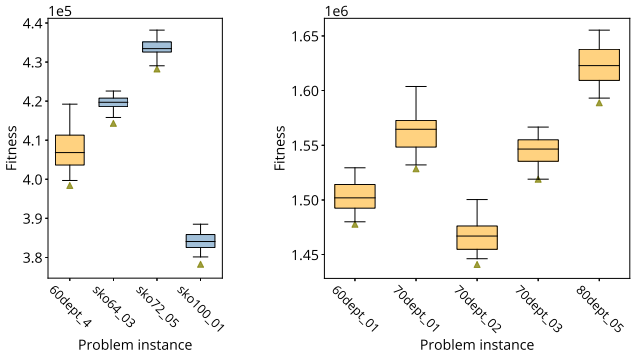
Fig. 2: Evolution of solutions to selected SRFLP instances by rkSOMA, DE, and PSO. Solid lines represent the fitness of the mean solutions found at given fitness function evaluation, coloured bands represent 95% confidence intervals around means.

different dimensions is illustrated in fig. 2. It can be immediately seen that the three compared algorithms lead to optimization processes with different high–level properties. Although rkSOMA delivered the best final solutions, it was outperformed by both the DE and the PSO at the initial phase of the evolutionary search process. The PSO provided better SRFLP solutions in the course of the first approximately 10,000 fitness function evaluations and the DE during the first 30,000 – 50,000 fitness function evaluations, as clearly demonstrated by the plots in fig. 2. The solutions found by the basic version of the PSO algorithm were during most fitness function evaluations better than the solutions found by

the basic DE. However, the DE discovered for 32 out of 43 test SRFLP instances solutions with better minimum fitness and for 34 test SRFLP instances solutions with better average fitness by the end of the evolution. This is an interesting observation that indicates that despite being outperformed by rkSOMA in the long run, the DE and the PSO can be useful for the evolution of SRFLP solutions when the number of fitness function evaluations is limited.

Finally, the SRFLP solutions obtained by the best performing algorithm, rkSOMA, were compared to the SRFLP solutions obtained by a state–of–the–art evolutionary method for the SRFLP, the genetic algorithm by Kothari and Gosh named GENALGO [16]. However, it should be noted that GENALGO uses a local search to improve the solutions during the evolution while no such mechanism is adopted by rkSOMA. The results of the comparison are summarized in table III and for selected problem instances illustrated in fig. 3.

The table shows that the SRFLP solutions evolved by rkSOMA do not match the best solutions found by GENALGO with local search. The pure metaheuristic method was able to discover SRFLP solutions with the best fitness higher by between 0.29 to 4.23% than those evolved by the genetic algorithm with local search. The comparison of the best SRFLP solutions found by rkSOMA and GENALGO is for selected problem instances also shown in fig. 3. The figure confirms that the fitness of the typical solutions evolved by rkSOMA is higher (i.e., worse) than the fitness of the best solutions found by GENALGO with local search. Nevertheless, the pure metaheuristic approach demonstrates the ability to evolve SRFLP solutions with error lower than 5% when compared to the genetic algorithm with local search.




(a) Problem instances with dimensions 60, 64, 72, and 100.

(b) Problem instances with dimensions 60, 70, and 80.

Fig. 3: Fitness of the final solutions evolved by rkSOMA (boxplots) and the best solutions obtained by GENALGO [16] (triangles) for selected SRFLP instances.

## VI. Conclusions

This work proposed and studied a novel nature–inspired metaheuristic optimization algorithm for the single row facility location problem. The solutions to this hard combinatorial optimization problem were modeled with the help of the widely used random key encoding and evolved by a powerful nature–inspired method for continuous optimization, the SOMA algorithm. The properties of SOMA as an SRFLP solver were studied on 43 well–known SRFLP instances and compared to two other popular evolutionary and swarm methods for multidimensional continuous optimization, the DE and the PSO, in a series of computational experiments.

The experimental evaluation showed that the proposed rkSOMA algorithm outperforms the basic variants of both, the DE and the PSO. In the course of 1,000,000 fitness function evaluations, rkSOMA was able to find better average SRFLP solutions than the DE and the PSO. Also, the best SRFLP solutions discovered by rkSOMA were better than the best solutions found by the DE and the PSO for all but one problem instance. Nevertheless, a detailed analysis of the progress of the evolutionary optimization revealed that the PSO and the DE delivered better solutions than rkSOMAduring the first around 10,000 to 50,000 fitness function evaluations. Only after that, rkSOMA took the lead and evolved better problem solutions. When it comes to the reference algorithms, the solutions evolved by the basic PSO were better than those discovered by the basic DE during the majority of fitness function evaluations.

The fitness of the best SRFLP solutions found by rkSOMA in a pure metaheuristic manner without any form of local search was only by less than 5% worse than the fitness of the solutions found by a genetic algorithm with local solution improvement. This is an encouraging result indicating a good ability of the nature–inspired continuous optimization strategy represented by rkSOMA to find good SRFLP solutions.

Future research in this field will pursue several directions. Other types of SRFLP instances with different properties and dimensions will be addressed by rkSOMA and other relevant optimization algorithms. Different high–level variants of the facility location problem will be considered, too. Next, various other metaheuristic algorithms for continuous multidimensional optimization, including advanced variants of SOMA, the DE, and the PSO will be used. Last but not least, the ability of metaheuristics for continuous optimization together with different variants of local search to solve the SRFLP will be studied.

## References


[1] S. Heragu, *Facilities Design*. CRC Press, 2018.

[2] R. Kothari and D. Ghosh, "The single row facility layout problem: state of the art," *OPSEARCH*, vol. 49, no. 4, pp. 442–462, Dec 2012.

[3] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237 – 267, 1976.

[4] M. F. Anjos, A. Fischer, and P. Hungerländer, "Improved exact approaches for row layout problems with departments of equal length," *European Journal of Operational Research*, vol. 270, no. 2, pp. 514 – 529, 2018.

[5] W.-C. Yeh, C.-M. Lai, H.-Y. Ting, Y. Jiang, and H.-P. Huang, "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)*, July 2017, pp. 267–270.

TABLE III: Fitness of SRFLP solutions evolved by rkSOMA compared to the fitness of solutions found by GENALGO [16].

| SRFLP | | | rkSOMA | | | |
|---|---|---|---|---|---|---|
| instance | n | GENALGO | min | mean | $\sigma$ | max |
| 60dept_01 | 60 | 1477834.0 | 1480068.0 | 1.504335e+06 | 1.38e+04 | 1529444.0 |
| 60dept_02 | 60 | 841776.0 | 842456.0 | 8.548742e+05 | 8.25e+03 | 871516.0 |
| 60dept_03 | 60 | 648337.5 | 650065.5 | 6.578629e+05 | 4.22e+03 | 667754.5 |
| 60dept_04 | 60 | 398406.0 | 399682.0 | 4.079065e+05 | 5.13e+03 | 419222.0 |
| 60dept_05 | 60 | 318805.0 | 318922.0 | 3.291339e+05 | 4.00e+03 | 337310.0 |
| 70dept_01 | 70 | 1528537.0 | 1532073.0 | 1.562690e+06 | 1.78e+04 | 1603783.0 |
| 70dept_02 | 70 | 1441028.0 | 1446300.0 | 1.467319e+06 | 1.44e+04 | 1500390.0 |
| 70dept_03 | 70 | 1518993.5 | 1519029.5 | 1.545280e+06 | 1.32e+04 | 1566652.5 |
| 70dept_04 | 70 | 968796.0 | 970546.0 | 9.911380e+05 | 1.19e+04 | 1018117.0 |
| 70dept_05 | 70 | 4218002.5 | 4246653.5 | 4.280043e+06 | 1.88e+04 | 4318357.5 |
| 75dept_01 | 75 | 2393456.5 | 2419973.5 | 2.433152e+06 | 8.30e+03 | 2448767.5 |
| 75dept_02 | 75 | 4321190.0 | 4334139.0 | 4.377426e+06 | 1.86e+04 | 4409768.0 |
| 75dept_03 | 75 | 1248423.0 | 1254154.0 | 1.271260e+06 | 8.96e+03 | 1295144.0 |
| 75dept_04 | 75 | 3941816.5 | 3962568.5 | 3.999076e+06 | 2.53e+04 | 4067007.5 |
| 75dept_05 | 75 | 1791408.0 | 1794612.0 | 1.820219e+06 | 1.23e+04 | 1852273.0 |
| 80dept_01 | 80 | 2069097.5 | 2078953.5 | 2.110553e+06 | 2.51e+04 | 2173877.5 |
| 80dept_02 | 80 | 1921136.0 | 1921590.0 | 1.965766e+06 | 2.06e+04 | 2007924.0 |
| 80dept_03 | 80 | 3251368.0 | 3275321.0 | 3.310982e+06 | 1.59e+04 | 3347221.0 |
| 80dept_04 | 80 | 3746515.0 | 3791544.0 | 3.818050e+06 | 1.27e+04 | 3842485.0 |
| 80dept_05 | 80 | 1588885.0 | 1593158.0 | 1.623936e+06 | 1.62e+04 | 1655295.0 |
| sko64_01 | 64 | 96881.0 | 96965.0 | 9.820226e+04 | 8.31e+02 | 99794.0 |
| sko64_02 | 64 | 634332.5 | 634708.5 | 6.467020e+05 | 5.58e+03 | 658519.5 |
| sko64_03 | 64 | 414323.5 | 415814.5 | 4.198194e+05 | 2.30e+03 | 427069.5 |
| sko64_04 | 64 | 297129.0 | 297735.0 | 3.005963e+05 | 1.82e+03 | 303910.0 |
| sko64_05 | 64 | 501922.5 | 504378.5 | 5.108472e+05 | 3.53e+03 | 517527.5 |
| sko72_01 | 72 | 139150.0 | 139211.0 | 1.412191e+05 | 1.29e+03 | 143862.0 |
| sko72_02 | 72 | 711998.0 | 712739.0 | 7.213017e+05 | 4.79e+03 | 731131.0 |
| sko72_03 | 72 | 1054110.5 | 1057619.5 | 1.069878e+06 | 5.69e+03 | 1085758.5 |
| sko72_04 | 72 | 919586.5 | 928741.5 | 9.350713e+05 | 5.13e+03 | 949646.5 |
| sko72_05 | 72 | 428226.5 | 429052.5 | 4.338203e+05 | 2.53e+03 | 439463.5 |
| sko81_01 | 81 | 205106.0 | 205560.0 | 2.075386e+05 | 1.09e+03 | 209306.0 |
| sko81_02 | 81 | 521391.5 | 523220.5 | 5.296382e+05 | 2.36e+03 | 532914.5 |
| sko81_03 | 81 | 970796.0 | 972096.0 | 9.845067e+05 | 4.99e+03 | 994937.0 |
| sko81_04 | 81 | 2031803.0 | 2041526.0 | 2.065163e+06 | 1.33e+04 | 2093460.0 |
| sko81_05 | 81 | 1302711.0 | 1307172.0 | 1.323458e+06 | 6.65e+03 | 1334665.0 |
| sko100_01 | 100 | 378234.0 | 380109.0 | 3.844677e+05 | 2.89e+03 | 392009.0 |
| sko100_02 | 100 | 2076008.5 | 2084031.5 | 2.108740e+06 | 1.20e+04 | 2129400.5 |
| sko100_03 | 100 | 16145614.5 | 16185228.5 | 1.640159e+07 | 9.72e+04 | 16655235.5 |
| sko100_04 | 100 | 3232522.0 | 3256219.0 | 3.281190e+06 | 1.27e+04 | 3301779.0 |
| sko100_05 | 100 | 1033080.5 | 1037890.5 | 1.048614e+06 | 5.60e+03 | 1058880.5 |
| SRFLP1 | 110 | 144296664.5 | 144663856.5 | 1.455824e+08 | 4.26e+05 | 146484273.5 |
| SRFLP2 | 110 | 86050037.0 | 86400517.0 | 8.681574e+07 | 2.25e+05 | 87245295.0 |
| SRFLP3 | 110 | 2234743.5 | 2247873.5 | 2.263356e+06 | 8.44e+03 | 2283441.5 |


[6] H. Samarghandi, P. Taabayan, and F. F. Jahantigh, "A particle swarm optimization for the single row facility layout problem," *Computers Industrial Engineering*, vol. 58, no. 4, pp. 529 – 534, 2010.

[7] D. Datta, A. R. Amaral, and J. R. Figueira, "Single row facility layout problem using a permutation-based genetic algorithm," *European Journal of Operational Research*, vol. 213, no. 2, pp. 388 – 394, 2011.

[8] G. Aiello, G. L. Scalia, and M. Enea, "A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 352 – 10 358, 2012.

[9] B. H. Ulutas, "Assessing the Performance of Two Bioinspired Algorithms to Solve Single-Row Layout Problem," *International Journal of Manufacturing Engineering*, vol. 2013, p. 11, 2013.

[10] I. Zelinka, "Soma—self-organizing migrating algorithm," in *Self-Organizing Migrating Algorithm: Methodology and Implementation*, D. Davendra and I. Zelinka, Eds. Cham: Springer International Publishing, 2016, pp. 3–49.

[11] D. Davendra, I. Zelinka, M. Pluhacek, and R. Senkerik, "Dsoma—discrete self organising migrating algorithm," in *Self-Organizing Migrating Algorithm: Methodology and Implementation*, D. Davendra and I. Zelinka, Eds. Cham: Springer International Publishing, 2016, pp. 51–63.

[12] P. Kromer, J. Janoušek, and J. Platoš, "Random key self–organizing migrating algorithm for permutation problems," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, June 2019, pp. 2878–2885.

[13] R. Kothari and D. Ghosh, "Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods," *European Journal of Operational Research*, vol. 224, no. 1, pp. 93 – 100, 2013.

[14] S. Atta and P. R. Sinha Mahapatra, "Population-based improvement heuristic with local search for single-row facility layout problem," *Sādhanā*, vol. 44, no. 11, p. 222, Oct 2019.

[15] G. Cravo and A. Amaral, "A grasp algorithm for solving large-scale single row facility layout problems," *Computers Operations Research*, vol. 106, pp. 49 – 61, 2019.

[16] R. Kothari and D. Ghosh, "An efficient genetic algorithm for single row facility layout," *Optimization Letters*, vol. 8, no. 2, pp. 679–690, Feb 2014.

[17] G. Palubeckis, "Single row facility layout using multi-start simulated annealing," *Computers Industrial Engineering*, vol. 103, pp. 1 – 16, 2017.

[18] C. Ou-Yang and A. Utamima, "Hybrid estimation of distribution algorithm for solving single row facility layout problem," *Computers & Industrial Engineering*, vol. 66, no. 1, pp. 95 – 103, 2013.

[19] J. Guan and G. Lin, "Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem," *European Journal of Operational Research*, vol. 248, no. 3, pp. 899 – 909, 2016.

[20] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.*, April 2003, pp. 243–246.

[21] T. Gong and A. L. Tuson, "Forma analysis of particle swarm optimisation for permutation problems," *J. Artif. Evol. App.*, vol. 2008, pp. 4:1–4:16, Jan. 2008.

[22] J. C. Bean, "Genetic algorithms and random keys for sequencing and

optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.

[23] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38–53, 2006.

[24] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930 – 1947, 2007.

[25] B. Qian, L. Wang, R. Hu, W.-L. Wang, D.-X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7, pp. 757–777, Sep 2008.

[26] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, vol. 55, pp. 10 – 31, 2013.

[27] A. Engelbrecht, *Computational Intelligence: An Introduction, 2nd Edition*. New York, NY, USA: Wiley, 2007.

[28] M. F. Anjos and G. Yen, "Provably near-optimal solutions for very large single-row facility layout problems," *Optimization Methods and Software*, vol. 24, no. 4-5, pp. 805–817, 2009.

[29] M. F. Anjos, A. Kennings, and A. Vannelli, "A semidefinite optimization approach for the single-row layout problem with unequal dimensions," *Discrete Optimization*, vol. 2, no. 2, pp. 113 – 122, 2005.

[30] A. R. S. Amaral and A. N. Letchford, "A polyhedral approach to the single row facility layout problem," *Mathematical Programming*, vol. 141, no. 1, pp. 453–477, Oct 2013.