The International Journal of Advanced Manufacturing Technology (2018) 98:2173–2184
https://doi.org/10.1007/s00170-018-2370-6

ORIGINAL ARTICLE

A constrained single-row facility layout problem

Zahnupriya Kalita[1] · Dilip Datta[1]

Received: 30 June 2017 / Accepted: 22 June 2018 / Published online: 2 July 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract

The single-row facility layout problem (SRFLP) seeks the arrangement of given facilities along a straight row in such a way that the total material handling cost among the facilities is minimized. The SRFLP is studied till date as an unconstrained problem allowing the placement of the facilities in any location in any order without any restriction. However, a practical SRFLP instance may need to satisfy different types of constraints imposed on the placement of its facilities, e.g., the operation sequencing with precedence constraints in a process planning can be modeled as a SRFLP with ordering constraints. Such a SRFLP model, named as the *constrained* SRFLP (cSRFLP), is introduced here by instructing to place some facilities in fixed positions, and/or in specified orders with or without allowing the placement of other facilities in between two ordered facilities. Since it would be computationally too expensive for any search technique to satisfy such constraints, a permutation-based genetic algorithm (pGA), named as the *constrained* pGA (cpGA in short), is also proposed with some specially designed operators for exploring only feasible solutions of cSRFLP. In the numerical experimentation, investigating three case studies of the operation sequencing problem of process planning as cSRFLP instances, the cpGA found new sequences of operations with the same best-known objective value for the smaller-size case study, while improved the best-known solutions of the other two case studies of larger sizes. Further, transforming some large-size benchmark instances of SRFLP into cSRFLP, the cpGA found marginally inferior solutions than their best-known SRFLP solutions, which is obvious due to the constraints imposed in the transformed cSRFLP instances.

Keywords Combinatorial optimization · Facility layout design · Operation sequencing · Constraint · Genetic algorithm

1 Introduction

The well-known single-row facility layout problem (SRFLP) deals with the effective arrangement of given facilities along a straight row on one side of a corridor, so as to minimize the total material handling cost among the facilities. It has applications in various areas, including installation of machines in flexible manufacturing systems where automated guided vehicles transport materials among the machines laid down along a straight line [12], or arrangement of departments in office buildings, stores in supermarkets, laboratories in hospitals, and so on [34, 38].

✉ Dilip Datta
ddatta@tezu.ernet.in; datta_dilip@rediffmail.com

Zahnupriya Kalita
zk@tezu.ernet.in

[1] Department of Mechanical Engineering, School of Engineering, Tezpur University, Tezpur, 784028, India

The SRFLP is an NP-hard combinatorial optimization problem [1, 10, 17], which has been investigated by both exact and heuristic approaches. Methods investigated for exact solutions of the SRFLP include branch-and-bound [38], dynamic programming [34], semidefinite programming [2], and mixed-integer linear programming [24]. To deal with large-size instances in lesser computational time, various heuristic and metaheuristic procedures have been adopted, e.g., hybrid simulated annealing [11], constructive greedy heuristic [20], and ant colony algorithm [39]. Recent work with heuristics for the SRFLP includes scatter search [21], particle swarm optimization [37], tabu search [36], and genetic algorithm [6].

The SRFLP is usually handled as an unconstrained optimization problem, in which the facilities can be arranged in any order without any restriction. However, while handling practical problems, one might encounter some positioning and ordering constraints on the placement of some given facilities, such as arranging a facility in a fixed position, two facilities together, and a facility prior to another. Although there are many other models taking care of such constraints, the SRFLP could not be found studying in that direction. For example, in the hospital facility layout design problem studied by Padgaonkar [32], the departments requiring more inter-floor movement were fixed closer to the elevator and those requiring frequent inter-departmental movement were fixed on the same or the next floor. Observing the possibility of more impulse buying if a customer passes through more items, Ozgormus [31] designed the block layout of a grocery store by dispersing basic commodities (such as milk, meat, frozen foods, and breads/cereals) at the end of the store. In regard of placing facilities in some given orders, operation sequencing in process planning is a well-known problem. The operation sequencing problem seeks the manufacturing operations, required for machining an industrial part, to be sequenced with some precedence (i.e., ordering) constraints in a way to minimize the total machining cost [16]. Precedence constraints among machining operations become essential if some features of an industrial part cannot be machined prior to machining some other features, e.g., a hole is to be drilled in a work-piece before it is tapped [25, 41]. It is also observed that the ordering of operations in sequence flexibility-based manufacturing systems can be varied by carrying out some operations in sequence, while others without any such restriction [22]. Since a manufacturing system is neither very restrictive like the classical job-shop problem (which involves strict ordering of the operations to be performed) nor so flexible as the generalized open-shop problem (in which the operations can be performed in any order), such restrictions in sequencing operations are quite common in many manufacturing systems [13]. Genetic algorithm [14, 35], ant colony algorithm [19], simulated annealing [14, 29], and particle swarm optimization [23, 27, 33] are widely used metaheuristics for finding optimal operation sequences in process planning.

Motivated by practical requirements as above, a constrained SRFLP model, named as the *constrained single-row facility layout problem* (or, cSRFLP in short), is introduced here considering positioning and ordering constraints on the placement of some facilities. Also, the operation sequencing problem with precedence constraints is illustrated as a variant of the cSRFLP with ordering constraints. It is to be noted that it would be computationally very expensive [28, 40], or even impossible, for a search technique to satisfy such integer constraints on its own. Therefore, a novel permutation-based genetic algorithm, named as the *constrained permutation-based genetic algorithm* (or, cpGA in short), is also proposed for exploring only the feasible space of the cSRFLP. In the cpGA, a greedy heuristic algorithm initializes its population with randomly generated feasible solutions, while a crossover operator generates new feasible solutions by exploiting old solutions. Similarly, a mutation operator is developed for exploring the neighborhood of newly generated solutions by preserving their feasibility. In the numerical experimentation, results of some cSRFLP instances of various sizes are presented by comparing with the results of the operation sequencing problem and their SRFLP-based values. It is to be mentioned that until Datta et al. [6] formulated the general SRFLP model as an unconstrained permutation-based optimization problem, it was studied with various constraints on overlapping of the facilities and their placement in a row of length equal to the sum of the lengths of all the individual facilities. The positioning and ordering constraints considered in the cSRFLP model are not to be confused with those overlapping and length-based constraints used in the exact mathematical formulation of the general SRFLP model. Apart from the operation sequencing problem as illustrated in the present work, the proposed cSRFLP model can also be applied for finding the optimum sequence of assembly operations in the product assembling problem.

The rest of the article is organized as follows: the cSRFLP model is introduced in Section 2, followed by the proposed cpGA in Section 3. Performing numerical experimentation with three instances of the operation sequencing problem of process planning and some selective large-size cSRFLP instances in Section 4, the article is finally concluded in Section 5.

2 The proposed cSRFLP model

As described in Section 1, the cSRFLP model introduced in the present work involves the effective arrangement of given facilities along a straight row by minimizing the total material handling cost among the facilities subject to the following two types of constraints:

(1) *Positioning constraints*: Some facilities are to be placed in specified fixed positions.
(2) *Ordering constraints*: Some pairs of facilities are to be placed in specified orders, with or without allowing the placement of other facilities in between an ordered pair.

2.1 The cSRFLP formulation

As per above discussion, the cSRFLP model can be defined as a problem seeking the effective arrangement of n number of given facilities along a straight row with some facilities in fixed positions and some pairs of facilities in specified orders with or without allowing any other facilities in between an ordered pair.

Accordingly, the following notations are defined for formulating the problem (constraint related parameters are denoted by uppercase alphabets):

Indices

i, j: Indices of facilities and positions
k: Index of ordered pairs

Fixed parameters

n: Number of facilities to be arranged
l_i: Length of facility i
c_{ij}: Material handling cost between facilities i and j; $i \neq j$
T_i: Indicator of the fixed position for facility i; $T_i \in$ {yes, no}
P_i: Specified fixed position of facility i (positioning constraint)
O_i: Indicator for ordering facility i on left side (ordering constraint); $O_i \in$ {yes, no}
M_i: Number of facilities to be placed on the right side of facility i; $M_i = 0$ if $O_i =$ no
R_{ik}: Facility in pair k to be placed right to facility i (ordering constraint); $k = 1, 2, \ldots, M_i$
A_{ik}: Indicator for placing other facilities in between facility i and its pairing facility in pair k; $A_{ik} \in$ {yes, no}

Variables

π: Index of a permutation of given n facilities
r_i^π: Facility placed at position i of π
$x_{r_i^\pi}$: Centroidal distance of facility r_i^π from the left corner of the row
f^π: Total material handling cost among the facilities of π

In terms of the above notations, the cSRFLP model for permutation π can be expressed mathematically as follows:

$$\text{Minimize} \quad f^\pi = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{r_i^\pi, r_j^\pi} \left| x_{r_j^\pi} - x_{r_i^\pi} \right| \tag{1}$$

$$\text{Subject to} \quad P_{r_i^\pi} = i \, ; \qquad \text{if} \quad T_{r_i^\pi} = \mathrm{Y} \, ; \; i = 1, 2, \ldots, n \tag{2}$$

$$j = i + 1 \, ; \qquad \text{if} \quad \begin{cases} O_{r_i^\pi} = \mathrm{Y} \, ; \; R_{r_i^\pi, k} = r_j^\pi \; ; \; A_{r_i^\pi, k} = \mathrm{N} \\ k = 1, 2, \ldots, M_i \; ; \; i, j = 1, 2, \ldots, n \end{cases} \tag{3}$$

$$j \geqslant i + 1 \, ; \qquad \text{if} \quad \begin{cases} O_{r_i^\pi} = \mathrm{Y} \, ; \; R_{r_i^\pi, k} = r_j^\pi \; ; \; A_{r_i^\pi, k} = \mathrm{Y} \\ k = 1, 2, \ldots, M_i \; ; \; i, j = 1, 2, \ldots, n \end{cases} \tag{4}$$

$$\text{where} \quad x_{r_i^\pi} = \frac{l_{r_1^\pi}}{2} + \frac{1}{2} \sum_{j=2}^{i} \left(l_{r_{j-1}^\pi} + l_{r_j^\pi} \right) \, . \tag{5}$$

In Eq. 1, f^π is the objective function for permutation π. The positioning constraints are expressed by Eq. 2, where $P_{r_i^\pi}$ is the specified fixed position of r_i^π when it is subject to the positioning constraint. On the other hand, Eq. 3 represents the ordering constraints without allowing the placement of other facilities in between an ordered pair of facilities ($A_{r_i^\pi, k} = \mathrm{N}$), where $R_{r_i^\pi, k} = r_j^\pi$ means that facilities r_i^π and r_j^π are ordered in pair k with r_j^π right to r_i^π. Similarly, Eq. 4 also represents the ordering constraints, but allowing the placement of other facilities in between an ordered pair of facilities ($A_{r_i^\pi, k} = \mathrm{Y}$). Finally, $x_{r_i^\pi}$ in Eq. 5 is the centroidal distance of r_i^π from the left corner of permutation π.

2.2 The true optimum of the cSRFLP model

Since the number of permutations of a string is finite, a cSRFLP instance can easily be solved by coding all the permutations of the given facilities. In this process, the feasible permutations may be sorted out first by checking the satisfactions of the constraints given by Eqs. 2–4, and then the optimum permutation (solution) can be identified by evaluating only the feasible permutations using Eqs. 1 and 5.

In such an attempt to find the true optimal solutions, a small illustrated cSRFLP instance of ten facilities is investigated here. The assumed length vector of the facilities and the material handling costs among them are given in Eq. 6.

$$\left.\begin{array}{l} \text{Length vector, } l = \{6, 4, 8, 5, 7, 3, 1, 2, 10, 9\} \\ \text{Material handling cost matrix, } c = \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 1 & 0 & 0 & 0 & 2 \\ & 0 & 3 & 7 & 7 & 2 & 4 & 5 & 2 & 4 \\ & & 0 & 4 & 4 & 2 & 2 & 3 & 1 & 1 \\ & & & 0 & 2 & 4 & 2 & 2 & 4 & 2 \\ & & & & 0 & 3 & 3 & 2 & 4 & 4 \\ & & & & & 0 & 3 & 3 & 1 & 1 \\ & & & & & & 0 & 2 & 2 & 1 \\ & & & & & & & 0 & 2 & 1 \\ & & & \multicolumn{3}{c}{c_{ij} = c_{ji}} & & & 0 & 2 \\ & & & & & & & & & 0 \end{bmatrix} \end{array}\right\} \tag{6}$$

Table 1 Constraints imposed to the illustrative cSRFLP instance of ten facilities ("Y" means "yes" and "N" means "no")

Facility (i)	1	2	3	4	5	6	7	8	9	10
Whether to be placed in fixed position (T_i)	N	N	Y	Y	N	N	N	N	N	N
Specified fixed position (P_i)	–	–	8	7	–	–	–	–	–	–
Whether to be ordered on left side (O_i)	N	Y	N	N	Y	Y	N	Y	N	Y
Number of facilities to be placed on right side (M_i)	–	1	–	–	1	1	–	1	–	1
Ordered facility on right side (R_{ik})	–	9	–	–	9	5	–	7	–	6
Other facilities allowed within the pair (A_{ik})	–	Y	–	–	Y	Y	–	N	–	N

Consider that facilities 3 and 4 of the illustrative instance are to be placed in the eighth and seventh positions, respectively. Also, facility 2 should be placed before facility 9, facility 5 is before facility 9, and facility 6 is before facility 5. Similarly, facility 8 should be placed before facility 7 and facility 10 before facility 6, but each of these two pairs should come together. Speaking in terms of the constraints imposed to the cSRFLP model as stated above, facilities 3 and 4 are to be placed in fixed positions, the pair of facilities 8 and 7 as well as that of facilities 10 and 6 are to be ordered without allowing any other facilities in between a pair, while the pair of facilities 2 and 9 as well as those of facilities 5 and 9 and facilities 6 and 5 are to be ordered allowing the placement of other facilities in between a pair (in fact, all the facilities of the instance are constrained except facility 1). For convenience in the implementation, all the constraints imposed to the instance are arranged in Table 1 in a matrix form in terms of the notations T_i through A_{ik} defined in Section 2.1. The value "Y" (yes) to T_i means that facility i is to be placed at a fixed position, which is specified by P_i. Similarly, values to M_i, R_{ik}, and A_{ik} are required for that facility only, which is paired (i.e., O_i = Y) with other facilities for arranging on its right side.

The illustrative instance, having the length vector and material handling cost matrix given by Eq. 6 and the constraints given in Table 1, is investigated here under five cases—without any constraint, with positioning constraints (Eq. 2) only, with positioning and ordering constraints allowing placement of other facilities in between an ordered pair of facilities (Eqs. 2 and 4), with positioning and ordering constraints without allowing placement of other facilities in between an ordered pair of facilities (Eqs. 2 and 3), and with positioning and ordering constraints both allowing and without allowing the placement of other facilities in between an ordered pair of facilities (Eqs. 2, 3, and 4). The detail results obtained in this exhaustive search based investigation are given in Table 2. Further, the optimum permutation (solution) of the fifth case of Table 2 is shown in Fig. 1, where the constrained facilities are shown in cyan color and the unconstrained one in gray color.

It is observed in Table 2 how drastically the number of feasible permutations as well as the required computational time for evaluating the functions reduce with the increasing number and/or complexity of constraints imposed to a cSRFLP instance. It is also observed that the optimum objective value (i.e., the minimum material handling cost) in a constrained solution (i.e., a solution of the cSRFLP model) is likely to be higher than that in an unconstrained solution (i.e., a solution of the SRFLP model). However, the true optimum cSRFLP solutions (SRFLP solutions as well) can be obtained through such an exhaustive search for small-size instances only. This is because of the inadequacy of today's computer to code the permutations of a large string (sizes beyond 15 or so) at a reasonable time. For example, the time required to code all the permutations of 10, 11, 12, 13, 14, and 15 were found to be 0.13 s, 1.35 s, 15.94 s, 208 s ($\approx$ 3.5 min), 2877 s ($\approx$ 48 min) and 42978 s ($\approx$ 12 h), respectively. It depicts how exponentially

Table 2 Results of the exhaustive search for an instance of 10 facilities under the constraints given in Table 1

Case (type of constraints)	Optimum f (Eq. 1)	Optimum permutation	Number of feasible permutations	Computational time (sec)
—	1441.5	9-3-4-6-8-7-2-5-10-1	3628800	175.64
Eq. 2	1483.5	9-5-6-7-8-2-4-3-10-1	40320	2.15
Eqs. 2 and 4	1585.5	1-10-6-5-2-7-4-3-8-9	5040	0.60
Eqs. 2 and 3	1596.5	1-10-6-2-8-7-4-3-5-9	528	0.35
Eqs. 2, 3, and 4	1596.5	1-10-6-2-8-7-4-3-5-9	75	0.33

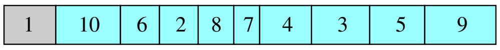

Fig. 1 The optimum solution of the fifth case of Table 2 (constrained facilities are shown in cyan color and others in gray color)

the required time increases in coding the permutations with the increasing size of a string. This is the reason why the implementation of some metaheuristics to such problems is sought for obtaining an approximate optimum at an allowable computational time.

3 The proposed cpGA for solving the cSRFLP model

The genetic algorithm (GA) is a metaheuristic that mimics the mechanisms of the Darwinian evolution based on the concept of the survival of the fittest. It works with a population comprising a set of individuals, where an individual is usually codified as an array (or some other complex forms) of the decision variables of a problem and represents a complete solution of the problem. The population is gradually evolved towards the optima of the problem through the repeated application of some operators analogous to those from natural evolution, namely the selection, crossover, and mutation operators. A selection operator is engaged for identifying the better individuals in the current population, and then a crossover operator generates some new individuals by exploiting the better individuals identified by the selection operator. Finally, a mutation operator explores the neighborhood of the new individuals generated by the crossover operator. In this way, the process of evolution of the population is continued until some termination criteria are met, usually until a predefined maximum number of generations are performed.

Initially, the GA was considered to be a problem-independent algorithm, i.e., it could be applied as a general algorithm to different classes of problems. It was realized sooner that all problem-specific complexities cannot be resolved without intelligence, implying that the incorporation of certain problem-specific components in the GA (as well in other evolutionary algorithms) is necessary in order to compete with other problem-specific classical optimization methods, what was termed by Wolpert and Macready [42] as the *no free lunch theorems for optimization*. Combinatorial problems, including the facility layout problem studied in the present work, belong to that category which cannot be solved effectively without incorporating problem-specific information in an optimizer. Further, as shown in Section 2.2 that the size of the search space (i.e., the number of feasible solutions) of a cSRFLP instance decreases drastically with the increasing number/complexity of constraints, a general-purpose metaheuristic may not be able to explore such limited number of feasible solutions. Accordingly, a novel GA, named as the *constrained permutation-based GA* (cpGA in short), is proposed here by incorporating the constraint-information of the cSRFLP model in the general GA, so as to generate only feasible solutions by limiting the GA search in the feasible region of the problem. The cpGA is explained in detail in Sections 3.1–3.5.

3.1 Individual representation and population initialization

An individual of the cpGA is taken as an array to represent a permutation of a string (i.e., the number of facilities to be arranged). The value of an element of the array represents a facility with the index of the element as the position of the facility in the permutation. Unlike in many real or integer-valued problems, a random initialization of an individual is likely to generate an invalid permutation by repeating many facilities, while omitting some others. Further, even if a valid permutation is generated, it might not be a feasible solution of the cSRFLP. Therefore, a heuristic is proposed here for initializing an individual with a feasible solution of the cSRFLP by satisfying the positioning and ordering constraints expressed by Eqs. 2–4. The step-by-step working procedure of the heuristic, based on downward complexities for initializing an individual, is explained below:

(1) *Placement of the fixed-positioned facilities*: Place all the fixed-positioned facilities, if any, in their respective specified positions in the cpGA individual.

(2) *Placement of the ordered facilities*: For placing the ordered facilities in the cpGA individual, first arrange the facilities in descending order of number of other facilities to be placed on their right side. Then place the facilities one by one in randomly selected vacant elements of the cpGA individual, leaving the minimum number of vacant elements on the right side of a given element for placing those facilities which should come on its right side. For example, there are q number of facilities to be placed on the right side of facility i and that number of vacant elements in the cpGA individual are available after its vth element ($n - v \geqslant q$, where n is the total number of elements of the cpGA individual). Then facility i will be placed in a randomly selected vacant element in the range of $(1, v)$. Further, for two ordered facilities without allowing any other facilities in between them, two consecutive vacant elements are to be obtained in the cpGA individual, if required by sliding some already placed facilities towards left or right without violating their any constraint requirement.

The above technique for placing ordered facilities is illustrated with the help of the instance of Fig. 1, whose facilities are arranged in Table 1 with respect to the imposed positioning and ordering constraints. As shown in Fig. 2a, in the first chain, there are a total of four facilities (6, 5, 2, and 9) on the right side of facility 10, three facilities (5, 2, and 9) on the right side of facility 6, and two facilities (2 and 9) on the right side of facility 5. There are only two facilities in the second chain shown in Fig. 2a, requiring facility 7 to be placed on the right side of facility 8. In Fig. 2a, facilities 10 and 6, as well as facilities 8 and 7, are shown jointly as these are to be placed together without allowing any other facilities in between them. Further, facilities 2 and 9 are shown disconnected as their placements are independent of each other. Now coming to the cpGA individual (the top version in Fig. 2b), its elements 7 and 8 are already filled with the fixed-positioned facilities 4 and 3, respectively. Hence, an element for placing facility 10, leaving a minimum of four vacant elements on its right side for the four facilities in the chain of this facility, is to be searched in the range of [1, 4]. Further, the immediate next element is also to be vacant for placing facility 6, which is paired with facility 10 without allowing any other facilities in between them. Say the random search selects element 2 for facility 10. Accordingly, facilities 10 and 6 are placed in elements 2 and 3, respectively. Since facility 6 is already placed, the search is continued in a similar way for selecting suitable elements for the remaining 3 facilities (5, 2, and 9) of this chain (the placement of all the five facilities of this chain is illustrated in Fig. 2b by the middle version of the cpGA individual). In the case of facilities 8 and 7 lying in the second chain in Fig. 2a, two consecutive vacant elements are required, which are not available in the present (middle) version of the cpGA individual shown in Fig. 2b. Hence, two such elements are to be obtained by sliding some already placed facilities towards left or right, the possible options for which include the sliding either of facilities 10 and 6 through one element towards left (vacating elements 3 and 4) or right (vacating elements 1 and 2), or the sliding of facility 5 through one element towards left (vacating elements 5 and 6) or right (vacating elements 4 and 5). Say, as shown in the bottom version of the cpGA individual in Fig. 2b, the search selects facilities 10 and 6 to slide towards left vacating elements 3 and 4 for facilities 8 and 7, respectively.

(3) *Placement of the unconstrained facilities*: It is obvious that all the left out unplaced facilities are unconstrained and their number is equal to the number of existing vacant elements of the cpGA individual. Therefore, these unconstrained facilities, if any, can be assigned randomly to the vacant elements of the cpGA individual. For this, all the left out facilities are first arranged serially in a temporary array. Then, the vacant elements of the cpGA individual are filled up one by one with the facilities taken from the randomly selected elements of the temporary array. Each time a facility is transferred to the cpGA individual, the temporary array is updated by eliminating the element of that facility. The process is continued until all the facilities of the temporary array are exhausted.

3.2 Selection operator

The binary tournament selection operator [8] is used for forming a mating pool with above-average individuals of the cpGA population (the crossover operator explained in Section 3.3 will generate new individuals by exploiting the mating pool). This selection operator takes two random individuals at a time from the cpGA population and stores in the mating pool a copy of the best one based on their objective values evaluated using Eq. 1. The process is repeated until the size of the mating pool equals that of the cpGA population.

3.3 Crossover operator

The problem-specific crossover operator of the cpGA, developed here for the cSRFLP model, generates new

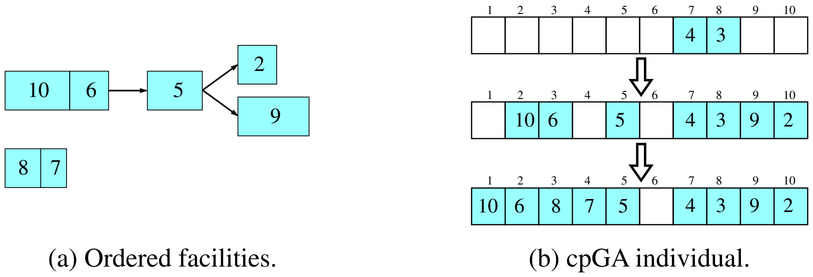

Fig. 2 Facilities of the instance of Fig. 1 in descending order of number of facilities to be arranged on the right side of a given facility (ordering constraints). **a** Ordered facilities. **b** cpGA individual

feasible individuals (i.e, solutions of the cSRFLP) by exploiting parent individuals from the mating pool generated by the binary tournament selection operator. It first chooses two random parent individuals from the mating pool and then a new individual is generated by crossing them with a predefined crossover probability as follows:

(1) *Placement of the fixed-positioned facilities*: The fixed-positioned facilities, if any, are placed in the new cpGA individual exactly in the same way as explained in Step (1) of Section 3.1.
(2) *Placement of the ordered facilities*: Positions of each pair of ordered facilities are checked in both the parent individuals if the facilities can be placed in the positions of the new individual corresponding to those of any of the two parent individuals, taking care of two cases: (i) the placement of two ordered facilities without allowing any other facilities in between them and (ii) the availability of minimum number of vacant positions on the right side of a facility for placing other facilities in its chain as illustrated in Fig. 2a. In the case where a pair of facilities can be placed in the new individual corresponding to both the parent individuals, the positions corresponding to one of the randomly selected parent individual are considered. If any further pair of ordered facilities is left, which cannot be placed according to any of the two parent individuals, then those are placed randomly in the vacant elements of the new individual exactly in the same way as explained in Step (2) of Section 3.1.
(3) *Placement of the unconstrained facilities*: It is obvious in this case also that all the left out unplaced facilities are unconstrained and their number is equal to the number of existing vacant elements of the cpGA individual. Positions of each of these facilities are checked in both the parent individuals if the facility can be placed in any of the corresponding position of the new individual. In the case where both the corresponding positions in the new individual are vacant, the facility is placed randomly in one of them. If any further facility is left, which cannot be placed in the same position(s) of any of the two parent individuals, then those are placed randomly in the vacant elements of the new individual exactly in the same way as explained in Step (3) of Section 3.1.

3.4 Mutation operator

The mutation operator of the proposed cpGA is a very simple one. It chooses two random elements of a new individual, generated by the crossover operator, with a low mutation probability. Then, the values of those two elements are simply swapped, provided no positioning or ordering constraint of the cSRFLP is violated, i.e., ensuring the formation of another feasible individual. Since no value is altered, but two elemental values are interchanged only, the operator successfully explores a feasible neighborhood of a given individual.

3.5 Elite preserving mechanism

Although expected, the random crossover and mutation operators of a genetic algorithm may fail to generate better individuals at some generations, or may even push the search away from the optimum of a problem by generating individuals worse than those of the current population. In order to prevent such a situation, generally the elite individuals of the population are preserved over generations. For this purpose, the elite preserving mechanism proposed by Deb et al. [9] is used in the cpGA, which first combines the old and new populations of a generation, and then the population for the next generation is formed by the best 50% of the combined individuals based on their objective values evaluated through (1).

4 Computational experiment

The cpGA explained in Section 3 is coded in C programming language implementing the cSRFLP model formulated in Eqs. 1–5 and it is executed in Fedora 15 Linux environment. The numerical experimentation is performed in two parts. Firstly, three case studies of the operation

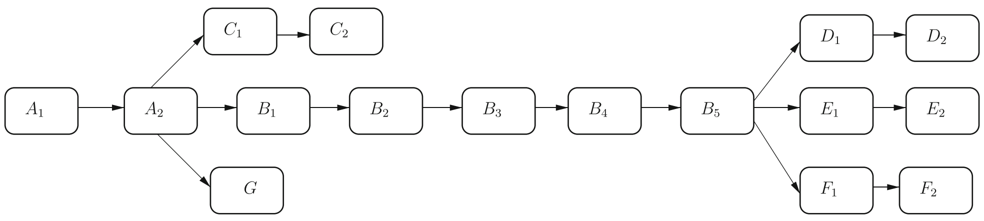
$$\begin{tikzcd}
 & C_1 \arrow[r] & C_2 & & & & D_1 \arrow[r] & D_2 \\
A_1 \arrow[r] & A_2 \arrow[r] \arrow[u] \arrow[d] & B_1 \arrow[r] & B_2 \arrow[r] & B_3 \arrow[r] & B_4 \arrow[r] & B_5 \arrow[r] \arrow[u] \arrow[d] & E_1 \arrow[r] & E_2 \\
 & G & & & & & F_1 \arrow[r] & F_2
\end{tikzcd}$$

Fig. 3 Operation precedence graph for the first case study of Section 4.1

Table 3 Precedence cost matrix for the first case study of Section 4.1

	Succeeding operations															
	A_1	A_2	B_1	B_2	B_3	B_4	B_5	C_1	C_2	D_1	D_2	E_1	E_2	F_1	F_2	G
A_1	0	1	999	999	999	999	999	999	999	999	999	999	999	999	999	999
A_2	999	0	10	999	999	999	999	999	999	999	999	999	999	999	999	999
B_1	999	999	0	2	999	999	999	2	999	999	999	999	999	999	999	2
B_2	999	999	999	0	2	999	999	2	999	999	999	999	999	999	999	2
B_3	999	999	999	999	0	2	999	999	999	999	999	999	999	999	999	999
B_4	999	999	999	999	999	0	1	999	999	999	999	999	999	999	999	999
B_5	999	999	999	999	999	999	0	2	999	2	1	2	999	999	999	2
C_1	999	999	2	2	999	999	999	0	2	2	999	999	999	999	999	2
C_2	999	999	999	999	999	999	999	999	0	2	999	999	999	999	999	2
D_1	999	999	999	999	999	999	999	2	999	0	2	2	999	999	999	2
D_2	999	999	999	999	999	999	999	999	999	999	0	2	2	2	999	999
E_1	999	999	999	999	999	999	999	999	999	2	999	0	2	2	999	999
E_2	999	999	999	999	999	999	999	999	999	999	2	999	0	2	2	2
F_1	999	999	999	999	999	999	999	999	999	999	2	2	2	0	2	2
F_2	999	999	999	999	999	999	999	999	999	999	999	2	2	999	0	2
G	999	999	2	2	999	999	999	2	2	2	999	999	999	2	2	0

sequencing problem in process planning are solved as special cases of cSRFLP, and then some large-size instances as general cSRFLP.

A well-established fact about metaheuristics, including GA, is that their performance may get influenced by the algorithmic parameter values. Hence, researchers sometime attempt to tune algorithmic parameter values by applying some transition rules, so as to tackle the problem at hand effectively using the obtained such values [4, 5, 7, 15, 30]. However, the prime aim of the present work is not to investigate how the performance of an algorithm can be improved, but to introduce an effective procedure for handling the proposed cSRFLP problem successfully. Accordingly, skipping the issue of fine tuning the parameter values for the cpGA, those based on some trial runs are fixed as follows: the population size is 60, crossover probability is 75%, mutation probability is 1%, and the maximum number of generations to be performed is 500. Using these parameter values, 30 independent runs of the cpGA are performed for each problem instance with different sets of initial solutions.

4.1 Operation sequencing in process planning

It is stated in Section 1 that operation sequencing in process planning is a practical manufacturing problem which can be handled as a special case of the proposed cSRFLP model with ordering (i.e., precedence) constraints only. Hence, three such case studies are taken from literature for demonstrating the application of the cSRFLP model. Since the operation sequencing problem seeks the manufacturing operations to be sequenced in a way to minimize the total machining cost arising from the relative machining cost between each pair of successive operations, the objective

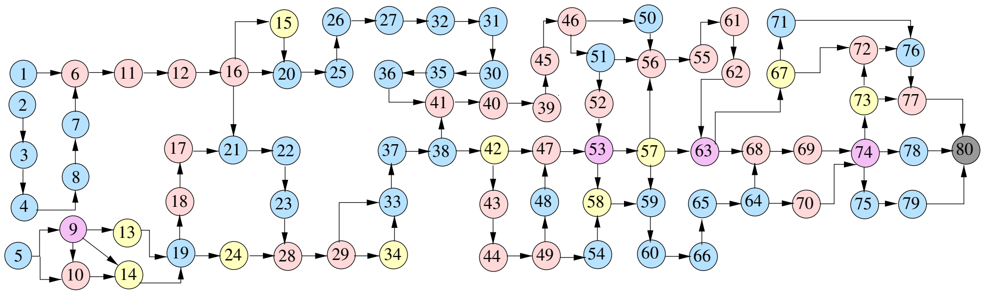

Fig. 4 Operation precedence graph for the third case study (having size of 80) of Section 4.1

Table 4 Comparison of the best results for the first case study of Section 4.1

Result	Ref. [19]	Ref. [29]	Present work
Machining cost	36	36	36
Number of function evaluation	–	–	3840
Computation time (in seconds)	18	7	0.442
Number of optimal sequences	1	6	2

function of the general cSRFLP model given by Eq. 1 can be modified for the operation sequencing problem as expressed by Eq. 7.

$$\text{Minimize} \qquad f^{\pi} \equiv \sum_{i=1}^{n-1} c_{r_i^{\pi}, r_{i+1}^{\pi}} \tag{7}$$

The first case study, introduced by Krishna and Rao [19], is a hand spike whose different features are prepared by 16 machining operations. These operations are rough plain milling (A_1), finish plain milling (A_2), milling the bose (B_1), drilling (B_2), boring (B_3), reaming (B_4), finish reaming (B_5), drilling (C_1), milling (C_2), drilling (D_1, E_1, F_1), tapping (D_2, E_2, F_2) and drilling (G). The precedence relations among the operations are shown in Fig. 3, where an operation connected with another operation by an arrow denote that the latter precedes the former, e.g., A_1 precedes A_2 and A_2 precedes C_1. The directed relative costs between each pair of the machining operations are shown in Table 3 which is to be read row-wise first and then column-wise, e.g., the relative machining cost between A_1 and A_2 is 1 if A_1 precedes A_2 and that between them is 999 if A_2 precedes A_1.

The other two case studies of the sequencing problem of sizes 70 and 80 are taken from Yun et al. [43]. In order to demonstrate their complexity, the precedence relations among the operations of the case study of size 80 are shown in Fig. 4 in the same way as in Fig. 3, with the only difference that the operations in Fig. 4 are denoted by 1,2,. . . ,80 (refer Yun et al. [43] for other detail of these two case studies).

For the first case study, the cpGA could find two solutions having the already reported best objective value (i.e., the total machining cost of 36) just in 46 generations taking 0.442s of computational time (note that the computational time is not a good measure for comparison as it would vary significantly with specifications of computers as well as programming skills also). Further, although the same best-known objective value is obtained, the corresponding sequences of operations are different from that reported by Nallakumarasamy et al. [29]. The comparison of the obtained best solutions and the corresponding new sequences of operations are reported in Tables 4 and 5, respectively.

Table 6 Comparison of the best objective values for the case studies of sizes 70 and 80 investigated in Section 4.1

SN	No. of operations	Best objective value (machining cost)			
		Ref. [44]	Ref. [43]	Ref. [26]	Present work
1	70	392	384	314	*312*
2	80	426	423	366	*363*

On the other hand, the cpGA could improve the best-known objective values of the investigated other two case studies of operation sequencing, i.e., the instances of sizes 70 and 80. Comparison of the obtained best objective values is presented in Table 6 marking the so far known best values in italics, and the sequences of operations corresponding to the best objective values obtained in the present work are shown in Table 7.

4.2 Benchmark instances

After solving three operation sequencing instances in Section 4.1 as special cases of cSRFLP, a set of 5 SRFLP instances each of size 100, introduced by Anjos and Yen [3], are studied in order to demonstrate the potentiality of the proposed general cSRFLP procedure on large-size instances with all types of constraints as explained in Section 2.1. For convenient and easy recognition, the constraints are imposed to the same facilities of all the instances. In the case of the positioning constraints, two facilities, namely facilities 1 and 60, are restricted to be placed at the first and sixtieth positions, respectively. Similarly, the ordering constraints are imposed to two pairs of facilities, seeking the pair of facilities 20 and 30 to be ordered without allowing any other facilities in between them, while the second pair of facilities 25 and 10 to be ordered allowing the placement of other facilities in between them.

The best results obtained in terms of the objective function values (i.e., the overall material handling cost among the facilities) for the considered problem instances

Table 5 The obtained best sequences of operations for the first case study of Section 4.1

SN	Sequence of operations	Machining cost
1	$A_1 \rightarrow A_2 \rightarrow B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_5 \rightarrow C_1 \rightarrow D_1 \rightarrow E_1 \rightarrow F_1 \rightarrow D_2 \rightarrow E_2 \rightarrow F_2 \rightarrow G \rightarrow C_2$	36
2	$A_1 \rightarrow A_2 \rightarrow B_1 \rightarrow C_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_5 \rightarrow D_1 \rightarrow E_1 \rightarrow F_1 \rightarrow D_2 \rightarrow E_2 \rightarrow F_2 \rightarrow G \rightarrow C_2$	36

Table 7 The obtained best sequences of operations for the case studies of sizes 70 and 80 investigated in Section 4.1

SN	No. of operations	Sequence of operations	Machining cost
1	70	3 → 4 → 1 → 2 → 7 → 6 → 8 → 5 → 9 → 13 → 17 → 12 → 10 → 11 → 14 → 18 → 15 → 23 → 28 → 16 → 22 → 21 → 20 → 26 → 27 → 19 → 25 → 24 → 29 → 30 → 31 → 32 → 33 → 35 → 34 → 39 → 38 → 37 → 36 → 40 → 41 → 47 → 42 → 46 → 45 → 43 → 44 → 48 → 50 → 49 → 55 → 54 → 61 → 51 → 60 → 52 → 53 → 59 → 58 → 64 → 63 → 68 → 57 → 65 → 56 → 62 → 66 → 69 → 67 → 70	312
2	80	2 → 3 → 4 → 8 → 1 → 7 → 6 → 5 → 9 → 13 → 11 → 10 → 12 → 14 → 19 → 16 → 15 → 18 → 17 → 20 → 24 → 25 → 26 → 27 → 21 → 32 → 22 → 31 → 30 → 23 → 28 → 29 → 34 → 33 → 35 → 37 → 36 → 38 → 42 → 43 → 41 → 40 → 44 → 39 → 45 → 46 → 49 → 54 → 51 → 48 → 47 → 52 → 50 → 53 → 58 → 57 → 59 → 56 → 60 → 66 → 55 → 61 → 62 → 65 → 64 → 63 → 68 → 70 → 69 → 74 → 67 → 73 → 75 → 72 → 79 → 71 → 78 → 76 → 77 → 80	363

are shown in Table 8, where the constrained facilities are shown in italic fonts.

Since it is the introductory work on constrained SRFLP (i.e., cSRFLP), the results presented in Table 8 could not be compared with any existing similar work. Therefore, the best known unconstrained SRFLP objective values of the instances, as reported by Kothari and Ghosh [18], are also shown in Table 8 for comparing with the same obtained for the cSRFLP model. It is observed that the overall material handling costs (i.e., the objective function values) of the cSRFLP instances are marginally inferior to those of the SRFLP instances, which is obvious due to the presence of the considered positioning and ordering constraints in the cSRFLP model.

Table 8 The best cSRFLP results obtained for the 5 instances of Anjos and Yen [3] and comparison of their objective values (overall material handling costs) with the best known corresponding SRFLP values reported by Kothari and Ghosh [18]

ID	Instance	Material handling cost			Permutation of the facilities (cSRFLP model)
		SRFLP	cSRFLP	% increase	
1	100-01	378234.0	378814.0	0.2	*1* 65 78 18 97 37 71 83 39 79 91 33 15 6 11 95 74 55 *25* 13 5 63 77 57 27 28 31 54 75 80 34 2 14 43 49 52 16 88 22 32 62 9 89 94 84 19 86 87 68 59 98 8 4 96 56 93 42 69 24 *60* 67 85 99 *20 30* 73 40 81 26 51 46 64 90 44 47 100 70 66 72 38 92 82 36 61 41 48 50 23 12 21 17 7 53 35 29 76 58 *10* 45 3
2	100-02	2076008.5	2104507.5	1.4	*1* 78 33 91 97 99 27 18 5 88 32 77 *25* 71 94 95 11 54 31 22 64 *20 30* 85 13 43 63 14 81 4 79 83 52 16 67 46 57 34 55 74 6 80 75 37 15 28 49 2 89 19 44 90 86 23 82 56 51 87 21 *60* 98 47 70 45 96 66 69 92 100 8 84 59 38 72 7 35 62 9 68 3 36 40 26 73 50 41 48 42 24 93 76 61 12 39 53 *10* 65 17 29 58
3	100-03	16145614.5	16527157.5	2.4	**1** 97 78 39 28 89 49 33 80 54 24 21 5 91 31 71 37 13 83 2 16 11 65 63 27 14 94 43 85 75 73 9 67 52 18 *25 10* 95 55 15 6 77 79 74 57 34 84 93 22 32 88 *20 30* 81 42 98 96 17 4 *60* 26 46 59 64 7 12 23 56 50 62 99 36 51 38 72 48 19 86 87 41 70 90 29 68 69 66 40 100 82 61 47 53 92 45 76 58 8 44 35 3
4	100-04	3232522.0	3291610.0	1.8	*1* 39 71 49 79 42 *25* 32 5 97 18 93 94 52 83 68 8 98 9 16 22 88 33 43 21 27 75 80 24 67 31 19 74 86 28 54 89 15 56 59 37 65 96 85 4 91 57 63 55 13 78 62 11 77 61 50 81 92 48 *60* 90 100 38 46 26 82 69 53 35 72 66 70 36 51 10 40 *20 30* 47 73 14 41 87 34 64 95 44 29 23 12 17 99 2 76 58 45 84 6 3 7
5	100-05	1033080.5	1056445.5	2.3	*1* 78 90 55 63 *25* 13 58 6 *10* 65 84 74 54 28 75 15 24 80 37 71 39 83 11 31 95 27 19 77 79 43 22 89 94 14 5 33 91 57 18 97 16 9 2 88 34 32 49 93 42 81 68 46 40 26 73 67 87 41 *60* 64 44 85 99 *20 30* 66 70 36 56 82 100 53 35 21 8 4 29 96 7 98 47 61 59 51 69 62 12 23 17 52 38 92 86 72 45 48 50 3 76

5 Conclusion

Although the single-row facility layout problem (SRFLP) is studied as an unconstrained design optimization problem by placing given facilities along a straight row without any restriction, a practical problem may be constrained to place some facilities in specified fixed positions and/or orders. Accordingly, a constrained SRFLP (cSRFLP) model is introduced here by imposing positioning and ordering constraints to some facilities. Owing the difficulties for any general algorithm to come out of such constraints, a novel constrained permutation-based genetic algorithm is also proposed with problem-specific operators for exploring only the feasible space of the cSRFLP. Applying the proposed procedure in the numerical experimentation, three case studies of operation sequencing in process planning are studied first as special cases of the cSRFLP model with ordering constraints only, and then some large-size instances of the general cSRFLP model are presented which could serve as benchmark instances in future work on the cSRFLP. In the case of the operation sequencing problem, new sequences of operations having the already known best objective values are obtained for the first case study, while the best-known solutions for the other two case studies could be improved also.

Since genetic algorithm is a well-established population-based metaheuristic for decades, its customization is presented here for solving the proposed cSRFLP model without going through its merits and demerits over other similar techniques. However, other appropriate techniques may also be attempted for solving the problem. Further, practical application of the introduced cSRLFP model may be investigated by applying it to more real-life problems, such as process and assembly operation sequencing problems. Moreover, apart from the material handling cost, other classical criteria, such as machining time, transportation time, quality, etc., may also be considered simultaneously.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Aiello G, Scalia GL, Enea M (2012) A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding. Expert Syst Appl 39:10352–10358
2. Anjos MF, Vannelli A (2008) Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. INFORMS J Comput 20:611–617
3. Anjos MF, Yen G (2009) Provably near-optimal solutions for very large single-row facility layout problems. Optim Method Softw 24(4):805–817
4. Ayad AR, Awad HA, Yassin A (2013) Parametric analysis for genetic algorithms handling parameters. Alexandria Eng J 52:99–111
5. Branke J (2012) Evolutionary optimization in dynamic environments. Hardcover, ISBN: 978-0-7923-7631-6. Springer, US
6. Datta D, Amaral ARS, Figueira JR (2011) Single row facility layout problem using a permutation-based genetic algorithm. Eur J Oper Res 213(2):388–394
7. Datta D, Figueira JR (2011) Graph partitioning by multi-objective real-valued metaheuristics: a comparative study. Appl Soft Comput 11(5):3976–3987
8. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
9. Deb K, Agarwal S, Pratap A, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
10. Drira A, Pierreval H, Hajri-Gabouj S (2007) Facility layout problems: a survey. Annu Rev Control 31:255–267
11. Heragu SS, Alfa AS (1992) Experiment analysis of simulated annealing based algorithms for the layout problem. Eur J Oper Res 57(2):190–202
12. Heragu SS, Kusiak A (1988) Machine layout problem in flexible manufacturing systems. Oper Res 36:258–268
13. Huang J, Lu X, Zhang G, Qu J (2014) Study on the rheological, thermal and mechanical properties of thermoplastic polyurethane/poly (butylene terephthalate) blends. Polym Test 36:69–74
14. Huang W, Lin W, Xu S (2017) Application of graph theory and hybrid GA-SA for operation sequencing in a dynamic workshop environment. Comput-Aid Des Appl 14(2):148–159
15. Javadi G, Aminian E (2017) A new method for tuning mutation and crossover rate in genetic algorithm. In: Proceedings of the 9th International Conference on Machine Learning and Computing, ICMLC 2017. ACM, New York, pp 217–220
16. Kafashi S (2011) Integrated setup planning and operation sequencing (ISOS) using genetic algorithm. Int J Adv Manuf Technol 56(5):589–600
17. Kothari R, Ghosh D (2013) Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. Eur J Oper Res 224(1):93–100
18. Kothari R, Ghosh D (2014) A scatter search algorithm for the single row facility layout problem. J Heuristics 20(2):125–142
19. Krishna AG, Rao K (2006) Optimisation of operations sequence in CAPP using an ant colony algorithm. Int J Adv Manuf Technol 29(1–2):159–164
20. Kumar KR, Hadjinicola GC, Lin TL (1995) A heuristic procedure for the single-row facility layout problem. Eur J Oper Res 87:65–73
21. Kumar S, Asokan P, Kumanan S, Varma B (2008) Scatter search algorithm for single row layout problem in FMS. Advan Production Eng Manag 3:193–204
22. Lafou M, Mathieu L, Pois S, Alochet M (2016) Manufacturing system flexibility: sequence flexibility assessment. In: 49th CIRP Conference on Manufacturing System (CIRP CMS2016)
23. Li X, Gao L, Wen X (2013) Application of an efficient modified particle swarm optimization algorithm for process planning. Int J Adv Manuf Technol 67(5–8):1355–1369
24. Love RF, Wong JY (1976) On solving a single row space allocation problem with integer programming. INFOR 14:139–143
25. Luo G, Wen X, Li H, Ming W, Xie G (2017) An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning

and scheduling. Int J Adv Manuf Technol 91(9–12):3145–3158
26. Maadi M, Javidnia M, Ramezani R (2017) Modified Cuckoo Optimization Algorithm (MCOA) to solve Precedence Constrained Sequencing Problem (PCSP). Applied Intelligence. https://doi.org/10.1007/s10489-017-1022-0
27. Miljković Z, Petrović M (2017) Application of modified multi-objective particle swarm optimisation algorithm for flexible process planning problem. Int J Comput Integr Manuf 30(2–3):271–291
28. Müller J (2016) MISO: Mixed-integer surrogate optimization framework. Optim Eng 17(1):177–203
29. Nallakumarasamy G, Srinivasan PSS, Raja KV, Malayalamurthi R (2011) Optimization of operation sequencing in CAPP using simulated annealing technique (SAT). Int J Adv Manuf Technol 54(5–8):721–728
30. Nguyen TT, Yang S, Branke J (2012) Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evolutionary Comput 6:1–24
31. Ozgormus E (2015) Optimization of block layout for grocery stores. PhD thesis. Auburn University, USA
32. Padgaonkar AS (2004) Modelling and analysis of the hospital facility layout problem Master's thesis. Department of Industrial and Manufacturing Engineering, New Jersey Institute of Technology
33. Petrović M, Mitić M, Vuković N, Miljković Z (2016) Chaotic particle swarm optimization algorithm for flexible process planning. Int J Adv Manuf Technol 85(9–12):2535–2555
34. Picard J, Queyranne M (1981) On the one dimensional space allocation problem. Oper Res 29(2):371–391
35. Reddy SB (1999) Operation sequencing in CAPP using genetic algorithms. Int J Prod Res 37(5):1063–1074
36. Samarghandi H, Eshghi K (2010) An efficient tabu algorithm for the single row facility layout problem. Eur J Oper Res 205:98–105
37. Samarghandi H, Taabayan P, Jahantigh FF (2010) A particle swarm optimization for the single row facility layout problem. Comput Industrial Eng 58:529–534
38. Simmons DM (1969) Single row space allocation: an ordering algorithm. Oper Res 17(5):812–826
39. Solimanpur M, Vrat P, Shankar R (2005) An ant algorithm for the single row layout problem in flexible manufacturing systems. Comput Oper Res 32:583–598
40. Sugden SJ (1992) A class of direct search methods for nonlinear integer programming. PhD Thesis, Bond University, Australia
41. Wang W, Li Y, Huang L (2016) Rule and branch-and-bound algorithm based sequencing of machining features for process planning of complex parts. Journal of Intelligent Manufacturing. https://doi.org/10.1007/s10845-015-1181-y
42. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
43. Yun Y, Chung H, Moon C (2013) Hybrid genetic algorithm approach for precedence-constrained sequencing problem. Comput Industr Eng 65(1):137–147
44. Yun Y, Moon C (2011) Genetic algorithm approach for precedence-constrained sequencing problems. J Intell Manuf 22(3):379–388