Research Article

International Journal of Advanced Technology and Engineering Exploration, Vol 11(115)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2023.10102424

Optimizing dynamic facility layout problems: genetic algorithm with local search integration

Vineetha G. R[1*] and Shiyas C. R[2]
Research Scholar, Department of Computer Science and Engineering, Cochin University College of Engineering, Kuttanad, Pulincunnu, Kerala, India[1]
Associate Professor, Department of Mechanical Engineering, Cochin University College of Engineering, Kuttanad, Pulincunnu, Kerala, India[2]

Received: 15-October-2023; Revised: 18-June-2024; Accepted: 21-June-2024

Abstract
The dynamic facility layout problem (DFLP) is one of the most complex combinatorial optimization challenges. Given that obtaining optimal solutions using exact methods requires substantial time and computational power, researchers often turn to nonconventional optimization techniques to achieve near-optimal solutions. This paper presents a genetic algorithm (GA) enhanced with a local search (LS) procedure for solving DFLPs. The algorithm employs roulette wheel selection (RWS), single-point crossover (SPC), and swap mutation (SM) as its genetic operators, with the 2opt neighborhood search serving as the LS operator. The termination criterion (TC) used in the proposed algorithm is the maximum number of generations. An extensive evaluation of the algorithm's performance was conducted in this research. It was tested on a diverse set of 48 problem instances, representing various problem sizes. To assess the effectiveness of the algorithm, the results produced were compared with those documented in existing literature and benchmarked against the best-known solutions previously reported. This rigorous comparison allows for an evaluation of the algorithm's performance relative to other established methods and state-of-the-art solutions available in the field. Through extensive experimentation on 48 test instances, the algorithm consistently delivers competitive results, achieving solutions within a margin of less than four percent deviation from the best-known solutions across all instances, with an average deviation ranging from 0% to 3.71%. Although the average runtime of the algorithm is provided, its comparison with existing literature is deemed irrelevant due to significant variations in machine configurations. This work introduces a hybrid genetic algorithm (hGA) specifically designed for solving DFLPs. By integrating fundamental genetic operations with a localized search approach, the proposed hGA demonstrates promising capabilities in tackling this complex optimization problem. The outcomes affirm the efficacy of the hGA in swiftly converging to near-optimal solutions for DFLPs, underscoring its potential for practical applications.

Keywords
Dynamic facility layout problem, Hybrid genetic algorithm, Local search, Roulette wheel selection, Swap mutation.

1.Introduction
The facility layout problem (FLP) revolves around the optimization of the arrangement of various elements within a manufacturing or distribution system. These elements can include machinery, production departments, and storage areas like warehouses. The primary objective in solving the FLP is to minimize the total material handling cost (MHC) involved in the system operation. MHCs encompasses various expenses related to moving materials or products within the facility, such as labour, equipment maintenance, and transportation [1]. MHCs is a significant concern for businesses, as they represent non-value-added expenses, which means they do not directly contribute to the product's value. These costs can account for a substantial portion of a manufacturing system's overall operating expenses, ranging from 20% to 50% [2]. Moreover, in the context of product cost, they can make up a significant share, varying from 15% to 70% [2].

Therefore, optimizing facility layouts to reduce MHCs is critical for enhancing operational efficiency, cost-effectiveness, and overall competitiveness in manufacturing and distribution industries. By implementing an effective layout, 10-30% of this cost can be decreased, while also contributing to the

*Author for correspondence

system's overall efficiency [2]. In the manufacturing industry, the layout of a plant is a decision to be made over time, since changing the layout involves a substantial investment that is not easily recouped [3].

1.1Challenges in dynamic facility layout problem (DFLP)

FLPs encompass a range of challenges related to optimizing the arrangement of elements within a facility. There are two primary categories of these problems based on the nature of the data used to devise the layout. Static facility layout problems (SFLP) are the first category, characterized by fixed schedules and based on flow data that do not change over time [4]. The second category is DFLP, which, for each planning period, has different flow patterns and as a result, may require changes in facility layout [5]. The objective of SFLP is to minimize the total material cost by positioning N different facilities in N possible locations.

SFLP belongs to the realm of combinatorial optimization problems, a field that has received extensive attention for many years. It is frequently represented as a quadratic assignment problem (QAP), initially formulated by Koopmans and Beckmann in 1957 [6]. Sahni and Gonzalez [7] established the NP-hardness of the QAP, demonstrating that obtaining an approximation close to the optimal solution is not achievable in polynomial time. Despite extensive research on the QAP, there is currently no exact algorithm capable of efficiently solving problems with sizes beyond N > 30, as observed by Loiola et al. [8]. QAP is still widely regarded as one of the most difficult combinatorial optimization problems. The DFLP was first defined and addressed by Rosenblatt [9]. The DFLP is an extension of the SFLP in which the layout of the facility under consideration remains the same throughout all the planning periods under consideration. The objective of SFLP is to minimize the total MHC by positioning N different facilities in N possible locations. The SFLP is commonly formulated as QAPs introduced by Koopmans and Beckmann [6]. The QAP is considered one of the most difficult to solve combinatorial optimization problems.

In the case of DFLP, the layout of the facility changes from period to period. Thus, the total cost involved in a DFLP has two components namely, MHC – to transport items between the departments and re-arrangement cost – to rearrange the facilities as the planning periods change [10]. The mathematical representation of the DFLP, initially proposed by Balakrishnan [11] is presented in Equation 1. It's worth mentioning that this particular formulation signifies an extension of the quadratic integer programming (QIP) framework employed in solving the QAP.

Minimize

$$Z = \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl}Y_{tijl} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} C_{tijkl}\,X_{tij}X_{tkl} \quad (1)$$

subject to

$$\sum_{j=1}^{N} X_{tij} = 1, \qquad i = 1,2,\dots,N \text{ and } t = 1,2,\dots,T \quad (2)$$

$$\sum_{i=1}^{N} X_{tij} = 1,\ j = 1,2,\dots,N \text{ and } t = 1,2,\dots,T \quad (3)$$

$$Y_{tijl} = X_{(t-1)ij}X_{til}, i,j,l = 1,2,\dots,N, t = 1,2,\dots,T \quad (4)$$

$$X_{tij} = \{0,1\}\ for\ all\ i,j,t \quad (5)$$

$$Y_{tijl} = \{0,1\}\ for\ all\ i,j,l,t \quad (6)$$

where

m = Number of departments and locations.
T = Number of periods.
A_{tijl} = Cost of shifting department *i* from location *j* to *l* in period *t* (where A_{tijj}= 0).
C_{tijkl} = Cost of material flow between department *i* located at location *j* and *k* located at *l* in period *t*.

$$X_{tij} = \begin{cases} 1 \text{ if department } i \text{ is assigned to location } j \text{ at period } t \\ 0 \text{ otherwise} \end{cases}$$

$$Y_{tijl} = \begin{cases} 1 \text{ if department } i \text{ is shifted from location } j \text{ to } l \text{ at the beginning of period } t \\ 0 \text{ otherwise} \end{cases}$$

The goal of Equation 1 is to reduce the total costs associated with both rearranging departments and the flow of materials between them. Constraint set Equation 2 assures that each location is exclusively assigned to one department during each period, while constraint set Equation 3 guarantees that each location is allocated to precisely one department for each period. Constraint set Equation 4 allows for the inclusion of rearrangement costs when a department is relocated between locations in consecutive periods, merging them with material flow costs. Constraints Equation 5 and Equation 6 enforce the required restrictions on the decision variables.

In this study, a hGA combining GA with local search (LS) is introduced to solve the DFLP. To evaluate the effectiveness of the proposed method, the hGA is applied to a set of 48 benchmark test instances for DFLPs provided by Balakrishnan and Cheng [12].

This application demonstrates the effectiveness of the hGA in optimizing complex layout configurations over time.

The remaining sections of the paper are structured as follows: Section 2 offers an overview of prior research on DFLP. Section 3 explains the fundamental principles of the proposed hGA and the diverse parameters and operators employed in the hGA, section 4 outlines the experimental methodology and in section 5, the results and associated discussions are presented. Finally, section 6 provides the paper's conclusion.

2.Literature review

In the context of DFLP, two primary categories of solutions have emerged: agile (or adaptive) and robust strategies. These approaches differ in their handling of layout changes and are selected based on specific considerations and constraints [13, 14]. The agile approach involves frequently adjusting the facility layout after each period within the planning horizon to accommodate changing flow patterns or operational requirements. This method, while effective in optimizing layouts to suit evolving conditions, typically incurs higher relocation costs. It is preferred when relocation costs are low and the facility can easily adapt to changes [15]. In contrast, the robust approach maintains a static layout throughout the planning horizon, designed to minimize total MHCs across all periods despite changing conditions. This strategy avoids the disruptions of frequent layout changes and is chosen when stability and consistency are crucial, and relocation costs are high. The decision between agile and robust strategies depends on factors such as facility nature, relocation costs, required flexibility, and tolerance for disruptions. Agile strategies are advantageous when relocation costs are low and adaptation needs are high, as they allow for cost-efficient, quick adjustments, and long-term savings. Thus, agile strategies are ideal for scenarios with low relocation costs, short adjustment times, and uncertain future planning periods, enabling facilities to stay flexible and efficient [16].

Ballou [17] provides a dynamic programming (DP) approach for determining the best site for a dynamic warehouse. This study can be seen as a catalyst for further research into dynamic layout issues. Exact algorithms, heuristic/metaheuristic algorithms and hybrid algorithms are three types of DFLP solution techniques. *Table 1* provides an overview of the methods and algorithms used, along with the authors of the respective papers.

Table 1 Consolidation of literature review of DFLP

Algorithm	Paper	Method
Exact Algorithms	(Ballou, 1968) [17]	DP
	(Rosenblatt, 1986) [9]	DP
	(Kim and Kim, 1999) [18]	Branch and Bound Algorithm
	(Pérez-gosende et al. 2024) [19]	Mixed integer non-linear programming
Heuristic/ Metaheuristic Algorithms	(Rosenblatt, 1986) [9]	Generating random layouts using computerized relative allocation of facilities technique (CRAFT) or computerized facilities design (COFAD) in each period
	(Urban, 1993) [20]	Steepest descent pairwise interchange
	(Conway and Venkataramanan, 1994) [21]	GA
	(Kaku and Mazzola, 1997) [22]	Tabu Search
	(Balakrishnan and Cheng, 2000) [12]	GA
	(Baykasoğlu and Gindy, 2001) [23]	Simulated annealing
	(McKendall et al. 2006) [24]	Simulated annealing
	(Rezazadeh et al., 2009) [25]	Discrete particle swarm
	(Nahas et al. 2010) [26]	Iterated great deluge
	(Yang et al. 2011) [27]	GA
	(Pillai et al. 2011) [15]	Simulated annealing
	(Molla et al. 2020) [28]	Chemical reaction optimization

Algorithm	Paper	Method
	(Zouein and Kattan, 2021) [29]	Improved construction approach using ant colony optimization
	(Palubeckis et al. 2022) [30]	Variable neighbourhood search (VNS) and fast LS
	(McKendall and Hakobyan 2021) [31]	GA
Hybrid Algorithms	(Balakrishnan et al. 2003) [32]	GA
	(McKendall and Shang, 2006) [33]	Ant Systems
	(Azimi and Saberi 2013) [34]	Discrete Particle Swarm and Simulation
	(Moslemipour 2018) [35]	Multi-Population GA
	(Hosseini et al.2014) [36]	Imperialist Competitive Algorithms, VNS, and Simulated Annealing
	(Chen, 2013) [37]	Ant Colony Algorithm
	(Pradeepmon et al. 2018) [10]	Estimation of distribution algorithm (EDA)
	(Khajemahalle et al. 2020) [38]	Hybrid nested partitions and simulated annealing algorithm
	(Hosseini et al. 2021) [39]	Modified GA and Cloud-based simulated annealing algorithm
	(Guan et al, 2022) [40]	Dynamic extended row facility layout problem (DERFLP)
	(Matai 2023) [41]	Simulated annealing algorithm
	(Sotamba et al. 2024) [42]	Mixed solution methodologies

The exact solution procedures guarantee the optimal solutions of the problem, but it may require a long time to provide the optimal solution.

Using a long time may not be practical in many situations, and thus researchers of the DFLP are normally satisfied with a near-optimal solution in a short span of time.

From the above discussion and *Table 1*, it is evident that non-conventional optimization methods are widely employed for solving DFLPs. Among the popular metaheuristic algorithms used for solving DFLPs, the GA has extensively studied over the years. The GA was in the limelight when researches tried to hybridise the metaheuristic algorithms for solving DFLPs. But none of the studies reported has not used the GA combined pair-wise exchange (PWX) LS or any other similar methods for solving DFLPs. Thus, in this work an hGA is proposed which hybridises GA with LS for solving the DFLPs.

3.Methodology

In the GA, a population of potential solutions is processed, with each member of the population represented as a valid solution in the form of a chromosome. The GA undergoes iterative processes aimed at steering the population toward enhanced solutions in terms of solution quality. These iterations of the GA involve several key steps, namely selection, reproduction, evaluation, and replacement, which are elaborated upon in the subsequent section. The algorithm concludes when it converges toward the optimal solution.

Figure 1 represents the flow chart for the proposed hGA and *Figure 2* represents the corresponding pseudo-code. In the proposed algorithm after the normal genetic operators present in GA (namely, selection, crossover and mutation) are over all the individuals in the population undergoes the LS procedure. After this LS is over the GA tests for the TC and if it satisfies, the algorithm is terminated reporting the obtained result.

Else, the genetic operators and LS are repeated. As a LS is incorporated into the GA, the neighborhood of the solutions is intensively searched for better solutions. Thus, the proposed hGA provides better quality solutions for the problems considered.

GA is a powerful tool for tackling complex optimization problems like DFLPs. However, they can sometimes get stuck in local optima, meaning they find a good solution but not necessarily the best one. Here's where hybridizing GA with a LS procedure can offer advantages. LS procedures excel at refining solutions within a specific area of the search space. By incorporating a LS after the GA's selection and crossover stages, the hGA can take promising solutions generated by the GA and further optimize them, potentially leading to better overall solutions for the DFLP.

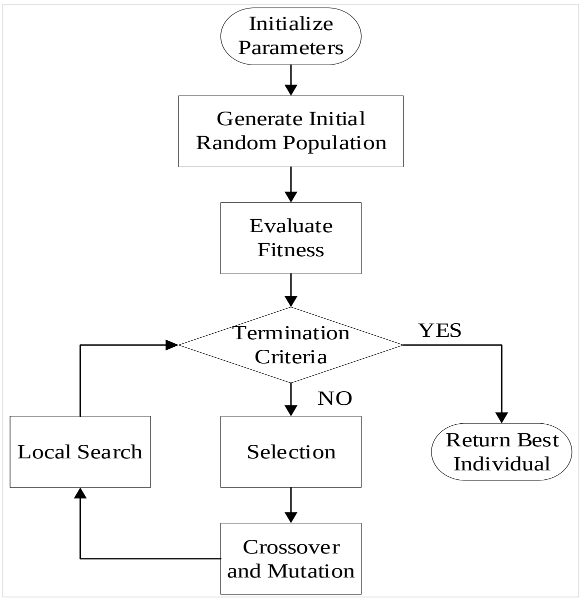

Figure 1 Flow diagram of hGA

```
hGA_pseudo_code

{

    1.    Initialise generation number $t = 0$ and choose a
          Termination Criteria (TC)
    2.    Generate an initial random population, $P(0)$ of
          individuals
    3.    Evaluate the fitness of individuals in $P(0)$
    4.    While TC not satisfied do
          {
              $t \leftarrow t + 1$;
              Select parents for offspring production;
              Apply crossover and mutation operations;
              Do the LS on all individuals in the population;
              Select a new population, $P(t)$, of survivors;
              Evaluate $P(t)$;
          }
    5.    Return the best individual of $P(t)$;
}
```

Figure 2 Pseudo code of GA

The GA's exploration capabilities help the hGA to search a broader range of solutions. This exploration can nudge the search out of areas with suboptimal solutions (local optima) that the GA might get trapped in on its own. The LS then takes over and refines these explored solutions, potentially leading to better overall optima. GA is good at exploring the entire solution space, but they might not always find the best solutions within a specific region. LS excels at exploiting promising areas but lacks the ability to explore widely. Hybridizing them combines these strengths, allowing the GA to explore broadly while the LS efficiently refines promising solutions identified during exploration. By focusing on refining promising solutions, the LS can potentially help the hGA converge to a good solution faster compared to a pure GA that might take longer to escape local optima. Overall, hybridizing GA with the LS for DFLPs has the potential to deliver better quality solutions, escape local optima, and converge faster compared to using a pure GA approach. In the proposed hGA, three critical parameters must be carefully determined: population size, crossover probability, and mutation probability.

Population size: Population size denotes the number of individuals or chromosomes present within the population at each generation. Typically, in hGAs, the population size remains constant throughout the evolutionary process. Keeping it fixed ensures that the algorithm explores the search space consistently. The choice of population size can vary depending on the specific problem, but it should be large enough to provide diversity and avoid premature convergence while being manageable in terms of computational resources.

Crossover probability determines the likelihood that selected parents undergo crossover to produce offspring. In hGAs, it is common to set the crossover probability relatively high, often around 0.9. A high crossover probability encourages the recombination of genetic material from parents, facilitating the exploration of potential solutions. This high value promotes genetic diversity within the population and helps in escaping local optima.

Mutation probability: Mutation probability signifies the likelihood of a mutation happening in a gene within a chromosome. Unlike crossover, mutation introduces small, random changes to individual chromosomes. The value of the mutation probability is typically kept low because biological mutations are relatively rare events. A low mutation probability ensures that the algorithm primarily relies on crossover for genetic diversity, using mutation as a mechanism to introduce occasional, small-scale changes that might lead to novel solutions.

When configuring an hGA, it's crucial to strike a balance between population size, crossover probability, and mutation probability. These parameters impact the exploration and exploitation of the search space. Population size influences the diversity of the population, while crossover and mutation probabilities determine the degree of genetic recombination and mutation, respectively. Finding the right combination of these parameters depends on the specific problem at hand and often involves experimentation and fine-tuning to achieve optimal performance. The hGA encompasses key operations, including selection, crossover, mutation, offspring insertion strategy and LS. These operations are explained below.

3.1Selection

The selection operator, inspired by Darwinian natural selection, is the core driving mechanism in GAs. It mimics nature's "survival of the fittest," influencing which individuals become parents in each generation. Striking the right balance is crucial: too much selection force can prematurely converge to suboptimal solutions, while too little can hinder progress. This operator steers the GA toward promising areas within the search space, ultimately shaping the evolution. Various selection schemes, like binary tournaments, RWS, Stochastic universal sampling, and rank selection, offer diverse strategies to achieve this, making it adaptable to different optimization tasks and preserving genetic diversity in the population. The selection procedure employed in the proposed hGA is RWS.

3.1.1Roulette wheel selection (RWS)

RWS, a widely used GA operator, mimics a roulette wheel by assigning probabilities based on an individual’s fitness level denoted as f(i), making it a key component in the selection process for GAs. This fitness value is computed as the reciprocal of the solution's associated cost. Within the current population, represented as P(t), each individual (i) is assigned a selection probability, p(i), which is in direct proportion to their fitness f(i). To streamline the selection process, every individual in the population is then provided with a cumulative probability range denoted as P(i) determined by their p(i) values. The selection process involves generating a uniform random number, denoted as r. Subsequently, the individual whose Ṗ(i) aligns with the generated random value, r, is replicated into the next population, P(t+1). The pseudocode of the implementation of RWS can be found in *Figure 3*. However, it's important to note that this method carries a potential drawback. It is susceptible to premature convergence toward local minima, especially when a dominant individual with high fitness dominates the population, potentially hindering the exploration of the entire solution space. Careful consideration and potential modifications are necessary to mitigate this risk and ensure the efficacy of the GA.

```
RWS_pseudo_code
{
    1.  Calculate the sum $S = \sum_{i=1}^{n} f(i)$
    2.  For each individual $1 \le i \le n$ do
        {
            Generate a random number $\alpha \in [0,S]$;
            $iSum = 0$ ; $j = 0$
            Do    {
                      $iSum$ □ $iSum + f(j)$
                      $j$ □ $j + 1$
                  } while ($iSum < \alpha$ and $j < n$)
            Select the individual $j$
        }
}
```

Figure 3 Pseudo code of RWS

3.2Crossover

The crossover operator mimics natural reproduction, enabling solutions to share information and create offspring solutions. It involves taking two or more parent solutions and using them to generate new solutions. While the selection process duplicates good solutions within the population, it doesn't introduce entirely novel ones. In contrast, the crossover operator is applied to produce improved offspring solutions. Numerous crossover operators are available, some tailored to specific problems, while others are more general in their applicability. Commonly used ones include single point crossover (SPC), order 1 crossover, cyclic crossover, position-based crossover, and partially mapped crossover. This research employs the SPC within the hGA. The SPC involves selecting a random crossover point in the parent solutions and exchanging the genetic material before and after that point to create offspring. This process allows for the combination of characteristics from both parents, potentially yielding solutions that inherit beneficial traits while promoting genetic diversity within the population. By implementing the SPC in the hGA, this work aims to harness its recombination power to discover improved solutions and enhance the algorithm's overall effectiveness.

3.2.1SPC

The SPC operator starts by selecting a single crossover point in the parent chromosomes. This operator splits the two parents at the selected position and the first part of the first parent is merged with the second part of the second parent and vice versa to obtain two new offspring. For example, consider the parent chromosomes P1: (1 2 3 4 5 6) and P2: (3 5 1 6 2 4), and suppose that the third position is selected as the crossover point. This leads to the following offsprings:

O1: (1 2 3 6 2 4) and O2: (3 5 1 4 5 6). Now, in O1 and O2 one element is repeating (2 in O1 and 5 in O2) and one element is missing (5 in O1 and 2 in O2). To make O1 and O2 feasible solutions, insert the missing element randomly in one of the positions of the repeating element. This gives the offsprings O1: (1 5 3 6 2 4) and O2: (3 5 1 4 2 6).

3.3Mutation

The mutation operator introduces random alterations to selected values within certain individuals (chromosomes) in the population. This mutation occurs with a low probability, mirroring its biological counterpart. By doing so, the mutation operator preserves genetic diversity in the population, strengthening the GAs capability to discover nearly optimal solutions. Common mutation techniques include displacement mutation, swap mutation (SM), insertion mutation, and inversion mutation. For this study, the SM method is applied, aiming to provide the GA with the means to explore new genetic variations and potentially uncover improved solutions in the optimization process.

3.3.1SM

The SM operator functions by randomly selecting two elements from the parent string and interchanging their positions. For instance, take the parent solution string (1 2 3 4 5 6). If the second and fifth elements are randomly selected, the result is the solution string (1 5 3 4 2 6). This process introduces variability in the genetic makeup of solutions, potentially leading to the discovery of better alternatives for optimization problems.

3.4Offspring insertion strategy

Options encompass the substitution of parents either with or without a specified probability, the replacement of parents when offspring exhibit higher fitness, enlarging the population size by merging offspring, and additional strategies. In this study, the strategy employed involves replacing parents in cases where the offspring demonstrate improved fitness. This approach ensures that the population continuously evolves towards better solutions by favouring the introduction of offspring that outperform their parent solutions.

3.4.1Parent replacement strategy

Once new solutions (offspring) are generated, there are several methods to incorporate them into the existing population. The offspring insertion method operates by comparing the fitness of offspring with that of their parents. When offspring exhibit superior fitness, they replace their parent solutions, while inferior offspring retain their parent's place in the population. This approach embodies a greedy selection strategy, retaining only the best solutions generated through crossover and mutation. Consequently, the population remains constant in size, always containing the most promising solutions in pursuit of optimization.

3.5LS

In this study, each individual within the population undergoes a LS process known as the PWX, LS method. PWX entails systematically exchanging two distinct elements within the current solution to seek improved solutions. If a better solution is identified through this exchange, it replaces the original, initiating a recursive PWX LS with the improved solution as the new focus. This process iterates until a solution is reached for which no superior solution exists within its PWX neighbourhood. Consequently, this study employs the recursive version of the PWX LS, ensuring that each individual is subjected to an iterative exploration of potential improvements, which can lead to the discovery of highly optimized solutions within the population.

3.6Dataset used

The test instances for DFLPs used in this research are those provided by Balakrishnan and Cheng [12]. The test data are grouped into six categories based on the number of departments and the number of periods as follows:

- 6 departments, 5 periods (6d × 5p)
- 6 departments, 10 periods (6d × 10p)
- 15 departments, 5 periods (15d × 5p)
- 15 departments, 10 periods (15d × 10p)
- 30 departments, 5 periods (30d × 5p)
- 30 departments, 10 periods (30d × 10p)

Each category consists of eight different instances and thus there are a total of 48 test instances.

3.7Illustration

To offer a comprehensive explanation of the proposed algorithm, the initial DFLP benchmark instance was employed, as introduced by Balakrishnan and Cheng [12], features a 6-department problem spanning 5 periods. *Table 2* provides crucial details such as the department distance matrix, matrix, MHCs between departments for all five periods, and the corresponding rearrangement costs.

Table 2 shows the Data for the 6-department × 5-period problem from Balakrishnan and Cheng [12]. Different period has been shown from a to g.

Table 2(a) Distance matrix

To / From	1	2	3	4	5	6
1	0	1	2	1	2	3
2	1	0	1	2	1	2
3	2	1	0	3	2	1
4	1	2	3	0	1	2
5	2	1	2	1	0	1
6	3	2	1	2	1	0

Table 2(b) Flow matrix: period 1

To / From	1	2	3	4	5	6
1	0	90	689	194	165	494
2	668	0	1324	811	241	206
3	631	387	0	125	281	375
4	80	495	615	0	222	221
5	276	204	1127	490	0	676
6	109	409	1780	394	200	0

Table 2(c) Flow matrix: period 2

To / From	1	2	3	4	5	6
1	0	257	1632	330	117	285
2	159	0	1309	297	803	404
3	98	82	0	271	222	383
4	110	404	1174	0	750	386
5	73	507	1679	190	0	107
6	152	487	355	646	315	0

Table 2(d) Flow matrix: period 3

To / From	1	2	3	4	5	6
1	0	1112	505	422	414	132
2	627	0	560	99	227	86
3	373	2007	0	235	384	205
4	482	1638	262	0	233	129
5	223	1196	520	55	0	75
6	200	782	271	292	235	0

Table 2(e) Flow matrix: period 4

To / From	1	2	3	4	5	6
1	0	1348	490	447	186	169
2	625	0	74	307	777	326
3	114	1645	0	288	975	68
4	156	578	447	0	554	212
5	353	732	118	373	0	283
6	328	1071	387	352	199	0

Table 2(f) Flow matrix: period 5

To / From	1	2	3	4	5	6
1	0	159	1103	218	297	95
2	631	0	1618	95	253	109
3	552	213	0	432	397	141
4	418	122	797	0	108	495
5	115	154	1610	425	0	158
6	167	214	2092	471	323	0

Table 2(g) Rearrangement costs

Department	**1**	**2**	**3**	**4**	**5**	**6**
Rearrangement Cost	898	911	627	538	738	977

3.7.1Working of the proposed algorithm

1. Generate population $\emptyset = \begin{Bmatrix} 5\ 3\ 6\ 1\ 2\ 4\ 5\ 3\ 4\ 1\ 2\ 6\ 5\ 3\ 4\ 1\ 2\ 6\ 5\ 3\ 4\ 1\ 2\ 6\ 5\ 3\ 4\ 1\ 2\ 6 \\ 2\ 3\ 1\ 4\ 5\ 6\ 2\ 3\ 1\ 4\ 5\ 6\ 2\ 3\ 6\ 1\ 5\ 4\ 2\ 3\ 6\ 1\ 5\ 4\ 2\ 3\ 6\ 1\ 5\ 4 \\ \vdots \\ 2\ 3\ 1\ 4\ 5\ 6\ 1\ 3\ 4\ 2\ 5\ 6\ 1\ 2\ 4\ 5\ 3\ 6\ 1\ 2\ 4\ 5\ 3\ 6\ 1\ 2\ 4\ 5\ 3\ 6 \end{Bmatrix}$ of size $d \times p / 2$ (= 15 for this problem) solutions by random permutations. Each solution consists of 30 numbers out of which, the first six numbers represent the order of arrangement of six facilities during the first period and the next six numbers represent the order of arrangement of six facilities during the second period and so on.
2. Determine the objective function value (the cost associated with the solution) by summing the transportation cost and rearrangement cost for each of the five periods.

$$\emptyset_{TC} = \begin{Bmatrix} 108053 \\ 109847 \\ \vdots \\ 108296 \end{Bmatrix}$$

And evaluate the fitness of each solution as 1/objective function value (cost associated with the solution).

3. Run the selection, crossover and mutation operations as described in the methodology section.
4. Execute the PWX LS on each of the solutions within the population.

$$\Phi = \begin{Bmatrix} 4\ 2\ 5\ 6\ 3\ 1\ 4\ 2\ 5\ 6\ 3\ 1\ 4\ 2\ 5\ 6\ 3\ 1\ 4\ 2\ 5\ 6\ 3\ 1\ 4\ 2\ 5\ 6\ 3\ 1 \\ 5\ 3\ 1\ 4\ 2\ 6\ 5\ 3\ 1\ 4\ 2\ 6\ 5\ 3\ 6\ 4\ 2\ 1\ 5\ 3\ 6\ 4\ 2\ 1\ 5\ 3\ 6\ 4\ 2\ 1 \\ \vdots \\ 2\ 3\ 1\ 4\ 5\ 6\ 1\ 3\ 4\ 2\ 5\ 6\ 1\ 2\ 4\ 5\ 3\ 6\ 1\ 2\ 4\ 5\ 3\ 6\ 1\ 2\ 4\ 5\ 3\ 6 \end{Bmatrix}$$

5. Determine the objective function value by adding the transportation cost and rearrangement cost for each of the five periods.

$$\Phi_{TC} = \begin{Bmatrix} 108024 \\ 108111 \\ \vdots \\ 108053 \end{Bmatrix}$$

6. Check for the termination criterion (TC), i.e., whether the current generation number is < the maximum number of generations (= 10 × d × p). If TC are met, end the algorithm and report the results, else calculate the fitness value for the objective function values and repeat steps 3 to 6.

3.8Experimental setup and parameters

The various parameters and operators employed in the proposed hGA are listed in *Table 3*. These parameters are obtained from the works of Pradeepmon et al. [43]. The proposed hGA is coded in Matlab, run on the Pentium 4, 2.60 GHz, 2 GB RAM processor and each of the 48 test instances are solved. The 48 test instances available fall into six different problem sizes with each problem size having eight different test instances. The six problem sizes of the test instances are 6d × 5p, 6d × 10p, 15d × 5p, 15d × 10p, 30d × 5p and 30d × 10p, where d in the problem size represents the number of departments and p represents the number of periods. The test instances are obtained from Balakrishnan and Cheng [12].

Table 3 Selected operations and parameters for hGA

S. No.	Parameter / Operator	Value
1	Population Size	$d \times p / 2$
2	Mutation Probability	0.04
3	Crossover Probability	0.9
4	Termination Criterion	Number of Generations = $10 \times d \times p$
5	Selection Procedure	RWS
6	Crossover Operator	SPC
7	Mutation Operator	SM
8	Offspring Insertion Strategy	Parent replacement strategy
9	LS Method	PWX LS
10	Fitness value	The inverse of the cost associated with the solution

d represents the number of departments and p represents the number of periods

For each test instance, ten solution runs are performed, and the resulting minimum and average costs are documented, along with a comparison to the best-known solutions found in the literature.

4.Results

Figure 4 represents a sample convergence graph of the algorithm for the 30d × 5p sized problems considered in this work. The graphs depict the average cost associated with the solutions over generations for the problem size 30d × 5p. Only the first instance is considered as a sample for plotting the convergence graph. The results of the experiments are presented in *Table 4*, which includes the minimum and average values achieved over ten algorithm trials for each test instance along with the reported average run time of the algorithm. *Table 5* displays a comprehensive comparison between the best solutions achieved and the best-known solutions from the literature. This comparison encompasses the results of the proposed algorithm and those published in various references, including Conway and Venkataraman (CVGA) [21], Mckendall et al. (modified simulated annealing (Mod SA))[24], McKendall and Shang (hybrid ant system (HAS))[33], Sahin and Turkbey (tabu list simulated annealing (TABUSA))[44], Nahas et al. (iterated great deluge (IGD))[26], Pillai et al. (simulated annealing (SA)) [15], Mckendall and Liu (heuristic tabu (HTABU))[45], Hosseini-nasab and Emami (hybrid particle swarm optimization (HPSO))[46], Turanoglu and Akkaya (simulated annealing bacterial foraging optimization (SABFO))[47], Zouein and Kattan (improved ant colony optimization (ASCOII))[29]. Notably all of these solution methodologies except for Pillai et al., employ adaptive approaches, while Pillai et al. utilizes a robust method for achieving near-optimal solutions. Detailed results of this comparison can be found in *Tables 5(a)* to *5(f).*

Figure 5 represents the percentage deviation of the solutions obtained by hGA from the best-known solutions available in the literature. Even though the solutions obtained for higher-sized problems are not as good as those provided in the literature, the simple hybridisation procedure provides near-optimal solutions. *Table 6* provides the comparison of solutions obtained using the hGA and SGA. From the table, it is evident that the results obtained using hGA are far better than those obtained using SGA. *Figure 6* illustrates the average percentage deviation of solutions obtained with hGA in comparison to those obtained with SGA. In *Figure 6*, the horizontal axis labels 1 to 8 represent the eight instances and 9 represents the average value over the eight instances.

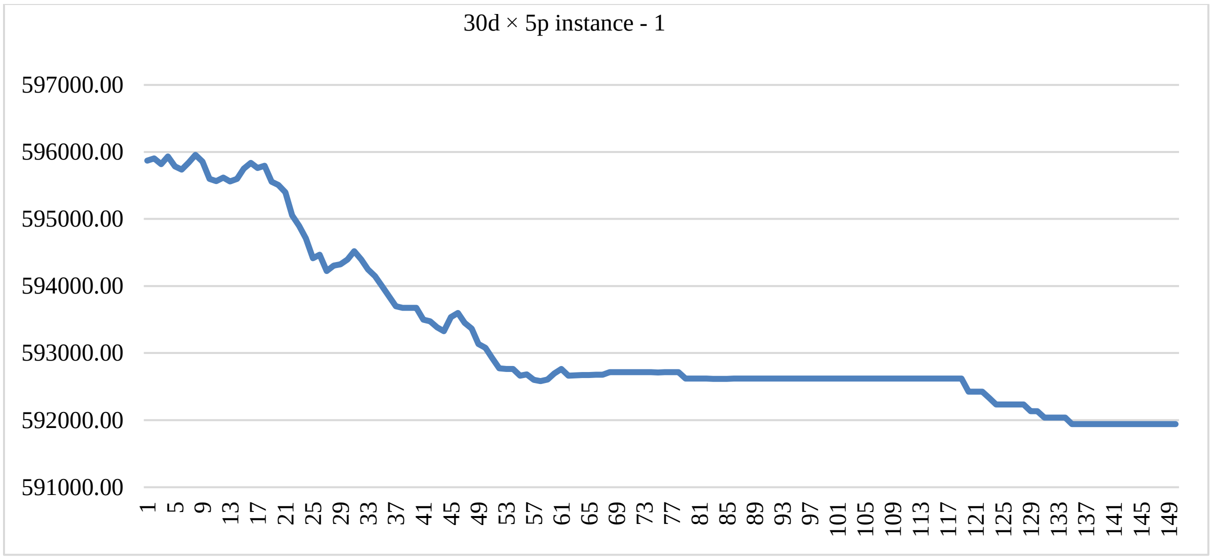

Figure 4 Average cost associated with the solutions over generations for the problem size 30d × 5p instance 1

Table 4 Minimum and average values and run time in seconds for the 48 test instances

Instance	**Description**	**Problem Size**					
		6d × 5p	**6d × 10p**	**15d × 5p**	**15d × 10p**	**30d × 5p**	**30d × 10p**
1	Minimum	106419	215143	483473	996130	588103	1197060
	Average	107624.6	217141.3	487718.1	997634.4	590614.9	1198805.1
	Run Time (s)	0.1	1.6	6.9	108.9	217.8	3616.9
2	Minimum	104834	212402	490757	990136	582605	1193757
	Average	105183.8	213861.4	491781.8	994215	585180.8	1198392.9
	Run Time (s)	0.1	1.5	7.0	110.9	219.7	3674.3
3	Minimum	104320	208605	496085	997860	586718	1185609
	Average	104745	209564.1	497083.3	1001937.3	587616.3	1189150
	Run Time (s)	0.1	1.5	7.0	104.2	223.3	3804.1
4	Minimum	106399	213858	489459	988390	577715	1171244
	Average	106631.4	214916.2	491461.1	990260.4	578618.3	1175855.7
	Run Time (s)	0.1	1.4	7.0	104.3	225.7	3745.8
5	Minimum	105628	211242	491444	991279	567854	1157242
	Average	106202.2	212827.2	493798.9	994503.1	571010.1	1160000.9
	Run Time (s)	0.1	1.5	7.0	104.9	221.5	3737.0
6	Minimum	103985	210707	492129	985221	577964	1173757
	Average	104748.9	211678.4	493306.2	987158.4	579492.1	1176024.5
	Run Time (s)	0.1	1.5	6.8	101.2	219.9	3665.9
7	Minimum	106439	215045	491214	992281	580421	1179738
	Average	107275.8	217440.3	492598.9	995177.3	582873.7	1184161.1
	Run Time (s)	0.1	1.5	6.8	101.2	225.4	3566.6
8	Minimum	103771	213900	496601	995096	586862	1192914
	Average	105112.6	216418.2	498004.6	997918.9	587649.7	1193989.9
	Run Time (s)	0.1	1.4	6.8	101.5	221.9	3628.8

Table 5(a) Comparison of results obtained using hGA with other results available in the literature for 6*d* × 5*p*

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
6 Department and 5 Period	CVGA [21]	1,08,976	1,05,170	1,04,520	1,06,719	1,05,628	1,05,606	1,06,439	1,04,485	**Average % Deviation**
	Mod SA [24]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	HAS [33]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	TABUSA [44]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,737	1,03,985	1,06,439	1,03,771	
	IGD [26]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	SA [15]	1,06,419	1,05,731	1,07,650	1,08,260	1,08,188	1,07,765	1,08,114	1,07,248	
	HTABU [45]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	HPSO [46]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	SABFO [47]	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	ACOII [29]	1,06,419	1,03,507	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	Best Cost	1,06,419	1,03,507	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	hGA	**1,06,419**	**1,04,834**	**1,04,320**	**1,06,399**	**1,05,628**	**1,03,985**	**1,06,439**	**1,03,771**	
	% Deviation	**0.00**	**1.28**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.16**

Table 5(b) Comparison of results obtained using hGA with other results available in the literature for 6d × 10p

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
6 Department and 10 Period	CVGA [21]	2,18,407	2,15,623	2,11,028	2,17,493	2,15,363	2,15,564	2,20,529	2,16,291	**Average % Deviation**
	Mod SA [24]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	HAS [33]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	TABUSA [44]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	IGD [26]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	SA [15]	2,20,776	2,17,412	2,19,024	2,17,350	2,17,142	2,17,397	2,19,788	2,20,144	
	HTABU [45]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	HPSO [46]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	SABFO [47]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	ACOII [29]	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	Best Cost	2,14,313	2,12,134	2,07,987	2,12,530	2,10,906	2,09,932	2,14,252	2,12,588	
	hGA	**2,15,143**	**2,12,402**	**2,08,605**	**2,13,858**	**2,11,242**	**2,10,707**	**2,15,045**	**2,13,900**	
	% Deviation	**0.39**	**0.13**	**0.30**	**0.62**	**0.16**	**0.37**	**0.37**	**0.62**	**0.37**

Table 5(c) Comparison of results obtained using hGA with other results available in the literature for 15d × 5p

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
15 Department and 5 Period	CVGA [21]	5,04,759	5,14,718	5,16,063	5,08,532	5,15,599	5,09,384	5,12,508	5,14,839	**Average % Deviation**
	Mod SA [24]	4,80,453	4,84,761	4,88,748	4,84,405	4,87,882	4,87,147	4,86,779	4,90,812	
	HAS [33]	4,80,453	4,84,761	4,88,748	4,84,446	4,87,722	4,86,685	4,86,853	4,91,016	
	TABUSA [44]	4,80,453	4,84,761	4,89,058	4,84,446	4,87,822	4,86,493	4,86,268	4,90,551	
	IGD [26]	4,80,453	4,84,761	4,89,058	4,84,446	4,87,822	4,86,493	4,86,268	4,90,812	
	SA [15]	5,06,847	5,00,284	5,08,011	5,03,699	5,02,622	4,99,891	5,02,919	5,07,970	
	HTABU [45]	4,80,453	4,84,761	4,88,748	4,84,446	4,87,911	4,86,493	4,86,592	4,90,812	
	HPSO [46]	4,80,453	4,78,310	4,86,987	4,83,813	4,84,968	4,86,493	4,85,384	4,89,150	
	SABFO [47]	4,80,453	4,84,853	4,89,981	4,86,006	4,88,556	4,88,196	4,87,476	4,91,789	
	ACOII [29]	4,80,453	4,84,761	4,88,748	4,84,446	4,87,753	4,86,493	4,86,732	4,90,551	
	Best Cost	4,80,453	4,78,310	4,86,987	4,83,813	4,84,968	4,86,493	4,85,384	4,89,150	
	hGA	**4,83,473**	**4,90,757**	**4,96,085**	**4,89,459**	**4,91,444**	**4,92,129**	**4,91,214**	**4,96,601**	
	% Deviation	**0.63**	**2.60**	**1.87**	**1.17**	**1.34**	**1.16**	**1.20**	**1.52**	**1.44**

Table 5(d) Comparison of results obtained using hGA with other results available in the literature for 15*d* × 10*p*

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
15 Department and 10 Period	CVGA [21]	10,55,536	10,61,940	10,73,603	10,60,034	10,64,692	10,66,370	10,66,617	10,68,216	**Average % Deviation**
	Mod SA [24]	9,79,468	9,78,065	9,82,396	9,72,797	9,77,188	9,67,617	9,79,114	9,83,672	
	HAS [33]	9,80,351	9,78,271	9,78,027	9,74,694	9,79,196	9,71,548	9,80,752	9,85,707	

Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
TABUSA [44]	9,78,848	9,77,338	9,81,172	9,71,720	9,76,781	9,68,362	9,78,660	9,82,880	
IGD [26]	9,78,848	9,78,304	9,81,172	9,71,759	9,77,234	9,68,067	9,78,930	9,82,888	
SA [15]	10,59,100	10,22,447	10,68,402	10,54,997	10,51,395	10,57,543	10,37,066	10,40,450	
HTABU [45]	9,81,412	9,78,004	9,83,109	9,71,720	9,77,100	9,71,287	9,78,576	9,83,341	
HPSO [46]	9,78,588	9,76,208	9,78,027	9,71,759	9,76,119	9,68,539	9,78,519	9,82,964	
SABFO [47]	9,82,087	9,79,095	9,82,914	9,74,144	9,79,376	9,70,247	9,83,527	9,84,664	
ACOII [29]	9,79,081	9,77,338	9,81,172	9,71,720	9,76,310	9,67,617	9,78,576	9,84,025	
Best Cost	9,78,588	9,76,208	9,78,027	9,71,720	9,76,119	9,67,617	9,78,519	9,82,880	
hGA	**9,96,130**	**9,90,136**	**9,97,860**	**9,88,390**	**9,91,279**	**9,85,221**	**9,92,281**	**9,95,096**	
% Deviation	**1.79**	**1.43**	**2.03**	**1.72**	**1.55**	**1.82**	**1.41**	**1.24**	**1.62**

Table 5(e) Comparison of results obtained using hGA with other results available in the literature for 30*d* × 5*p*

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	**Average % Deviation**
30 Department and 5 Period	CVGA [21]	6,32,737	6,47,585	6,42,295	6,34,626	6,39,693	6,37,620	6,40,482	6,35,776	
	Mod SA [24]	5,76,039	5,68,095	5,73,739	5,66,248	5,58,460	5,66,077	5,67,131	5,73,755	
	HAS [33]	5,76,886	5,70,349	5,76,053	5,66,777	5,58,353	5,66,762	5,67,131	5,75,280	
	TABUSA [44]	5,74,624	5,68,256	5,72,865	5,66,231	5,57,356	5,66,599	5,67,628	5,73,487	
	IGD [26]	5,75,386	5,69,045	5,72,104	5,64,398	5,55,555	5,64,124	5,67,775	5,72,802	
	SA [15]	5,79,704	5,76,350	5,86,831	5,84,318	5,70,492	5,72,782	5,71,703	5,96,835	
	HTABU [45]	5,74,657	5,67,481	5,71,462	5,64,868	5,55,628	5,65,100	5,66,993	5,73,023	
	HPSO [46]	5,77,248	5,69,175	5,72,105	5,66,124	5,55,551	5,64,804	5,67,131	5,73,755	
	SABFO [47]	5,78,415	5,70,630	5,77,390	5,68,289	5,58,345	5,72,536	5,69,993	5,77,873	
	ACOII [29]	5,73,722	5,66,776	5,65,411	5,64,171	5,54,281	5,64,110	5,64,682	5,72,207	
	Best Cost	5,73,722	5,66,776	5,65,411	5,64,171	5,54,281	5,64,110	5,64,682	5,72,207	
	hGA	**5,88,103**	**5,82,605**	**5,86,718**	**5,77,715**	**5,67,854**	**5,77,964**	**5,80,421**	**5,86,862**	
	% Deviation	**2.51**	**2.79**	**3.77**	**2.40**	**2.45**	**2.46**	**2.79**	**2.56**	**2.72**

Table 5(f) Comparison of results obtained using hGA with other results available in the literature for 30*d* × 10*p*

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	**Average % Deviation**
30 Department and 10 Period	CVGA [21]	13,62,513	13,79,640	13,65,024	13,67,130	13,56,860	13,72,513	13,82,799	13,83,610	
	Mod SA [24]	11,63,222	11,61,521	11,56,918	11,45,918	11,26,432	11,45,146	11,40,744	11,61,437	
	HAS [33]	11,66,164	11,68,878	11,66,366	11,48,202	11,28,855	11,41,344	11,40,773	11,66,157	
	TABUSA [44]	11,61,751	11,60,656	11,55,406	11,44,821	11,25,968	11,43,480	11,45,830	11,64,322	
	IGD [26]	11,57,887	11,58,243	11,55,319	11,40,395	11,23,385	11,40,723	11,45,098	11,62,700	
	SA [15]	11,72,691	11,82,286	11,88,620	11,98,487	11,98,674	12,02,033	12,10,573	12,09,088	
	HTABU [45]	11,59,589	11,57,942	11,54,799	11,43,110	11,23,446	11,41,144	11,45,951	11,60,484	
	HPSO [46]	11,60,388	11,58,243	11,56,198	11,49,753	11,23,673	11,47,935	11,42,031	11,60,658	
	SABFO [47]	11,68,453	11,70,042	11,60,204	11,49,944	11,32,136	11,44,677	11,60,830	11,72,857	
	ACOII [29]	11,57,703	11,56,900	11,52,546	11,41,149	11,19,496	11,40,883	11,44,727	11,06,651	

Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
Best Cost	11,57,703	11,56,900	11,52,546	11,40,395	11,19,496	11,40,723	11,40,744	11,06,651	
hGA	**11,97,060**	**11,93,757**	**11,85,609**	**11,71,244**	**11,57,242**	**11,73,757**	**11,79,738**	**11,92,914**	
% Deviation	**3.40**	**3.19**	**2.87**	**2.71**	**3.37**	**2.90**	**3.42**	**7.79**	**3.71**

Table 6 Comparison of results obtained using hGA with that using simple genetic algorithm (SGA)

	Algorithm	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	
6d × 5p	**SGA**	1,16,485	1,13,061	1,11,989	1,18,608	1,14,450	1,14,642	1,16,045	1,14,686	**Average**
	hGA	1,06,419	1,04,834	1,04,320	1,06,399	1,05,628	1,03,985	1,06,439	1,03,771	
	% Deviation	**-8.64**	**-7.28**	**-6.85**	**-10.29**	**-7.71**	**-9.30**	**-8.28**	**-9.52**	**-8.48**
6d × 10p	**SGA**	2,42,166	2,36,601	2,38,773	2,45,369	2,42,769	2,40,554	2,44,123	2,43,187	**Average**
	hGA	2,15,143	2,12,402	2,08,605	2,13,858	2,11,242	2,10,707	2,15,045	2,13,900	
	% Deviation	**-11.16**	**-10.23**	**-12.63**	**-12.84**	**-12.99**	**-12.41**	**-11.91**	**-12.04**	**-12.03**
15d × 5p	**SGA**	5,52,377	5,53,215	5,57,421	5,53,667	5,52,013	5,52,713	5,53,560	5,58,456	**Average**
	hGA	4,83,473	4,90,757	4,96,085	4,89,459	4,91,444	4,92,129	4,91,214	4,96,601	
	% Deviation	**-12.47**	**-11.29**	**-11.00**	**-11.60**	**-10.97**	**-10.96**	**-11.26**	**-11.08**	**-11.33**
15d × 10p	**SGA**	11,42,206	11,38,465	11,47,048	11,39,578	11,44,124	11,31,681	11,30,856	11,32,656	**Average**
	hGA	9,96,130	9,90,136	9,97,860	9,88,390	9,91,279	9,85,221	9,92,281	9,95,096	
	% Deviation	**-12.79**	**-13.03**	**-13.01**	**-13.27**	**-13.36**	**-12.94**	**-12.25**	**-12.14**	**-12.85**
30d × 5p	**SGA**	6,80,790	6,83,892	6,78,446	6,76,468	6,73,117	6,79,331	6,86,388	6,85,919	**Average**
	hGA	5,88,103	5,82,605	5,86,718	5,77,715	5,67,854	5,77,964	5,80,421	5,86,862	
	% Deviation	**-13.61**	**-14.81**	**-13.52**	**-14.60**	**-15.64**	**-14.92**	**-15.44**	**-14.44**	**-14.62**
30d × 10p	**SGA**	14,06,557	14,09,617	13,99,763	14,03,826	13,98,035	14,02,854	14,11,010	14,01,967	**Average**
	hGA	11,97,060	11,93,757	11,85,609	11,71,244	11,57,242	11,73,757	11,79,738	11,92,914	
	% Deviation	**-14.89**	**-15.31**	**-15.30**	**-16.57**	**-17.22**	**-16.33**	**-16.39**	**-14.91**	**-15.87**

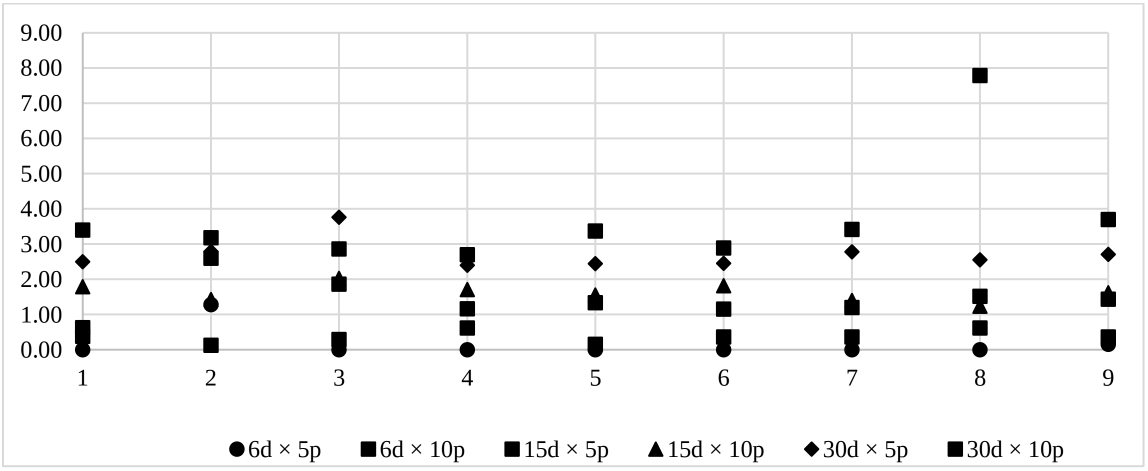

Figure 5 Percentage deviation between the solutions obtained through hGA and the best-known solutions is being assessed

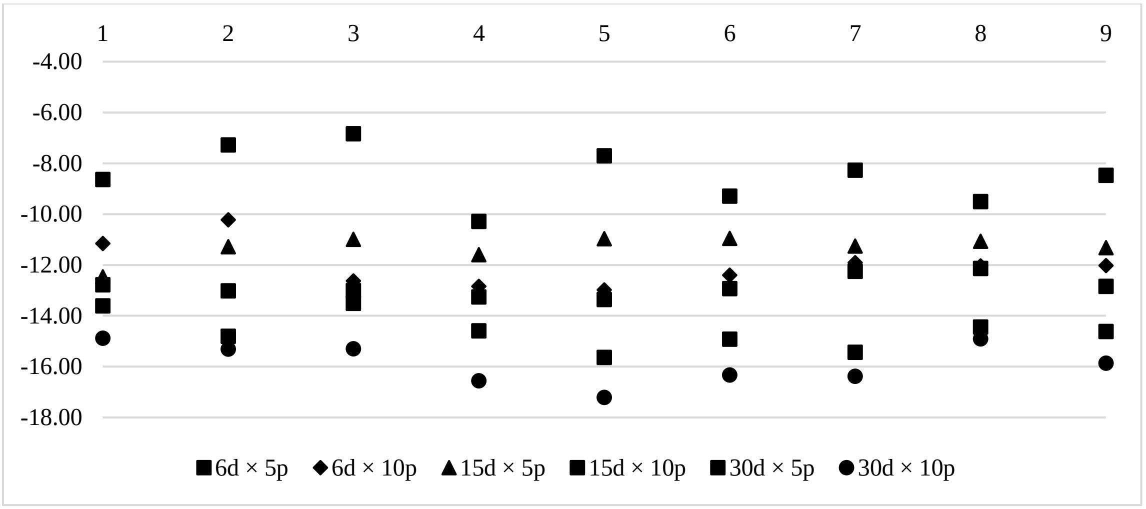

Figure 6 Percentage deviation of solutions obtained using hGA over that using SGA

5.Discussions

The experimental results encompass an analysis of convergence behaviour, solution quality, and comparative performance with existing methodologies. Firstly, *Figure 4* illustrates the convergence trajectory of the algorithm for a representative problem size (30d × 5p). The graph showcases the average cost associated with solutions across generations, providing insights into the algorithm's convergence dynamics which displays the algorithm's convergence behaviour and its efficiency in finding optimal or near-optimal solutions. Moving to *Table 4*, which summarizes the minimum and average solution values obtained over ten algorithm trials for each test instance along with the average runtime. This table offers a comprehensive overview of the algorithm's performance across different problem instances, shedding light on its consistency and computational efficiency. *Table 5* presents a comparative analysis between the solutions achieved by the proposed algorithm and the best-known solutions reported in the literature. This comparison, spanning various methodologies, offers valuable insights into the efficacy of the proposed approach. Notably, while the solutions obtained for higher-sized problems may not match those in the literature, the hGA demonstrates its capability in providing near-optimal solutions, particularly through its hybridization strategy. *Figure 5* further elucidates the percentage deviation of solutions obtained by the hGA from the best-known solutions in the literature. Despite discrepancies, especially for larger problem sizes, the hGA's simple hybridization technique proves effective in yielding solutions close to optimality. *Table 6* extends the discussion by comparing the solutions obtained using the hGA with those obtained using a SGA. The results showcase the superiority of the hGA over SGA, emphasizing the effectiveness of the hybrid approach in enhancing solution quality. Moreover, *Figure 6* offers a visual representation of the average percentage deviation of solutions obtained by the hGA compared to SGA across different instances. This graphical depiction underscores the consistent performance improvement achieved by the hGA over its standard counterpart.

This study utilized parameters from another published work by Pradeepmon et al. [43], which were initially derived for optimizing the GA used for QAPs. However, it would have been more appropriate if these parameters had been specifically tailored for optimizing the GA for DFLPs. Additionally, a limitation of the current research is that the proposed algorithm was only applied to DFLPs, and its effectiveness in solving other types of combinatorial optimization problems has not been evaluated. A complete list of abbreviations is listed in *Appendix I.*

6.Conclusion and future work

In this work, hGA is proposed for solving DFLPs. The hGA incorporates basic genetic operations along with a LS procedure. The algorithm is applied to 48 test instances, with the minimum and average values obtained over ten runs reported for each instance. Additionally, the average time taken to achieve these results is documented. The results demonstrate that the hGA consistently delivers solutions that are within four percent of the best-known solutions for all test instances, with the average percentage deviation ranging from 0% for smaller instances to 3.71% for larger ones. Although the average runtime for these instances is mentioned, it is not compared with other published times due to significant variations in machine configurations used in different studies, rendering such comparisons irrelevant. While the results for larger test instances show some deviation from the best-known solutions, the hGA proves effective at solving DFLPs to near-optimal levels efficiently.

Future works may focus on:

- Further hybridizing the SGA with other algorithms or incorporating advanced search techniques like VNS into the GA framework.
- Integrating machine learning techniques such as clustering algorithms to select candidates for LS, which could improve runtime.
- Applying the proposed hGA to solve unequal area DFLPs, opening up new research possibilities.

Acknowledgment

The authors would like to thank Professors Jaydeep Balakrishnan and Chun Hung Cheng for providing the datasets.

Conflicts of interest

The authors have no conflicts of interest to declare.

Data availability

The DFLP dataset used in this study is considered from Balakrishnan and Cheng [12] and is publicly accessible. It can be found at: https://prism.ucalgary.ca/items/3cc97aa9-fa0b-4cf6-b6de-8bb3bb998498/full

Authors contribution statement

Vineetha G. R: Data Collection, Investigation, Conceptualization, Data curation, Writing – original draft, review and editing, Analysis and interpretation of results and manuscript preparation. **Shiyas C. R:** Supervision, Guidance, Investigation, Conceptualization, Analysis and correction of paper

References

[1] Aziz FA. Manufacturing system. BoD–Books on Demand, 2012.

[2] Tompkins JA, White JA, Bozer YA, Tanchoco JM. Facilities planning. John Wiley & Sons; 2010.

[3] Abdollahi P, Aslam M, Yazdi AA. Choosing the best facility layout using the combinatorial method of gray relation analysis and nonlinear programming. Journal of Statistics and Management Systems. 2019; 22(6):1143-61.

[4] Montoya-torres JR, Aponte A, Rosas P, Caballero-villalobos JP. Applying GRASP meta-heuristic to solve the single-item two-echelon uncapacitated facility location problem. International Journal of Applied Decision Sciences. 2010; 3(4):297-310.

[5] Balakrishnan J, Cheng CH. Dynamic layout algorithms: a state-of-the-art survey. Omega. 1998; 26(4):507-21.

[6] Koopmans TC, Beckmann M. Assignment problems and the location of economic activities. Econometrica: Journal of the Econometric Society. 1957:53-76.

[7] Sahni S, Gonzalez T. P-complete approximation problems. Journal of the ACM. 1976; 23(3):555-65.

[8] Loiola EM, De ANM, Boaventura-netto PO, Hahn P, Querido T. A survey for the quadratic assignment problem. European journal of Operational Research. 2007; 176(2):657-90.

[9] Rosenblatt MJ. The dynamics of plant layout. Management Science. 1986; 32(1):76-86.

[10] Pradeepmon TG, Panicker VV, Sridharan R. A heuristic algorithm enhanced with probability-based incremental learning and local search for dynamic facility layout problems. International Journal of Applied Decision Sciences. 2018; 11(4):352-89.

[11] Balakrishnan J. Solutions for the constrained dynamic plant layout problem. Indiana University; 1991.

[12] Balakrishnan J, Cheng CH. Genetic search and the dynamic layout problem. Computers & Operations Research. 2000; 27(6):587-93.

[13] Braglia M, Zanoni S, Zavanella L. Robust versus stable layout design in stochastic environments. Production Planning & Control. 2005; 16(1):71-80.

[14] Balakrishnan J, Cheng CH. The dynamic plant layout problem: incorporating rolling horizons and forecast uncertainty. Omega. 2009; 37(1):165-77.

[15] Pillai VM, Hunagund IB, Krishnan KK. Design of robust layout for dynamic plant layout problems. Computers & Industrial Engineering. 2011; 61(3):813-23.

[16] Chen GY. Multi-objective evaluation of dynamic facility layout using ant colony optimization. The University of Texas at Arlington; 2007.

[17] Ballou RH. Dynamic warehouse location analysis. Journal of Marketing Research. 1968; 5(3):271-6.

[18] Kim JG, Kim YD. A branch and bound algorithm for locating input and output points of departments on the block layout. Journal of the Operational Research Society. 1999; 50(5):517-25.

[19] Pérez-gosende P, Mula J, Díaz-madroñero M. A bottom-up multi-objective optimisation approach to dynamic facility layout planning. International Journal of Production Research. 2024 ;62(3):626-43.

[20] Urban TL. A heuristic for the dynamic facility layout problem. IIE Transactions. 1993; 25(4):57-63.

[21] Conway DG, Venkataramanan MA. Genetic search and the dynamic facility layout problem. Computers & Operations Research. 1994; 21(8):955-60.

[22] Kaku BK, Mazzola JB. A tabu-search heuristic for the dynamic plant layout problem. INFORMS Journal on Computing. 1997; 9(4):374-84.

[23] Baykasoğlu A, Gindy NN. A simulated annealing algorithm for dynamic layout problem. Computers & Operations Research. 2001; 28(14):1403-26.

[24] Mckendall JAR, Shang J, Kuppusamy S. Simulated annealing heuristics for the dynamic facility layout problem. Computers & Operations Research. 2006; 33(8):2431-44.

[25] Rezazadeh H, Ghazanfari M, Saidi-mehrabad M, Jafar SS. An extended discrete particle swarm optimization algorithm for the dynamic facility layout problem. Journal of Zhejiang University-Science A. 2009; 10:520-9.

[26] Nahas N, Kadi DA, El FMN. Iterated great deluge for the dynamic facility layout problem. CIRRELT; 2010.

[27] Yang CL, Chuang SP, Hsu TS. A genetic algorithm for dynamic facility planning in job shop manufacturing. The International Journal of Advanced Manufacturing Technology. 2011; 52:303-9.

[28] Molla MR, Naznin M, Islam MR. Dynamic facility layout problem using chemical reaction optimization. In 4th international conference on computer, communication and signal processing 2020 (pp. 1-5). IEEE.

[29] Zouein PP, Kattan S. An improved construction approach using ant colony optimization for solving the dynamic facility layout problem. Journal of the Operational Research Society. 2022; 73(7):1517-31.

[30] Palubeckis G, Ostreika A, Platužienė J. A variable neighborhood search approach for the dynamic single row facility layout problem. Mathematics. 2022; 10(13):1-27.

[31] Mckendall A, Hakobyan A. An application of an unequal-area facilities layout problem with fixed-shape facilities. Algorithms. 2021; 14(11):1-14.

[32] Balakrishnan J, Cheng CH, Conway DG, Lau CM. A hybrid genetic algorithm for the dynamic plant layout problem. International Journal of Production Economics. 2003; 86(2):107-20.

[33] Mckendall JAR, Shang J. Hybrid ant systems for the dynamic facility layout problem. Computers & Operations Research. 2006; 33(3):790-803.

[34] Azimi P, Saberi EJ. An efficient hybrid algorithm for dynamic facility layout problem using simulation technique and PSO. Economic Computation & Economic Cybernetics Studies & Research. 2013; 47(4).

[35] Moslemipour G. A hybrid CS-SA intelligent approach to solve uncertain dynamic facility layout problems considering dependency of demands. Journal of Industrial Engineering International. 2018; 14(2):429-42.

[36] Hosseini S, Khaled AA, Vadlamani S. Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. Neural Computing and Applications. 2014; 25:1871-85.

[37] Chen GY. A new data structure of solution representation in hybrid ant colony optimization for large dynamic facility layout problems. International Journal of Production Economics. 2013; 142(2):362-71.

[38] Khajemahalle L, Emami S, Keshteli RN. A hybrid nested partitions and simulated annealing algorithm for dynamic facility layout problem: a robust optimization approach. INFOR: Information Systems and Operational Research. 2021; 59(1):74-101.

[39] Hosseini SS, Azimi P, Sharifi M, Zandieh M. A new soft computing algorithm based on cloud theory for dynamic facility layout problem. RAIRO-Operations Research. 2021; 55:S2433-53.

[40] Guan C, Zhang Z, Zhu L, Liu S. Mathematical formulation and a hybrid evolution algorithm for solving an extended row facility layout problem of a dynamic manufacturing system. Robotics and Computer-Integrated Manufacturing. 2022; 78:102379.

[41] Matai R. A unique discrete formulation for unequal area dynamic facility layout problem. In international conference on industrial engineering and engineering management 2023 (pp. 622-6). IEEE.

[42] Sotamba LM, Peña M, Siguenza-Guzman L. Driver analysis to solve dynamic facility layout problems: a literature review. In international conference on flexible automation and intelligent manufacturing 2024 (pp. 242-9). Springer, Cham.

[43] Pradeepmon TG, Panicker VV, Sridharan R. Genetic algorithm for quadratic assignment problems: application of Taguchi method for optimisation. International Journal of Operational Research. 2020; 38(2):193-220.

[44] Şahin R, Türkbey O. A new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem. International Journal of Production Research. 2009; 47(24):6855-73.

[45] Mckendall JAR, Liu WH. New tabu search heuristics for the dynamic facility layout problem. International Journal of Production Research. 2012; 50(3):867-78.

[46] Hosseini-nasab H, Emami L. A hybrid particle swarm optimisation for dynamic facility layout problem. International Journal of Production Research. 2013; 51(14):4325-35.

[47] Turanoğlu B, Akkaya G. A new hybrid heuristic algorithm based on bacterial foraging optimization for the dynamic facility layout problem. Expert Systems with Applications. 2018; 98:93-104.

Vineetha G. R born on May 5, 1986, in Kollam, India, is an accomplished researcher currently pursuing a PhD in the realm of Optimization with a focus on Dynamic Facility Layout Problems, employing Genetic Algorithms as a principal methodology. Armed with a Master's degree in Technology (M.Tech), Vineetha has demonstrated exceptional dedication to pushing the boundaries of knowledge in this complex field. With an impressive track record of over ten publications in esteemed journals and conference proceedings, she has established herself as a respected authority in the domain of optimization and is committed to unravelling innovative solutions to practical problems

through her pioneering work with genetic algorithms. Vineetha's academic journey is marked by her unwavering commitment to the advancement of optimization research, solidifying her position as a distinguished scholar with a promising future in pushing the frontiers of knowledge in dynamic facility management.
Email: grvineethavysakh@gmail.com

Dr. Shiyas C. R, is an Associate Professor at Cochin University of Science and Technology (CUSAT), brings extensive expertise to the field of Industrial Engineering. Born on April 30, 1976, Shiyas holds a Ph.D. in Industrial Engineering and has made invaluable contributions to academia during a two-decade-long tenure at CUSAT. With a strong academic foundation and an impressive publication record in SCI-indexed and Scopus-indexed journals. Dr. Shiyas is a prominent figure in the research community, exemplifying dedication and excellence in advancing knowledge within the discipline of Mechanical Engineering. His wealth of experience and scholarly accomplishments indicate his significant role in shaping the academic landscape.
Email: crshiyas@gmail.com

Appendix I

S. No.	Abbreviation	Description
1	ASCOII	Improved Ant Colony Optimization
2	COFAD	Computerized Facilities Design
3	CRAFT	Computerized Relative Allocation of Facilities Technique
4	CVGA	Conway and Venkatraman Genetic Algorithm
5	DERFLP	Dynamic Extended Row Facility Layout Problem
6	DFLP	Dynamic Facility Layout Problem
7	DP	Dynamic Programming
8	EDA	Estimation of Distribution Algorithm
9	FLP	Facility layout problem
10	GA	Genetic algorithm
11	HAS	Hybrid Ant System
12	hGA	Hybrid Genetic Algorithm
13	HPSO	Hybrid Particle Swarm Optimization
14	HTABU	Heuristic Tabu
15	IGD	Iterated Great Deluge
16	LS	Local Search
17	MHC	Material Handling Cost
18	Mod SA	Modified Simulated Annealing
19	PWX	Pair-Wise Exchange
20	QAP	Quadratic Assignment Problem
21	QIP	Quadratic Integer Programming
22	RWS	Roulette Wheel Selection
23	SA	Simulated Annealing
24	SABFO	Simulated Annealing Bacterial Foraging Optimization
25	SFLP	Static Facility Layout Problem
26	SGA	Simple Genetic Algorithm
27	SM	Swap Mutation
28	SPC	Single Point Crossover
29	TABUSA	Tabu List Simulated Annealing
30	TC	Termination criteria
31	VNS	Variable neighbourhood search