ENHANCING PLANT LAYOUT OPTIMIZATION: A MULTI-CRITERIA DECISION-MAKING APPROACH USING METAHEURISTIC ALGORITHMS CORELAP, ALDEP AND PYTHON PROGRAMMING

Sachin S. Pund[1], Dr. D. R. Zanwar[2]

[1] Assistant Professor, Department of Industrial Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra, India

[2] Associate Professor, Department of Industrial Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra, India

Email: [1] pundss@rknec.edu

Abstract

This paper presents a comprehensive study on the optimization of plant layout using two prominent Meta-heuristic algorithms: CORELAP and ALDEP. The plant layout optimization problem, recognized as an NP-hard challenge, concentrates on reducing the overall material handling cost by strategically arranging facilities and departments within a factory. CORELAP and ALDEP have emerged as pivotal tools for addressing this complex issue. In this study, we conduct a comparative analysis of these two algorithms, applying them to a real-world case study involving a manufacturing plant. The findings reveal that both algorithms excel in optimizing the layout; however, CORELAP demonstrates superior performance in terms of computational efficiency and solution quality. Moreover, this study introduces an innovative approach that integrates Python programming with ALDEP to generate a multitude of alternative solutions. This hybrid methodology significantly enhances the ability to visualize and assess numerous layout possibilities. Through exhaustive experimentation, it becomes evident that this hybrid approach, when combined with ALDEP, yields exceptional results, surpassing the capabilities of CORELAP. In conclusion, the research underscores the vital role of optimization in modern, competitive markets. It emphasizes the significance of minimizing waste, enhancing productivity, and achieving operational excellence through the optimization of manufacturing processes, material handling, and plant layout design. The study provides valuable insights for both industry practitioners and researchers in the realm of plant layout optimization, highlighting the potential of hybrid solutions like the one presented here to drive efficiency and competitiveness in manufacturing operations.

Keywords: *Plant Layout, ALDEP, CORELAP, optimization, python programming*

1. Introduction

This paper presents a study on optimizing plant layout using two Meta-heuristic algorithms: ALDEP and CORELAP. ALDEP (Automated Layout Design Program) considers various layout parameters, including the size and shape of the facility, the number and type of resources to be arranged, and the required material flow paths. ALDEP is a widely used algorithm that has been successfully applied in several industries, including manufacturing, warehousing, and healthcare. CORELAP (Computerized Relationship Layout Planning) considers the relationships between different resources within a facility and the flow paths of materials. The algorithm generates a layout design by iteratively improving upon an initial design, optimizing layout parameters such as material handling distance, resource proximity, and accessibility. Both ALDEP and CORELAP algorithms is successfully applied in various industries and found effective in optimizing facility layouts. However, their effectiveness is largely dependent on the accuracy of input data and the constraints considered during the optimization process. Thus, it is essential to select algorithms based on the specific requirements and constraints of the facility layout problem at hand. This paper aims to provide a comprehensive review of the use of ALDEP and CORELAP algorithms in facility layout optimization, highlighting their strengths, weaknesses, and applications in different industries. The plant layout optimization problem is a well-known NP-hard problem in which the objective is to minimize the total material handling cost by determining the optimal arrangement of facilities and departments within a factory. CORELAP and ALDEP are two widely used algorithms for solving this problem. In this study, the performance of these two algorithms is compared by implementing them on a case study of a manufacturing plant. The results show that both algorithms are effective in optimizing the layout, but CORELAP outperforms ALDEP in terms of computational time and solution quality. ALDEP is having capacity to generate the n! Alternate solutions. Where n is the number of departments. Generating these many solutions manually is next to impossible. It finds a wide array of solutions but not manually. The innovative solution to this issue is tried to resolve by incorporating python programming. It generates n! Possible outcomes of the alternative layouts with program. This novelty is implemented in this experimentation. The study provides useful insights for practitioners and researchers in the field of plant layout optimization. In today's highly competitive global market, optimization is a critical factor for the success and survival of any business organization. Optimization plays a crucial role in minimizing waste, improving productivity, and achieving operational efficiency across different functions within the organization. One of the key areas where optimization can be implemented is in the manufacturing process, material handling, and plant layout design. To achieve this, suitable techniques and algorithms are required for decision-making. Process optimization involves adjusting a process to optimize a set of specified parameters while adhering to the necessary constraints. The primary goals of process optimization are to minimize costs, maximize throughput, and improve efficiency. It is a critical quantitative tool used in industrial decision-making.

2. Problem Definition

Plant layout optimization is a critical aspect of industrial facility design, aiming to minimize material handling costs, enhance operational efficiency, and adhere to various constraints. In this study, we present a comprehensive examination of plant layout optimization using two well-established Meta-heuristic algorithms, ALDEP and CORELAP. Additionally, introduced a novel hybrid approach that integrates ALDEP's logic with Python programming to generate a multitude of alternative solutions. The problem definition is framed around the objective of comparing the solutions obtained through ALDEP and CORELAP while exploring a wide range of alternative layouts through the hybrid approach. Our focus is on achieving optimal layouts that minimize material handling costs.

The key components of our methodology, which involves:

1. **ALDEP Algorithm:** We employ ALDEP to generate initial layouts and explore its stochastic nature, which often results in multiple solutions.
2. **Python Integration:** A custom Python program is developed to integrate with ALDEP, enabling the generation of a vast number of alternative layouts.
3. **CORELAP Comparison:** To evaluate the effectiveness of our layouts, we run CORELAP on the same dataset and compare the material handling costs, distances traveled, and solution quality with those obtained through ALDEP and the hybrid approach.

Results showcase the comparative performance of ALDEP and CORELAP while highlighting the hybrid approach's ability to produce a wide array of layout alternatives. The evaluation metrics include material handling cost reduction, distances traveled per trip, and solution quality. The outcomes of this study provide valuable insights for practitioners and researchers in the field of plant layout optimization. By leveraging the strengths of ALDEP, CORELAP, and Python programming, here in this paper, offer a holistic approach to address complex layout challenges, ultimately contributing to improved operational efficiency and cost savings in manufacturing facilities.

3. List of the Departments and Relationship Chart:

A	I	E	O	O	I	U	U	U	U	U	O	U	U	U	U	U	U	15	1
	O	E	E	O	I	U	U	U	U	U	E	U	U	U	U	U	U	13	2
		A	U	U	O	O	O	U	U	U	U	U	U	U	U	U	U	7	6
			A	U	U	O	U	U	U	U	U	A	U	U	U	U	U	9	3
				O	U	O	O	U	U	U	U	I	U	U	U	A	U	9	4
					U	U	U	U	U	U	U	O	U	U	U	U	U	1	14
						A	U	U	U	U	U	U	U	U	U	U	U	4	8
							O	U	U	U	U	U	U	U	U	U	E	4	11
								U	U	U	U	U	U	U	U	U	U	0	15
									A	I	I	U	U	U	U	U	U	8	5
										A	I	U	U	U	U	U	U	6	7
											A	U	U	U	U	U	U	4	9
												U	E	U	U	U	U	3	12
													U	U	U	U	U	0	16
														I	U	U	U	2	13
															A	U	U	4	10
																U	U	0	17
																	U	0	18
																		0	19

Department	Abbreviation	Area	Round off Area	Grids	Label
Polishing	PO	24.7572	20	2	3
Drilling Machine 1	DM1	12.271	10	1	4
Drilling Machine 2	DM2	10.3334	10	1	5
Cutting Machine	CUM	15.3925	10	1	18
Countering Machine	CON	11.8404	10	1	14
Reaming	R	11.8404	10	1	9
Grinding	G	31.9691	30	3	8
Threading	T	36.167	40	4	10
1st Bending	B1	38.7504	40	4	11
2nd Bending	B2	22.3891	20	2	12
Tack Welding	TW	53.82	50	5	1
Full Welding	FW	38.7504	40	4	2
Chipping	C	24.219	20	2	7
3-in-1 Chemical	3-1C	42.6254	40	4	15
Priming	PR	106.564	110	11	6
Oven Baking	O	80.73	80	8	16
Assembly	A	38.7504	40	4	17
Qualifying	Q	68.8896	70	7	13
Storage	S	159.845	160	16	19

4. Processing Sequence:

Sr. No.	Part	Operation Sequence of Machines
1	Lower Link Plate	J→F→I→A→D→H→B
2	Connecting Link	J→I→E→A→E→H→B
3	V Link	J→G→E→A→H→B
4	Stay Rod	J→C→G→E→A→C→K→B
5	Draw Bar	J→I→A→D→H→B

5. Legends of Relationship:

Legends	Relationship	Score	Score
Absolute Necessary	A	4^3	64
Especially Important	E	4^2	16
Important	I	4^1	4
Ordinary Important	O	4^0	1
Unimportant	U	0	0
Undesirable	X	-1024	-1024

6. Experimentation Using CORELAP:

Table 6.1: Relationship Matrix of LLP:

Processing Sequence		P	DM-1	DM-2	CM	CO2WM-1	CO2WM-2	G	C	TCR	PLACEMENT RANK
1	P		E	O	U	A	U	U	U	8	1
2	DM-1			I	O	I	U	U	U	4	5
3	DM-2				I	U	U	U	U	2	7
4	CM					A	O	U	U	5	3
5	CO2 WM-1						I	I	U	4	4
6	CO2 WM-2							A	I	6	2
7	G								E	3	6
8	C									0	8

Table 6.2: Relationship Matrix of CL:

Processing Sequence		P	DM -1	DM -2	CM	CO2WM-1	CO2WM-2	G	C	TCR	PLACEMENT RANK
1	P		E	O	U	A	U	U	U	8	1
2	DM-1		0	E	O	I	U	U	U	5	3
3	DM-2				E	U	U	U	U	3	6
4	CM					A	O	U	U	4	4
5	CO2WM-1						I	I	U	4	5
6	CO2WM-2							A	I	6	2
7	G								E	3	7
8	C									0	8

Table 6.3: Relationship Matrix of VL:

Processing Sequence		T	B1	B2	CO2WM-1	CO2WM-2	Q	C	3-1 C	P	OB	A	TCR
1	T		I	U	U	U	U	U	U	U	U	U	2
2	B1			E	U	U	O	U	U	U	U	U	4
3	B2				I	U	U	U	U	U	U	U	2
4	CO2WM-1					E	O	U	U	U	U	U	4
5	CO2WM-2						E	E	U	U	U	U	6
6	Q							A	U	U	U	U	4
7	C								E	U	U	U	3
8	3-1 C									I	U	U	2
9	P										A	U	4
10	OB											A	4
11	A												0

Table 6.4: Relationship Matrix of DBSP:

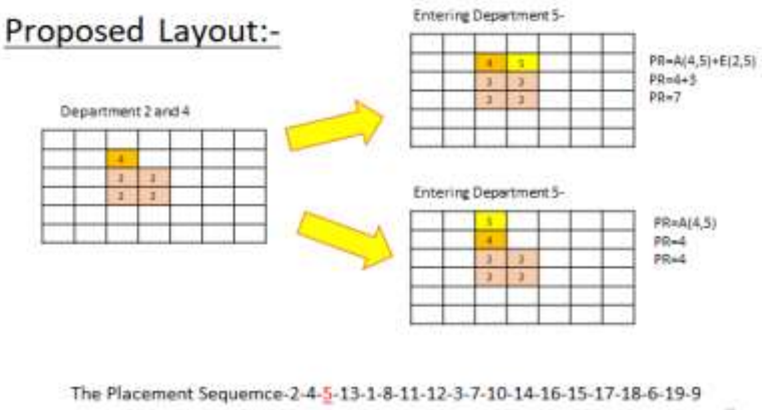
Processing Sequence		CO2WM-1	CO2WM-2	P	C	S	TCR	PLACEMENT RANK
1	CO2WM-1		A	U	U	U	4	1
2	CO2WM-2		0	E	O	U	4	3
3	P			0	E	U	3	4
4	C					A	4	2
5	S						0	5

Table 6.5: Relationship Matrix of SR:

Processing Sequence		CO2WM-1	CO2WM-1	P	C	R	TCR	PLACEMENT RANK
1	CO2WM-1		A	U	U	U	4	1
2	CO2WM-1			E	O	U	3	4
3	P				E	U	3	3
4	C					A	4	2
5	R						0	5

7. The placement Sequence generated Using CORELAP:

2-4-5-13-1-8-11-12-3-7-10-14-16-15-17-18-6-19-9



8. Decision on Placement Rank:

CL-Machine	Placement Rank
CO2 welding machine 1	1
CO2 welding machine 2	4
Priming	3
Chipping	2
Reaming	5

V-Link-Machine	Placement Rank
CO2 welding machine 1	1
CO2 welding machine 2	3
Priming	4
Chipping	2
Storage	5

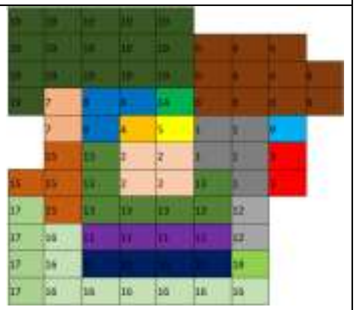
Stay Rod Machine	Placement Rank
Threading	9
1st Bending	6
2nd bending	10
CO2 Welding Machine 1	5
CO2 welding machine 2	1
Qualifying	4
Chipping	7
3-in-1 Chemical	11
Priming	2
Oven Baking	3
Assembly	12

LLP-Machine	Placement Rank
Polishing	1
Drilling Machine 1	5
Drilling Machine 2	7
Cutting Machine	3
CO2 welding machine 1	4
CO2 welding machine 2	2
Grinding	6
Chipping	8

DBSP-Machine	Placement Rank
Polishing	1
Drilling Machine 1	3
Drilling Machine 2	6
Countering Machine	4
CO2 welding machine 1	5
CO2 welding machine 2	2
Grinding	7

9. Output Generated Using Python Programming:

Result is generated using PYTHON Programming with the given input of the area of the department, Number of grids developed according to the area ,relationship between the department, relationship score, adjacency etc. and around 100000 (One Lakh) alternative layout were obtained using the same. Out of the generated solution, layout number 37427 has generated a score of 670 as shown in Fig. The numbers of 20340 pages of the solution are generated from the programming as solution.

Existing Plant Layout constructed using ALDEP	The proposed layout generated using ALDEP and Python	Layout Generated Using CORELAP
		
The adjacency and relationship score Generated for five layouts 83,169,127,148,94,136	Out of one Lakh alternative layout and its adjacency layout number 37427 has generated a score of **670**.	The adjacency and relationship Score generated **440**.

Adjacent station		Relationship	Relationship score
17	[illegible]	U	0
16	[illegible]	U	0
16	[illegible]	U	0
[illegible]	[illegible]	U	0
[illegible]	15	U	0
[illegible]	7	U	0
15	[illegible]	U	0
[illegible]	2	U	0
[illegible]	[illegible]	E	16
[illegible]	18	U	0
[illegible]	4	U	0
[illegible]	[illegible]	U	0
[illegible]	14	U	0
[illegible]	[illegible]	U	0
7	13	U	0
[illegible]	13	U	0
14	13	U	0
[illegible]	13	U	0
[illegible]	5	U	0
4	18	U	0
18	14	U	0
[illegible]	5	O	1
[illegible]	2	U	0
2	[illegible]	U	0
2	12	U	0
5	1	O	1
14	1	U	0
18	1	U	0
13	1	O	1
1	[illegible]	U	0
12	[illegible]	A	64
Total Relationship Score			83

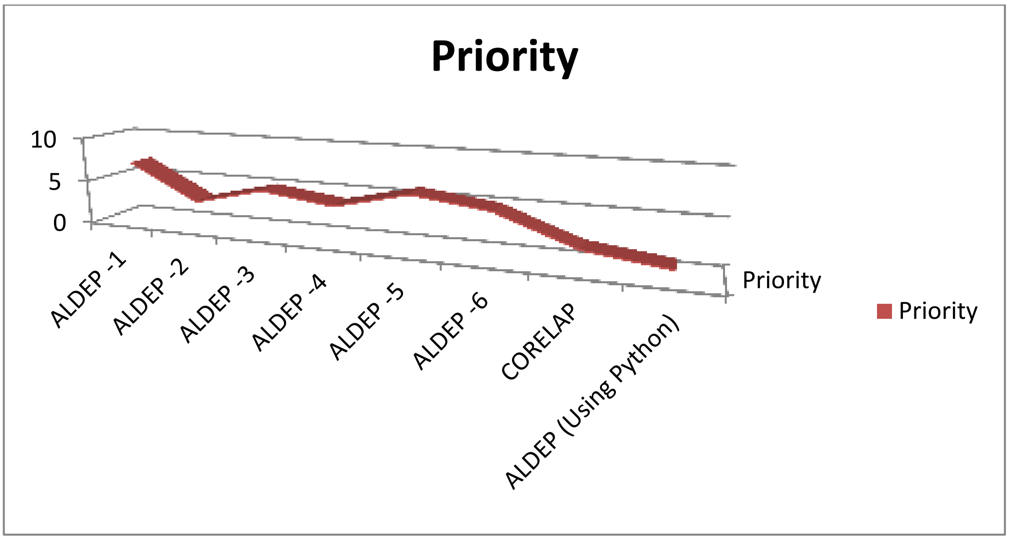
Adjacent station		Relationship	Relationship score
[illegible]	7	U	0
17	15	U	0
17	16	A	64
16	[illegible]	U	0
16	11	U	0
15	13	E	16
7	[illegible]	A	64
[illegible]	4	O	1
13	2	E	16
4	5	A	64
[illegible]	14	U	0
[illegible]	[illegible]	U	0
14	[illegible]	O	1
5	1	O	1
2	1	A	64
2	13	E	16
[illegible]	18	U	0
11	12	A	64
13	12	A	64
13	1	O	1
1	9	U	0
1	[illegible]	I	4
Total Relationship Score			440

10. Result:

Sr. No.	Proposed Options	Relationship score	Distance per trip	Priority
1	ALDEP -1	89	1483.14	7
2	ALDEP -2	169	1371.76	3
3	ALDEP -3	127	1437.51	5
4	ALDEP -4	148	1388.59	4
5	ALDEP -5	94	1455.69	6
6	ALDEP -6	136	1428.46	5
7	CORELAP	440	1328.51	2
8	ALDEP (Using Python)	670	1126.98	1

11. Conclusion:

Results obtained showcase the comparative performance of ALDEP and CORELAP while highlighting the hybrid approach's ability to produce a wide array of layout alternatives. The evaluation metrics include material handling cost reduction, distances traveled per trip, and solution quality. The outcomes of this study provide valuable insights for practitioners and researchers in the field of plant layout optimization. By leveraging the strengths of ALDEP, CORELAP, and Python programming, study offers a holistic approach to address complex layout challenges, ultimately contributing to improved operational efficiency and cost savings in manufacturing facilities.



References

1. James A. Tompkins, John A. White, Facilities Planning, J. Wiley & Sons, Inc., 3rd Edition, New York.
2. Production and Operations Management, Panneerselvam R, Publisher: PHI
3. Buffa, Modern Production/Operations Management, Wiley Eastern Ltd.
4. Plant layout & Material Handling – by Prof. G.K. Agrawal
5. R. Muther, “Practical Plant Layout”, 1st Edition, McGraw Hill Book Company Inc., New-York, 1955.
6. J.M. Moore, “Plant Layout and Design”, Macmillan Publishing Co. Inc., New-York, 1962.
7. J.M. Apple, “Plant Layout and Material Handling”, 3rd edition, Wiley, New AYork, 1977.
8. Gordon C. Armour, Elwood S. Buffa, (1963) A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities, Management Science 9(2):294-309
9. Afentakis, P. A. (1989). Loop layout design problem for flexible manufacturing systems. International Journal of Flexible Manufacturing Systems, 1, 143–175.
10. Aiello, G., & Enea, M. (2001). Fuzzy approach to the robust facility layout in uncertain production environments. International Journal of Production Research, 39(18), 4089–4101.

11. Aiello, G., Enea, M., & Galante, G. (2002). An integrated approach to the facilities and material handling system design. International Journal of Production Research, 40(15), 4007–4017.
12. Aiello, G., Enea, M., & Galante, G. (2006). Multi-objective approach to facility layout problem by genetic search algorithm and Electre method. Robotics and Computer-Integrated Manufacturing, 22, 447–455.
13. Al-Hakim, L. (2000). On solving facility layout problems using genetic algorithms. International Journal of Production Research, 38(11), 2573– 2582.
14. Aleisa, E. E., & Lin, L. (2005). For effectiveness facilities planning: Layout optimization then simulation, or vice versa? In Proceedings of the 2005 Winter Simulation Conference.
15. Armour, G. C., & Buffa, E. S. (1963). A heuristic algorithm and simulation approach to relative allocation of facilities. Management Science, 9(2), 294–300.
16. Asef-Vaziri, A., & Laporte, G. (2005). Loop based facility planning and material handling. European Journal of Operational Research, 164(1),1–11.
17. Azadivar, F., & Wang, J. (2000). Facility layout optimization using simulation and genetic algorithms. International Journal of Production Research, 38(17), 4369–4383.
18. Balakrishnan, J., & Cheng, C. H. (1998). Dynamic layout algorithms: A state- of-the-art survey. Omega, 26(4), 507–521.
19. Balakrishnan, J., & Cheng, C. H. (2000). Genetic search and the dynamic layout problem. Computers & Operations Research, 27(6), 587–593.
20. Balakrishnan, J., Cheng, C. H., & Wong, K. F. (2003a). FACOPT: A user friendly facility layout optimization system. Computers & Operations Research, 30(11), 1625–1641
21. Balakrishnan, J., Cheng, C. H., Conway, D. G., & Lau, C. M. (2003b). A hybrid
22. genetic algorithm for the dynamic plant layout problem. International Journal of Production Economics, 86(2), 107–120.
23. Balakrishnan, J., Robert Jacobs, F., & Venkataramanan, M. A. (1992). Solutions for the constrained dynamic facility layout problem. European Journal of Operational Research, 57(2), 280–286.
24. Banerjee, P., & Zhou, Y. (1995). Facilities layout design optimization with single loop material flow path configuration. International Journal of Production Research, 33(1), 183–204.
25. Baykasoglu, A., Dereli, T., & Sabuncu, I. (2006). An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems. Omega, 34(4), 385–396.
26. Baykasoglu, A., & Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic layout problem. Computers & Operations Research, 28(14), 1403–1426.
27. Benjaafar, S., Heragu, S. S., & Irani, S. A. (2002). Next generation factory layouts: Research challenges and recent progress. Interface, 32(6), 58–76. Bock, S., & Hoberg, K. (2007).
28. Detailed layout planning for irregularly-shaped machines with transportation path design. European Journal of Operational Research, 177, 693–718.
29. Bozer, Y. A., Meller, R. D., & Erlebacher, S. J. (1994). An improvement-type layout algorithm for single and multiple floor facilities. Management Science, 40(7), 918–932.
30. Braglia, M. (1996). Optimization of a simulated-annealing-based heuristic for single row machine layout problem by genetic algorithm. International Transactions in Operational Research, 3(1), 37–49.
31. Braglia, M., Zanoni, S., & Zavanella, L. (2003). Layout design in dynamic environments: Strategies and quantitative indices. International Journal of Production Research, 41(5), 995–1016.
32. Chaieb, I.(2002).Conception et exploitation des system's production flexible manufacturer's: Introduction des taˆ ches de transport. Ph.D. dissertation (in French). France: Specialite´ en productive automatique et informatique industrially, Ecole central de Lille.
33. Chen, C. W., & Sha, D. Y. (2005). Heuristic approach for solving the multi- objective facility layout problem. International Journal of Production Research, 43(21), 4493–4507.
34. Chen, D. S., Wang, Q., & Chen, H. C. (2001). Linear sequencing for machine layouts by a modified simulated annealing. International Journal of Production Research, 39(8), 1721–1732.
35. Cheng, R., & Gen, M. (1998). Loop layout design problem in flexible manufacturing systems using genetic algorithms. Computers & Industrial Engineering, 34(1), 53–61.
36. Cheng, R., Gen, M., & Tosawa, T. (1996). Genetic algorithms for designing loop layout manufacturing systems. Computers & Industrial Engineering, 31(3–4), 587–591.
37. Chiang, W. C., & Kouvelis, P. (1996). An improved tabu search heuristic for solving facility layout design problems. International Journal of Production Research, 34(9), 2565–2585.
38. Chittratanawat, S., & Noble, J. S. (1999). An integrated approach for facility layout, P/D location and material handling system design. International Journal of Production Research, 37(3), 683–706.
39. Chung, Y. K. (1999). A neuro-based expert system for facility layout construction. Journal of Intelligent Manufacturing, 10(5), 359–385.
40. Chwif, L., Pereira Barretto, M. R., & Moscato, L. A. (1998). A solution to the facility layout problem using simulated annealing. Computers in Industry,36(1–2), 125–132.Co, H. C., Wu, A., & Reisman, A. (1989).
41. A throughput-maximizing facility planning and layout model. International Journal of Production Research,27(1), 1–12.
42. Das, S. K. (1993). A facility layout method for flexible manufacturing systems. International Journal of Production Research, 31(2), 279–297.
43. Deb, S. K., & Bhattacharyya, B. (2005). Fuzzy decision support systems for manufacturing facilities layout planning. Decision Support Systems, 40,305–314.

44. Devise, O., & Pierreval, A. (2000). Indicators for measuring performances of morphology and materials handling systems. International Journal of Production Economics, 64(1–3), 209–218.Dilworth, J. B. (1996). Operation management. McGraw Hill. Djellab, H., & Gourgand, A. (2001).
45. A new heuristic procedure for the single- row facility layout problem. International Journal of Computer Integrated Manufacturing, 14(3), 270–280.
46. Drezner, Z. (1987).
47. A heuristic procedure for the layout of a large number of facilities. International Journal of Management Science, 33(7), 907–915.
48. Dunker, T., Radonsb, G., & Westka¨mpera, E. (2003). A co evolutionary algorithm for a facility layout problem. International Journal of Production Research, 41(15), 3479–3500.Dunker, T., Radonsb, G., & Westka¨mpera, E. (2005).
49. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. European Journal of Operational Research, 165(1), 55–69. Dweiri, F., & Meier, F. A. (1996).
50. Application of fuzzy decision-making in facilities layout planning. International Journal of Production Research, 34(11), 3207–3225.El-Baz, M. A. (2004).

Annexure:

Python programming code:

```
# Import necessary libraries
import random

# Define parameters
num_iterations = 100000  # Number of iterations for optimization
facility_count = 19   # Number of facilities or machines
layout = []           # Initialize an empty layout

# Generate an initial random layout
for i in range(facility_count):
    layout.append({
        'x': random.uniform(0, 100),  # Random x-coordinate
        'y': random.uniform(0, 100)   # Random y-coordinate
    })

# Define a function to calculate the total distance of material movement
def calculate_total_distance(layout):
    total_distance = 0
    for i in range(len(layout)):
        for j in range(i + 1, len(layout)):
            distance = ((layout[i]['x'] - layout[j]['x']) ** 2 +
                        (layout[i]['y'] - layout[j]['y']) ** 2) ** 0.5
            total_distance += distance
    return total_distance

# Main optimization loop
for iteration in range(num_iterations):
    # Randomly select two facilities to exchange
    facility_a, facility_b = random.sample(range(facility_count), 2)

    # Create a new layout by exchanging the positions of facility_a and facility_b
    new_layout = layout.copy()
    new_layout[facility_a], new_layout[facility_b] = new_layout[facility_b], new_layout[facility_a]

    # Calculate the total distance of material movement for the new layout
    new_distance = calculate_total_distance(new_layout)

    # If the new layout is better (reduces total distance), accept it as the current layout
    if new_distance < calculate_total_distance(layout):
        layout = new_layout

# Print the final optimized layout
for i, facility in enumerate(layout):
    print(f'Facility {i+1}: ({facility["x"]}, {facility["y"]})')

# Calculate and print the total distance of material movement for the final layout
final_distance = calculate_total_distance(layout)
 print(f'Total Distance: {final_distance}')
```