IJSES

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

Selection and Sequencing of Part Families in Reconfigurable Manufacturing Systems

Eswara Satyanarayana G[1], B Anjaneya Prasad[2]
[1]Sr Manager. Contracts Management, BHL, Hderabad, Telangana, India
[2]Professor, Department of Mechanical Engineering, Jawaharlal Nehru Technological University Hyderabad, Hyderabad, Telangana, India
Corresponding mail id: luckysridhar08@gmail.com

Abstract— *Modern manufacturing companies face highly volatile and unpredictable market conditions driven by global competition and rapid technological change. Customers increasingly demand customized, technologically advanced, and functionally complex products. To remain competitive and profitable, industries must constantly adapt to shifting social, economic, and technological landscapes, emerging markets, and evolving customer preferences. These pressures have pushed manufacturing systems to evolve continuously, and this trend is expected to accelerate. Traditional systems—such as dedicated, cellular, and flexible manufacturing systems—are no longer capable of meeting these dynamic requirements. This challenge has led to the development of the Reconfigurable Manufacturing System (RMS)****,*** *designed to rapidly adjust production capacity and functionality. A key requirement in RMS is the identification of suitable part families, since each family is produced using an appropriately configured Reconfigurable Machine Tool (RMT). The system must be reconfigured before processing the next family, making accurate part family formation critical. Part families are recognized using Group Technology, which identifies similarities among parts based on design or manufacturing attributes. Similarity is measured using Jaccard's coefficient****,*** *and the resulting matrix is clustered using Agglomerative Average Linkage Clustering to generate a dendrogram of possible families. Costs are evaluated at each dendrogram level, and the grouping with the lowest cost is selected. The sequencing problem is formulated as a Traveling Salesman Problem and solved using the Ant Colony Optimization—Ant Colony System to minimize reconfiguration and idle machine costs.*

I. INTRODUCTION

In today's rapidly changing markets, customers demand not only low-cost products but also high quality, advanced technology, and extensive customization. Global competition forces companies to offer diverse product models, frequent upgrades, and individualized features while maintaining cost efficiency. This shift has led to the emergence of mass customization (MC), a manufacturing paradigm that satisfies individual needs with the efficiency of mass production. Implementing MC requires managing complex customer choices, achieving economies of scale, and ensuring high manufacturing flexibility and responsiveness.

Traditional manufacturing systems struggle with these requirements because any change in factory operations generally increases resource consumption. As product variety expands, system reconfiguration becomes essential for launching new products. Reconfigurable Manufacturing Systems (RMS) meet MC needs by offering productivity along with the ability to rapidly change system configuration. RMS relies on CNC machines and Reconfigurable Machine Tools (RMTs), modular machines whose structures can be adjusted as needed.

A crucial aspect of RMS design is recognizing suitable part families, since each family requires a specific RMS configuration. Only one family is produced at a time, and the system is reconfigured before switching to the next. This classification is achieved through Group Technology (GT), which identifies similarities in design and manufacturing attributes, reducing the number of required reconfigurations.

Traditional Cellular Manufacturing Systems (CMS) use GT to form permanent cells, but these are inefficient under unpredictable market conditions. RMS overcomes this by allowing each family to be produced under a different configuration while using GT to minimize reconfigurations.

This work applies the Hierarchical Agglomerative Average Linkage Clustering Algorithm (HAALCA), using a similarity matrix based on Jaccard's similarity coefficient, to generate a dendrogram of possible part families. Each dendrogram level is evaluated based on (i) reconfiguration sequence cost and (ii) machine idle cost. The least-cost family grouping is selected.

The sequencing problem is modeled as a Traveling Salesman Problem (TSP), aiming to minimize total reconfiguration cost. Because TSP is NP-complete, the Ant Colony Optimization – Ant Colony System (ACS) is proposed to obtain efficient reconfiguration sequences for the RMS model.

$$S_{pq} = \frac{a}{a+b+c} \quad 0 \le S_{pq} \le 1 \tag{1}$$

In this expression, a represents machines visiting both parts p and q, b those visiting only p, and c those visiting only q. Thus, Spq = 1 indicates identical machine requirements, while Spq = 0 means completely different routes. Parts with the highest similarity values are grouped together. In part-family or cell formation, agglomerative clustering methods are commonly used, though they tend to produce "chaining," creating large clusters and leaving some parts ungrouped. Among them, the Average Linkage Clustering Algorithm (ALCA) shows the least chaining effect. It groups parts with high similarity first, then recalculates similarities as average values for subsequent clustering.

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

$$S_{ij} = \frac{\sum_{p \in i} \sum_{q \in j} S_{pq}}{N_i . N_j} \qquad (2)$$

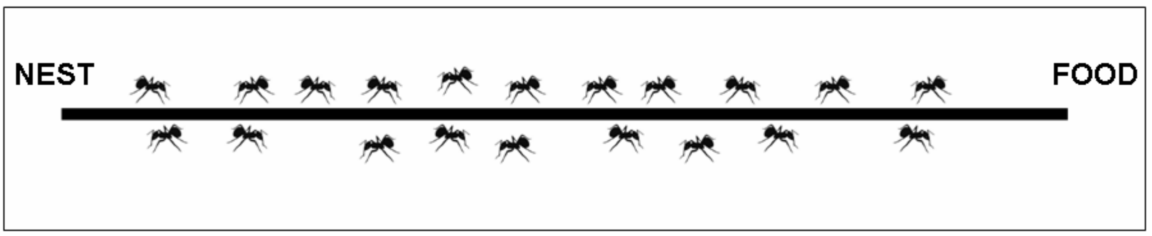
Spq coefficient of similarity between parts p and q, N_i, N_j number of parts in family i and j, respectively, Only limited research exists on modeling Reconfigurable Manufacturing Systems (RMS), though interest is increasing as RMS concepts gain importance. Because RMS operates in environments filled with uncertainty, stochastic modeling is essential. Zhao Xiabo et al. [25] proposed a stochastic framework addressing optimal configuration design, selection policies, and performance measures; however, reconfiguration time was not included. Mustapha Nourelfath et al. [24] studied RMS optimization under reliability considerations, defining RMS as a system capable of maintaining service continuity at reduced functionality when components fail. They examined two key problems: maximizing production rate under resource constraints and achieving a target production rate at minimum cost.

Abdi and Labib [1] introduced a methodology for grouping products into families before machine selection using the Analytical Hierarchy Process (AHP), balancing both market and manufacturing needs. Validation was carried out using an industrial case. Hegui Ye and Ming Liang [19] developed a genetic algorithm (GA) with a parallel chromosome structure to integrate product scheduling and production cell configuration in RMS, demonstrating improved performance over traditional GA methods. Their later work [20] presented an integrated model for scheduling modular product operations and selecting system configurations to minimize reconfiguration, idle, handling, and WIP costs, showing superior results to LINGO.

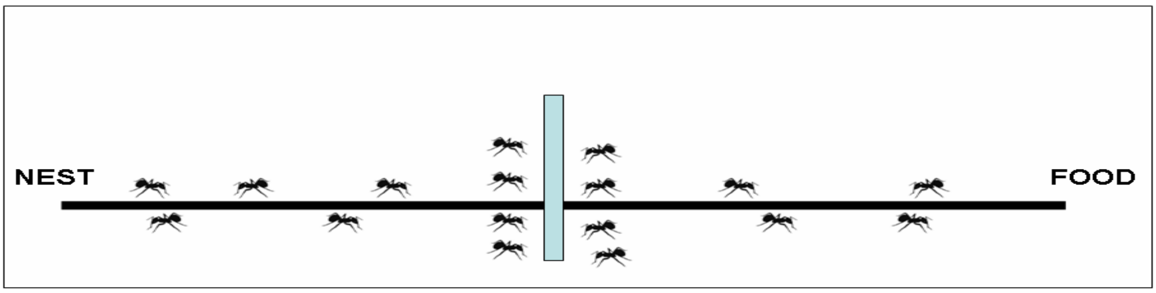
Ayman M. A. and Hoda A. ElMaraghy [2] proposed an RMS configuration selection approach involving two phases: first, generating near-optimal configurations using real-coded GAs and Tabu Search; second, selecting configurations that maximize reconfiguration smoothness using integer-coded techniques. R. Galan et al. [10] emphasized the importance of forming suitable product families for effective RMS operation. Their methodology evaluates modularity, commonality, compatibility, reusability, and demand using AHP and applies Average Linkage Clustering to form dendrogram-based family groupings. Another study by Galan et al. [11] presented an ALCA-based RMS reconfiguration framework for adapting production systems to new product requirements.

Research Related to Ant Colony Optimization for TSP (Compressed) The Traveling Salesman Problem (TSP) seeks the shortest Hamiltonian cycle through a set of cities and is widely used as a benchmark in combinatorial optimization. Many practical applications—from vehicle routing to PCB drilling—can be modeled as TSP variants. Reinelt [13] highlighted its broad relevance. Ant Colony Optimization (ACO), introduced by Dorigo and Stutzle [7], was first tested on TSP due to its simplicity and importance. Real ant behavior—pheromone communication, path reinforcement through positive feedback, and collective intelligence—inspired the design of ACO algorithms, which have proven highly effective for TSP and many related problems.

1.1 Characteristics of Ant Colony Metaheuristics (Ant colony metaheuristics exhibit several useful properties [6]. They are versatile and easily adaptable to variations of a problem, such as extending the TSP to its asymmetric version. They are also robust, requiring minimal structural changes when applied to other combinatorial optimization problems like the quadratic assignment or job-shop scheduling problems. Being population-based, the approach benefits from positive feedback, enabling rapid reinforcement of promising solution paths.



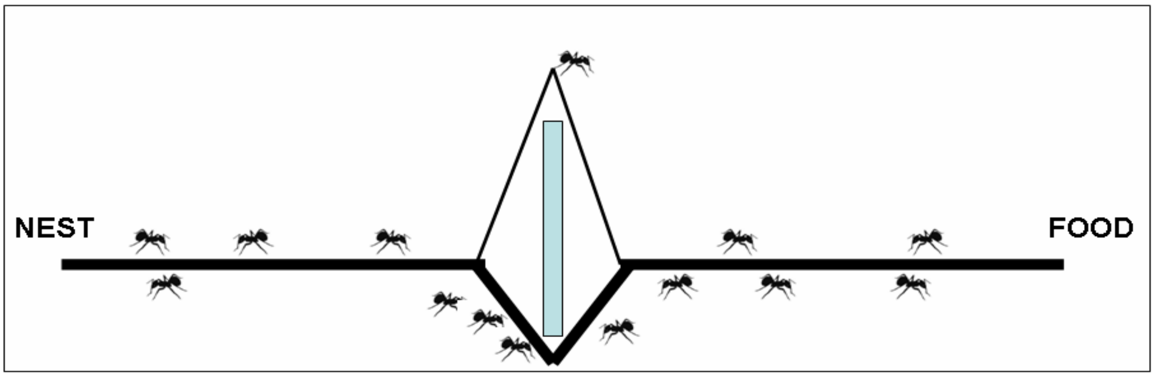
(1) At the beginning, all the ants follow their trail normally



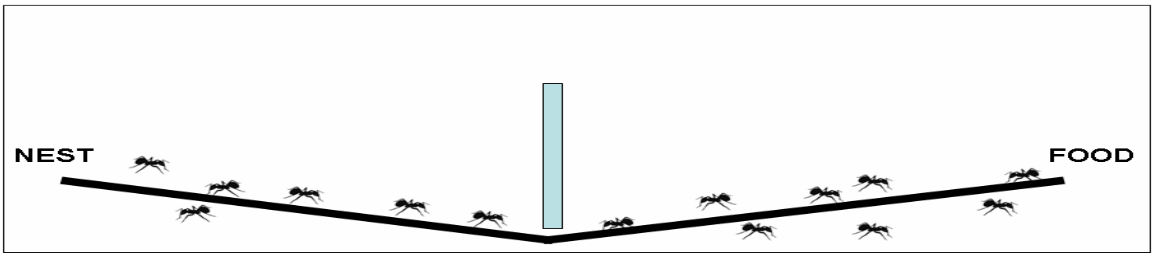
(2) At the time an obstacle is inserted; the ants decide which way to take on a random basis. With p=0.5 they turn right, with p=0.5 they turn left

http://ijses.com/

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361



(3) After some time the shorter path is preferred. However, a few ants take the longer path



(4) After a long time the trail is changed to the shortest path
Figure 1 (1-4). Naturally Observed Behavior of Ants [4]

1.2 ACO Algorithms for TSP Applying ACO to the TSP is straightforward because the construction graph mirrors the problem graph. Ants generate feasible tours by visiting each city once and returning to the start. Pheromone trails are linked to arcs, with trail intensity indicating the desirability of moving from city i to city j. Heuristic information is typically the inverse of inter-city distance. Ants probabilistically construct tours by choosing a starting city, selecting unvisited cities based on pheromone and heuristic values, and completing a full cycle. After tour construction, pheromone is deposited on the arcs used, optionally after local search refinement. Most ACO variants follow this general structure except Ant Colony System (ACS), which updates pheromone during tour construction.

Figure outlines the ACO framework: after parameter initialization, ants iteratively construct tours, optionally improve them, and then update pheromones through evaporation and deposition based on search experience. The original Ant System (AS) [4] performed well initially but was surpassed by extensions such as elitist AS, rank-based AS, MAX-MIN AS, Ant-Q, ACS, ANTS, and the hyper-cube framework. These algorithms differ mainly in pheromone update strategies.

In AS, m ants build tours using the probabilistic “random proportional rule,” where selection depends on pheromone intensity and heuristic distance. Parameters α and β control their relative influence, and memory is maintained to track visited cities and compute tour length.

1.3 Ant Colony System (ACS) ACS improves on AS in three ways: it uses a more aggressive action-choice rule, updates pheromone only on the best-so-far path, and applies local pheromone reduction each time an arc is traversed. Tour construction uses the pseudorandom proportional rule, balancing exploitation and exploration through parameter q_0. Global pheromone updating reinforces only arcs in the best-so-far tour, reducing computational load. Local updating decreases pheromone on used arcs to encourage exploration and prevent stagnation.

ACS implementations commonly use parallel tour construction and candidate lists to reduce computational effort and improve solution quality, especially for large TSP instances. Experimental studies confirm ACS as one of the most effective ACO variants.

1.4 The Problem Formulation:

The problem in hand involves the development of a detailed planning for a manufacturing/reconfiguration cycle of the proposed RMS. It is essentially consisting of two sub problems:

(1) Recognition of part families (chapter 4) and
(2) Generation of the manufacturing cycle plan (i.e. final sequencing and selection of part families) (Chapter 5)

1.5 The Proposed Methodology: Reconfiguration Cost Calculations

Finding out the cost of reconfiguration while switching over the manufacturing from one part family (say family A) to the next (say family B), a “divide and defeat” strategy has been proposed. It is proposed here that the task of reconfiguring the RMS can be decomposed into eight different tasks (table 1) each of which is described as given below.

(1) Task “a”: An RMT is required in the next configuration without any significant modification.

(2) Task “b”: An existing RMT is required partially for reconfiguration of a new RMT. In fact the reconfiguration task ‘b’ corresponds to a situation where the BM and at least one of the AMs of an existing RMT are retained for reconfiguration of a new RMT.

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

(3) Task "c": A basic module removed from an RMT in the existing RMS to be reutilised for the reconfiguration of a new RMT.

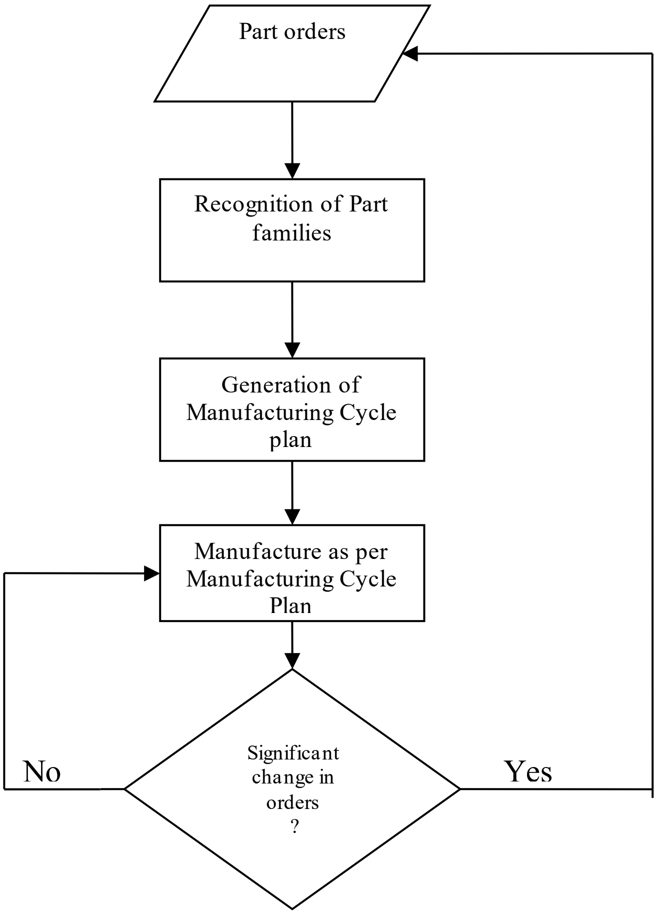


Figure 2. Overall Strategy for the RMS Manufacturing Cycle Planning

(4) Task "d": A basic module removed from an RMT in the existing RMS is not required for reconfiguration of any new RMT. Therefore, we need to move it to the store.
(5) Task "e": A basic module required for the configuration of an RMT need to be brought from the store.
(6) Task "f": An auxiliary module removed from an RMT in the existing RMS to be reutilised for the reconfiguration of a new RMT.
(7) Task "g": An auxiliary module removed from an RMT in the existing RMS is not required for reconfiguration of any new RMT. Therefore, we need to move it to the store.
(8) Task "h": An auxiliary module required for the configuration of an RMT need to be brought from the store.

1.6 Algorithm for finding the number of times each reconfiguration task is required during part family switchover

Step 1: For finding the number of times each reconfiguration task is required while switching over the manufacturing from one family (say A) to another family (say B), a working table as shown in table 1 is formed. In this table column 2 and column 4 have binary entries to represent whether the operations shown in first column are required by family A and B respectively. Column 3 and column 5, contains the RMT vectors corresponding to each 1 entry in column 2 and column 4 respectively. All other cells in column 3 and column 5 are left blank. The length of each RMT vector is made equal to that of the one which has largest number of modules by adding zeroes at the end.

TABLE 1. Working table for calculation of no. of reconfiguration Tasks for switching over from family **A** to family **B**

Column 1 Operation	Column 2 Family A	Column 3 RMT of A	Column 4 Family B	Column 5 RMT of B
1	1	[1 1 0 0]	0	
2	**1**	[2 3 0 0]	**1**	[2 3 0 0]
3	1	[4 7 0 0]	0	
4	1	[3 12 9 0]	0	
5	**1**	[1 2 3 0]	**1**	[1 2 3 0]
6	0		1	[5 11 1 0]
7	0		1	[4 7 5 0]
8	**0**		**0**	
9	0		1	[4 2 10 0]
10	0		0	
11	0		0	
12	1	[5 8 3 2]	0	

Step 2: *Finding number of "a"*

Find number of 1-1 matches in columns 2 and 4. This is equal to "a". In the example taken, a = 2 (shown by bold 1s).
Step 3: Update the table eliminating the rows corresponding to the 1-1 matches found in the last stage as well as all 0-0 matches, if there.

TABLE 2. Update of working table 1

Column 1 Operation	Column 2 Family A	Column 3 RMT of A	Column 4 Family B	Column 5 RMT of B
1	1	[1 1 0 0]	0	
3	1	[4 7 0 0]	0	
4	1	[3 12 9 0]	0	
6	0		1	[5 11 1 0]
7	0		1	[4 7 5 0]
9	0		1	[4 2 10 0]
12	1	[5 8 3 2]	0	

Step 3: *Finding number of "b"*
Numbers of "b" are calculated from the updated table using following algorithm:

Let 'u' be the length of the vectors. Let number of vectors in column 2 and column 4 is 'v' and 'w' respectively. Let i and j are indices for vectors in column 2 and column 4 respectively.

(i) b = 0, , k = u
(ii) i = 1, j = 1
(iii) Consider 'ith' vector in column 3 and check whether its first entry is a zero. If yes go to step (v). Otherwise go to next step.
(iv) Compare 'ith' vector's first k-1 entries with first k-1 entries of all the w vectors in column 5. Whenever first such match is found, b = b + 1, and update the working table so that all the matching entries in both the columns are replaced by 0s and then go to next step. Otherwise go to next step directly.
(v) i = i +1 and check whether i <= v. If yes go to step (iii). Otherwise go to next step.

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

(vi) k = k-1 and check if k >1. If yes go to step (ii). Otherwise end.

In the example considered, u = 4, v = 4, w=3 (refer table 2). First, first 3 (u-1) entries of first vector in the column 3 are compared with first three entries of all the vectors (w) in column 5. No match is found. Then first 3 (u-1) entries of second vector in the column 3 are compared with first three entries of all the vectors (w) in column 5. Again no match is found. The procedure is repeated for all the 4 vectors of the column 3. In this case no match is observed in all these cases. After that the process is again repeated by considering first two entries of all the vectors and a match is found between the second vector of column 3 and second vector of column 5. Therefore, the working matrix is updated as shown in table 5. The process is repeated till k = 2. In this example, b = 1.

TABLE 3. Update of working table 2

Column 1 Operation	Column 2 Family A	Column 3 RMT of A	Column 4 Family B	Column 5 RMT of B
1	1	[1 1 0 0]	0	
3	1	[0 0 0 0]	0	
4	1	[3 12 9 0]	0	
6	0		1	[5 11 1 0]
7	0		1	[0 0 5 0]
9	0		1	[4 2 10 0]
12	1	[5 8 3 2]	0	

Step 4: *Finding number of "c", "d" and "e"*

For determining the number of "c", "d" and "e" a methodology based on philosophy of sets has been proposed. From the updated table 4, following sets are formed with regard to the basic modules.

IBM: It represents a set of all basic modules in column 3 present in updated table 4.

FBM: It represents a set of all basic modules in column 5 present in updated table 4.

In this example IBM = [1 3 5] and FBM = [5 4].

The following situations may arise with these sets:

(i) the two sets are non intersecting,

(ii) the two sets are intersecting.

In the second case the number of modules in the intersection portion of IBM and FBM represents number of reconfiguration task "c" i.e. the number of BMs to be reutilized in configuring new RMTs for new RMS, whereas it is '0' for the first case. The number of modules in non-intersecting portions of both the sets represents the number of tasks "d" and "e" respectively.

For the given example, therefore

$$c = (IBM) \cap (FBM) = 1$$
$$d = IBM - FBM = 3 - 1 = 2$$
$$e = FBM - IBM = 2 - 1 = 1$$

Step 5: *Finding number of "f", "g" and "h"*

Similarly, for determining the number of "f", "g" and "h" a methodology based on philosophy of sets has been proposed. From the updated table 3, following sets are formed with regard to the auxiliary modules.

IAM: It represents a set of all auxiliary modules in column 3 present in updated table 3.

FBM: It represents a set of all auxiliary modules in column 5 present in updated table 3.

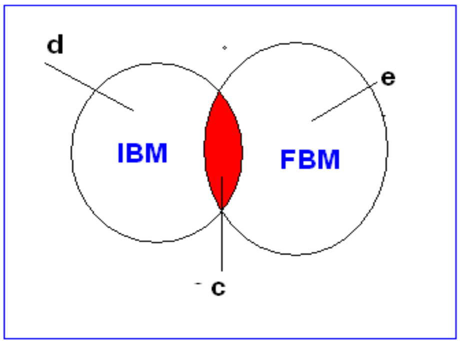


Figure 3. Determining numbers of tasks "c", "d" and "e"

In this example IAM = [1 12 9 8 3 2] and FAM = [11 1 5 2 10].

The following situations may arise with these sets:

(i) the two sets are non intersecting,

(ii) the two sets are intersecting.

In the second case the number of modules in the intersection portion of IAM and FAM represents number of reconfiguration task "f" i.e. the number of AMs to be reutilized in configuring new RMTs for new RMS. Whereas, it is '0' for the first case. The number of modules in non-intersecting portions of both the sets represents the number of tasks "g" and "h" respectively.

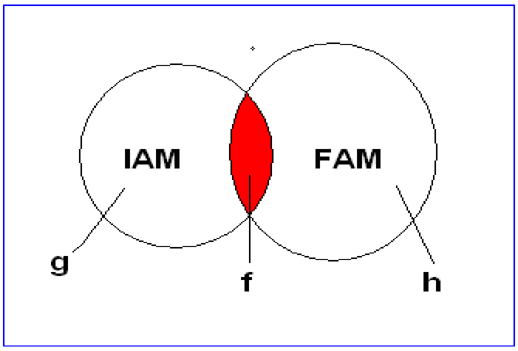


Figure 4. Determining numbers of tasks "f", "g" and "h"

For the given example, therefore

$$f = (IAM) \cap (FAM) = 2$$
$$g = IBM - FBM = 6 - 2 = 4$$
$$h = FBM - IBM = 5 - 2 = 3$$

1.7 Finding the reconfiguration cost incurred during a part family switchover:

Consolidated information on the number of times each task is required while switching over the manufacturing from one family to the next by applying the procedure described in section and referring the table 4, is compiled as table 4. The total reconfiguration cost is simply calculated as the summation of products of unit cost of each task and number of times it is required. For the example taken, it has been found to be equal to 51 for switchover of the manufacturing from part family A to part family B.

1.8 The Proposed Methodology: Finding Least Cost Reconfiguration Sequence for a set of families

1. To illustrate the methodology, consider four part families—A, B, C, and D—to be produced in one manufacturing cycle

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

(level 1 of the dendrogram). Since the reconfiguration cost from one family to another is direction-dependent (for example, A→B differs from B→A), these four families can be arranged in 4! = 24 possible sequences, each resulting in a different total reconfiguration cost. In general, for N families, there are N! possible sequences; hence identifying the least-cost sequence is essential for economical RMS operation.

TABLE 4. Reconfiguration cost calculation for the switchover of part family A to B

Reconfiguration task	Unit cost (UC)	No. of times task required (N)	Cost (RC= UC X N)
a	0	2	0
b	1	1	1
c	6	1	6
d	5	2	10
e	8	1	8
f	3	2	6
g	2	4	8
h	4	3	12
Total cost			51

2. This sequencing problem is modeled as a Traveling Salesman Problem (TSP), assuming production orders repeat after each cycle. In this analogy, part families represent cities, and reconfiguration costs represent distances. The objective is to minimize total reconfiguration cost, similar to minimizing travel distance in TSP. As both problems are NP-complete, the proposed solution applies the Ant Colony Optimization—Ant Colony System (ACS) algorithm.
3. Algorithm
4. Step 1: Compile the Process–Operation Matrix (POM) for the selected families.

Step 2: Construct the Reconfiguration Cost Matrix (RCM). For each pair of families (i, j):

- If i = j, set the cost to zero.
- Otherwise, determine the required reconfiguration tasks and compute the corresponding cost.

This process is repeated for all N × N family pairs to complete the RCM.

Step 3: Using the RCM, solve the TSP-modeled sequencing problem with ACS to obtain the least-cost reconfiguration sequence.

TABLE 5. Part-family Operation-group Matrix (POM)

Part family \operation	1	2	3	4	5	6	7	8	9	10	11	12
A	1	1	1	1	1	0	0	0	0	0	0	1
B	0	1	0	0	1	1	1	0	1	0	0	0
C	0	0	0	0	1	1	1	0	1	0	0	1
D	0	1	0	0	1	0	0	1	0	1	1	0

TABLE 6. Reconfiguration cost matrix (RCM) among families (level 1)

	A	B	C	D
A	0	51	57	69
B	56	0	24	69
C	58	20	0	89
D	74	69	93	0

1.9 Ant Colony System (ACS) Algorithm

This study applies Ant Colony Optimization (ACO), specifically the Ant Colony System (ACS), to determine the least cost reconfiguration sequence for part families in a Reconfigurable Manufacturing System (RMS). The sequencing problem is analogous to an Asymmetric Traveling Salesman Problem (TSP), where part families act as cities and reconfiguration costs act as distances. Since the objective is to minimize total reconfiguration cost, ACS is applied at each dendrogram level to evaluate all possible sequences. A heuristic desirability factor is used, defined as the reciprocal of reconfiguration cost between two families (1/RCij). The probability of selecting the next part family is computed from pheromone levels and heuristic values using standard ACS rules. Parameter values adopted include $\alpha = 1$, β between 2–5, $\rho = 0.1$, m = 10, and 30 iterations.

Several assumptions consistent with typical ACO applications are used: each ant can generate a complete solution, ants select paths probabilistically, pheromone intensity reflects solution quality, and ants are restricted from revisiting families already in the sequence. The distance traveled by each ant corresponds to the total reconfiguration cost of the sequence generated.

The ACS implementation proceeds as follows. First, parameters are initialized and a set of artificial ants is generated, each with a tabu list used to record the sequence of visited families. Initial pheromone values on all edges are set to a constant. In each iteration, ants construct complete family sequences. For each ant, the next family is selected based on the highest probability calculated from the pheromone intensity and heuristic desirability. Once all ants complete their sequences, the corresponding reconfiguration costs are computed.

Next, pheromone levels are updated. Additional pheromone is deposited on edges belonging to the best sequence, with the amount inversely proportional to total cost. Simultaneously, pheromone evaporation is applied to prevent early convergence and allow exploration. At the end of each iteration, tabu lists are cleared, and pheromone changes are reset. The process repeats until the maximum number of iterations is reached or stagnation occurs. For the example considered (families A, B, C, D), the final ACS output after 30 iterations yields the optimal sequence A–D–B–C with a minimum reconfiguration cost of 220.

Machine idle cost for each part family is computed separately using a structured working table. Demand volumes, required operations, unit idle costs, and the number of idle occurrences per operation are compiled. Summing idle costs for all operations provides the total machine idle cost, as demonstrated for family BC, where the calculated idle cost is 65.

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{y \in M_i^a} [\tau_{iy}]^{\alpha} * [\eta_{iy}]^{\beta}} & \text{if } j \in M_i^k \\ 0 \quad \text{otherwise} & \end{cases} \tag{3}$$

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025.
ISSN (Online): 2456-7361

TABLE 7. Ant variable with completed tabu list

a_{ks}	s			
		1	2	3	4
k	**1**	4	2	1	3
	2	3	2	4	1
	3	2	1	4	3
	4	1	4	2	3

Table 8. Part family sequence and corresponding reconfiguration costs

a_{ks}		s				
		1	**2**	**3**	**4**	**RC**
k	**1**	4	2	1	3	**271**
	2	3	2	4	1	**221**
	3	2	1	4	3	**238**
	4	1	4	2	3	**220**

TABLE 9. Working table for calculation of machine idle cost for family **BC**

Part (i)	**Demand (d_i)**	**Operation/RMT Required, j**					
		2	**5**	**6**	**7**	**9**	**12**
B	50	1	1	1	1	1	0
C	100	0	1	1	1	1	1
No. of times RMT is idle (n_j)		100	0	0	0	0	50
Unit machine idle cost (mic_j)		0.2	0.7	0.6	0.3	0.2	0.9
Machine idle cost(MIC_j)		20	0	0	0	0	45
Total machine idle cost (MIC_{BC})							**65**

modeled as TSP and is solved by ACS (section 5.4.2). Table 10 gives the least cost reconfiguration sequences for all dendogram levels of the example problem.

TABLE 10. Least cost reconfiguration sequence for illustrated example

Dendogram level (% Similarity)	**Part families**	**Least cost reconfiguration sequence**	**Least reconfiguration cost**
1 (100)	[A] [B] [C] [D]	A-D-B-C	220
2 (67)	[A] [BC] [D]	A-D-BC	204
3 (22)	[AD] [BC]	AD-BC	165
4 (20)	[ADBC]	ADBC	0

(iii) find out the total machine idle cost. Section 5.5 gives a detailed methodology to find out the machine idle cost for a family. Therefore, the total machine idle cost corresponding to a dendogram level can be determined by summing up the machine idle cost of its each family. Table 11 gives the total machine idle cost all dendogram levels of the example problem.

TABLE 11. Total machine idle cost for illustrated example

Dendogram level	**Part families**	**Total machine idle cost**
1	[A] [B] [C] [D]	0
2	[A] [BC] [D]	65
3	[AD] [BC]	183
4	[ADBC]	669

Step 3: Calculate the total cost of each level by summing up the two costs as calculated in the previous step. Table 12 gives the total cost for all dendogram levels of the example problem.
Step 4: Finally, select the dendogram level that offers the minimum total cost. Corresponding to the selected dendogram level, a manufacturing plan is developed for the next reconfiguration/ manufacturing cycle. For the example being illustrated, it is clear from table 13, the set of part families corresponding to dendogram level '1' is to be selected. The final manufacturing plan is as shown in table 13 for the example problem

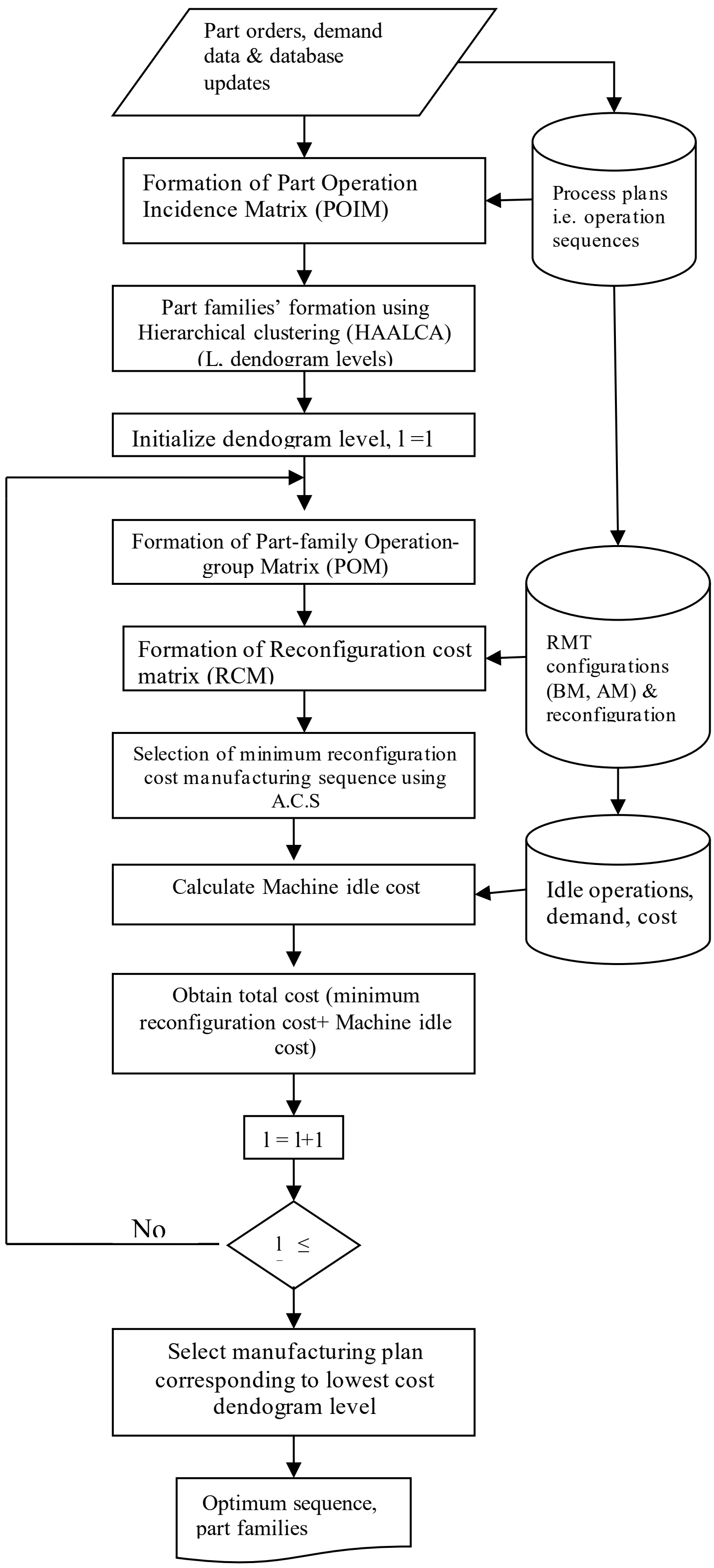


Figure 5. Proposed methodology for selection & sequencing of part families

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

TABLE 12. Total cost for illustrated example

Dendogram level	Least reconfiguration cost	Total machine idle cost	Total cost
1	220	0	220
2	204	65	269
3	165	183	348
4	0	669	669

TABLE 13. Manufacturing plan for reconfiguration cycle

Item	Details	remarks
Family selected	[A][B][C][D]	Corresponding to dendogram level '1'
Manufacturing sequence	A-D-B-C	Final sequence

Step 5: The planned reconfiguration sequence is recommended for implementation and it can be repeated as long as there are no significant changes in the next reconfiguration cycle

II. CASE STUDY

To demonstrate the successful implementation of the proposed methodology, two case studies are taken from literature, after appropriate modifications. These are presented as given below.

2.1 Case Study - I

2.1.1 The Problem Description

In this case study, the input data is the one used by R. Galan et al. (2005) [11] and is illustrated in table 14. It consists of 4 parts and 12 operations. The relevant information for various parts, their demands for the next reconfiguration cycle and their respective operation sequences has been given. RMT database, reconfiguration tasks and their costs and data related to machine idle cost are given in table 16.

TABLE 14. Basic Data

Part	Process plan (Operation sequences required)	Demand
A	1-2-3-4-5-12	20
B	2-6-7-5-9	50
C	5-6-7-12-9	100
D	5-2-8-10-11	40

TABLE 15. Operations and corresponding RMT configurations

Operations	BM	AM
1	4	7
2	2	12,9
3	1	1
4	3	3
5	1	11,1
6	5	1,6
7	4	7,5
8	2	5,4
9	4	3,2,8
10	3	9,4
11	3	2,3
12	5	10, 2

TABLE 16. Reconfiguration tasks & their costs

Reconfiguration task	Notation	Cost/task
RMT is required without any reconfiguration	a	0
RMT is required with some of its modules retained	b	1
Basic Module to be retained in configuring new RMT	c	6
Basic Module not required(to be removed)	d	5
New Basic Module is required	e	8
Auxiliary Module to be reutilized in configuring new RMT	f	3
Auxiliary Module not required(to be removed)	g	2
New Auxiliary Module is required	h	4

TABLE 17. Operations/RMTs and their idle cost

Operation/RMT	Unit machine Idle cost
1	0.1
2	0.2
3	0.4
4	0.8
5	0.7
6	0.6
7	0.3
8	0.5
9	0.2
10	0.4
11	0.6
12	0.9

2.2 The Solution Steps

Table 18 shows the generated POIM using input data from table 19. The dendogram information (part families formed) generated after implementation of HAALCA has been shown in table 20.

TABLE 18. Part Operation Incidence Matrix (POIM)

Part\operation	1	2	3	5	5	6	7	8	9	10	11	12
A	1	1	1	1	1	0	0	0	0	0	0	1
B	0	1	0	0	1	1	1	0	1	0	0	0
C	0	0	0	0	1	1	1	0	1	0	0	1
D	0	1	0	0	1	0	0	1	0	1	1	0

TABLE 19. Dendogram information

Dendogram level	Precision level (% similarity)	Part families formed
1	100	[A] [B] [C] [D]
2	67	[A] [D] [BC]
3	22	[AD] [BC]
4	20	[ADBC]

Table 20 gives the details of reconfiguration cost matrix (RCM) generated using the proposed methodology. This is a comprehensive RCM which considers all the families at all dendogram levels. It is utilized in parts corresponding to each dendogram level while calculating least cost reconfiguration sequences for each.

TABLE 20. Reconfiguration cost matrix (RCM) among families

	A	B	C	D	AD	BC
A	0	48	58	65	-	49
B	47	0	25	68	-	-
C	57	25	0	93	-	-
D	66	70	95	0	-	86
AD	-	-	-	-	0	70
BC	41	-	-	77	83	0

By applying ACS using RCM data, least cost reconfiguration sequence is obtained for each dendogram level as shown in table 19. Total machine idle cost calculated for least cost reconfiguration sequence is shown table 20. Table 21 gives the total cost for all dendogram levels.

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025.
ISSN (Online): 2456-7361

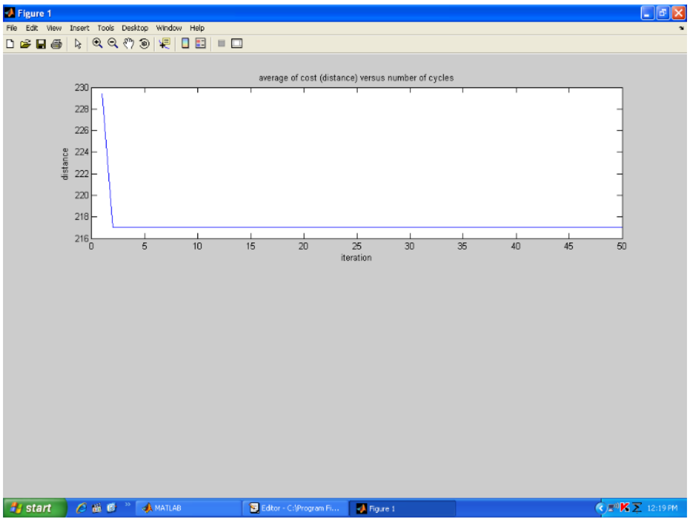


Figure 6. Optimum reconfiguration cost for dendogram level1

TABLE 21. Total machine idle cost

Dendogram level	Part families	Total machine idle cost
1	[A] [B] [C] [D]	0
2	[A] [BC] [D]	65
3	[AD] [BC]	183
4	[ADBC]	669

TABLE 22. Total cost for Case Study I

Dendogram level	Least reconfiguration cost	Total machine idle cost	Total cost
1	217	0	217
2	192	65	257
3	153	183	336
4	0	669	669

III. RESULTS

It is clear from table 22 that the final solution corresponds to dendogram level 1. Final manufacturing cycle plan generated has been shown in table 23. The planned reconfiguration sequence (i.e. A-C-B-D) is recommended for implementation and it can be repeated as long as there are no significant changes in the next reconfiguration cycle.

TABLE 23. Manufacturing plan for reconfiguration cycle

Item	Details	remarks
Family selected	[A][B][C][D]	Corresponding to dendogram level '1'
Manufacturing sequence	A-C-B-D	Final sequence

3.1 Case Study – II

3.1.1 The Problem Description

In this case study, the input data required is illustrated in table 24. It consists of 6 parts and 7 operations. The relevant information for various parts, their demands for the next reconfiguration cycle and their respective operation sequences has been given. RMT database, data related to machine idle cost are given in table 24 & table 25. Reconfiguration tasks and their costs are similar in both studies.

TABLE 24. Basic Data

Parts	Process plans	Demand
A	1-2-3-5-7	10
B	1-3-4-7-2-6	20
C	1-5-4	40
D	4-7-5	30
E	2-5-4	60
F	3-4-5-7	50

TABLE 25. Operations and corresponding RMT configurations

Operations	BM	AM
1	1	1, 2
2	2	7, 5, 3
3	3	6, 2
4	2	7, 5
5	1	3, 5
6	4	1, 6
7	3	4, 1

TABLE 26. Operations/RMTs and their idle cost

Operation/RMT	Unit machine Idle cost
1	0.2
2	0.5
3	0.4
4	0.7
5	0.3
6	0.9
7	0.6

3.1.2 The Solution Steps

Table 25 shows the generated POIM using input data from table 26. The dendogram information (part families formed) generated after implementation of HAALCA has been shown in table 27.

TABLE 27. Part Operation Incidence Matrix (POIM)

Part \operation	1	2	3	4	5	6	7
A	1	1	1	0	1	0	1
B	1	1	1	1	0	1	1
C	1	0	0	1	1	0	0
D	0	0	0	1	1	0	1
E	0	1	0	1	1	0	0
F	0	0	1	1	1	0	1

TABLE 28. Dendogram information

Dendogram level	Precision level (% similarity)	Part families formed
1	100	[A] [B] [C] [D] [E] [F]
2	75	[A] [B] [C] [DE] [F]
3	57	[AB] [C][DE] [F]
4	50	[AB] [C] [DEF]
5	43	[AB] [CDEF]
6	39	[ABCDEF]

TABLE 29. Reconfiguration cost matrix (RCM) among families

	A	B	C	D	E	F	AB	DE	DEF	CDEF
A	0	38	31	34	43	12	-	40	35	65
B	31	9	48	38	47	32	-	52	63	81
C	33	71	0	42	29	35	39	43	56	-
D	41	54	33	0	29	27	67	-	-	-
E	57	70	27	27	0	43	47	-	-	-
F	21	48	28	29	38	0	61	73	-	-
AB	-	-	46	52	71	69	0	61	43	80
DE	61	47	89	-	-	80	53	0	-	-
DEF	74	63	58	-	-	-	67	-	0	-
CDEF	49	91	-	-	-	-	75	-	-	0

Table 29 gives the details of reconfiguration cost matrix (RCM) generated using the proposed methodology. This is a comprehensive RCM which considers all the families at all dendogram levels. It is utilized in parts corresponding to each

International Journal of Scientific Engineering and Science
Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

dendogram level while calculating least cost reconfiguration sequences for each.

By applying ACS to RCM, least cost reconfiguration sequence obtained is shown in table 30. Total machine idle cost calculated for least cost reconfiguration sequence is shown table 31. Table 32 gives the total cost for all dendogram levels.

TABLE 30. Least cost reconfiguration sequence

Dendogram level (% Similarity)	Part families	Least cost reconfiguration sequence	Least reconfiguration cost
1 (100)	[A] [B] [C] [D] [E] [F]	A-B-C-D-E-F	251
2 (75)	[A] [B] [C] [DE] [F]	A-B-C-DE-F	230
3 (57)	[AB] [C] [DE][F]	AB-C-DE-F	226
4 (50)	[AB] (C) [DEF]	AB-C-DEF	169
5 (43)	[AB] [CDEF]	AB-CDEF	155
6 (39)	[ABCDEF]	ABCDEF	0

TABLE 31. Total machine idle cost

Dendogram level	Part families	Total machine idle cost
1	[A][B[[C] [D] [E] [F]	0
2	[A][B] [C] [DE] [F]	51
3	[AB] [C] [DE][F]	73
4	[AB] [C] [DEF]	134
5	[AB] [CDEF]	222
6	[ABCDEF]	384

TABLE 32. Total cost for Case Study II

Dendogram level	Least reconfiguration cost	Total machine idle cost	Total cost
1	251	0	251
2	230	51	281
3	226	73	299
4	169	134	303
5	155	222	377
6	0	384	384

3.1.3 Results

It is clear from table 32 that the final solution corresponds to dendogram level 1. Final manufacturing cycle plan generated has been shown in table 33. The planned reconfiguration sequence (i.e. A-B-C-D-E-F) is recommended for implementation and it can be repeated as long as there are no significant changes in the next reconfiguration cycle.

TABLE 33. Manufacturing plan for reconfiguration cycle

Item	Details	remarks
Family selected	[A][B[[C] [D] [E] [F]	Corresponding to dendogram level '1'
Manufacturing sequence	A-B-C-D-E-F	Final sequence

IV. Conclusions

In this work, a methodology has been developed to minimize the manufacturing costs in RMS through optimized selection and sequencing of part families. By illustration through case studies, the successful implementation, characteristics and advantages of the proposed RMS manufacturing/reconfiguration cycle planning strategy, have been brought to the light. Few of the most important observations are as follows:

1. Through a literature survey of the existing CMS cell formation methodologies, it is concluded that adopting "Hierarchical Agglomerative Average Linkage Clustering Algorithm" (HAALCA) is the most appropriate for RMS model considered fulfilling the requirements of its characteristic features.
2. In this work, the problem of finding the least cost reconfiguration sequence has been modeled as a TSP. It is a well established fact that evolutionary algorithms based on imitation of behaviour of ant colonies are found to give accurate results for solving TSP successfully, economically and efficiently. One such algorithm called 'Ant Colony System' has been adapted in this work to solve the problem of finding the least cost sequence of part families
3. The manufacturing plan generated can be used again and again until there are no significant change in part orders for a manufacturing/reconfiguration cycle. Whenever, there are changes in part orders, change of technology, introduction of new parts etc., we can use the methodology again to generate a new plan.
4. The methodology developed is straightforward and is easily implement-able into a economical computer program.

References

1. Abdi M.R., Labib A.W. (2004), "Grouping and selecting products: the design key of reconfigurable manufacturing systems (RMSs)", *International Journal of Production Research*, 42(3), 521-546.
2. Ayman M. A. Youssef Æ Hoda A. ElMaraghy, (2007), "Optimal configuration selection for Reconfigurable Manufacturing Systems", *Int J Flex Manuf Syst*, 19:67–106.
3. Chung-Hsien Kuo (2001), "Resource allocation and performance evaluation of the reconfigurable manufacturing systems", *IEEE International Conference on Systems, Man, and Cybernetics*, Volume: 4, 2451-2456.
4. Dorigo, M.; Maniezzo, V.; Colorni, A. (1996), "The Ant system: optimization by a colony of cooperating agents", Volume: 26 Issue: 1 Date: Feb 1996, *IEEE Transactions on Systems, Man & Cybernatics*, Part B, Page(s): 29-41
5. Dorigo M. and Gambardella L.M., (1997), "Ant colony system: A cooperative learning approach to the traveling salesman problem". *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, April, 53–66.
6. Dorigo M., Di Caro G., and Gambardella L.M., (1999), "Ant Algorithms for Discrete Optimization" *Artificial life*, Vol. 5, No. 3, 137-172.
7. Dorigo, M. and Sutzle T., (2004), "Ant Colony Optimization", Massachusetts Instiute of technology press, Vol.1, USA, Prentice Hall of India, page no. 65-117
8. Dorigo, M., and Stutzle, T., (2002), "The ant colony optimization metaheuristics: Algorithms", applications and advances. In F.Glover and G. Kochenberger (Eds.), Handbook of Metaheuristics, vol 57 of International Series in Operation Research and Management Science, 251-185. Norwell, MA, Kluwer Academic Publishers.
9. Galan R, Racero J, Eguia I, Setchi R. (2005) "Product grouping in reconfigurable manufacturing systems". *Proceedings of the 15th International conference on flexible automation and intelligent manufacturing*. p. 833–41.
10. Galan R., Racero J., Eguia I. and Garcia J.M. (2007), "A systematic approach for product families formation in Reconfigurable Manufacturing Systems", *Robotics and Computer-Integrated Manufacturing*, 23(5), 489-502.
11. Galan R., Racero J., Eguia I., Canca D. (2007), "A methodology for facilitating reconfiguration in manufacturing: the move towards reconfigurable manufacturing systems", *International Journal of Advanced Manufacturing Technology*, 33, 3&4, 345-353

Volume 9, Issue 11, pp. 105-115, 2025. ISSN (Online): 2456-7361

12. Gambardella L. M., Dorigo M. (1996) “Solving Symmetric and Asymmetric TSPs by Ant Colonies”, 1-6, *IEEE Conference on Evolutionary Computation (ICEC'96)* Japan
13. Gerhard Reinelt, (1994), “The Traveling Salesman Computational Solutions for TSP Applications”, Springer-Verlag Berlin Heidelberg 1994 edition.
14. Gershenson JK, Prasad GJ, Zhang Y (2003), “Product modularity: definitions and benefits”. *J Eng Des* 14:295–313
15. Groover M.P. and Zimmers E.W. (1984), “CAD/CAM: Computer Aided Design and Manufacturing”, Prentice Hall of India, 55-69.
16. Groover M.P. (2001), “Automation, Production Systems, and Computer Integrated Manufacturing”, Pearson Education Asia, 1st Indian edition, 71-86.
17. Gupta S., Krishnan V., (1998), “Product family-based assembly sequence design methodology”, *IIE Trans* 30:933–945.
18. Gupta T. (1991), “Clustering algorithms for the design of a cellular manufacturing system – an analysis of their performance”, *Comput Ind Eng* 20:461–468.
19. Hegui Ye, Ming Liang, (2005), “Solving the Combined Modular Product Scheduling and Production Cell Reconfiguration Problem- a GA Approach with Parallel Chromosome Coding”, 1348-1355.
20. Hegui Ye, Ming Liang, (2006), “Simultaneous Modular Product Scheduling and Manufacturing Cell Reconfiguration Using a Genetic Algorithm”, *Journal of Manufacturing Science & Engineering*, 128, 11, 984-995.
21. Hoda A. Elmaraghy (2006), “Flexible and reconfigurable manufacturing systems paradigms”, *Int J Flex Manuf Syst*, 17:261–276.
22. Jain P.K., Rakesh K., Mehta N.K., (2008), “An approach for simultaneous part families recognition and operation sequence selection for Reconfigurable Manufacturing systems”, *ISPE 24th International Conference on CARS and FOF*, 29-31July 2008, Nihon University, Koriyama, Japan.
23. Jain P.K., Mehta N.K., Rakesh K., Kumar M., “A systematic approach for selection and sequencing of part families in Reconfigurable manufacturing Systems”, *2nd International & 23rd All India Manufacturing Technology Design and Research Conference (AIMTDR 2008)*, Dec. 15-17, 2008 (paper communicated).
24. Jiao J, Ma Q, Tseng MM (2003), “Towards high value-added products and services: mass customization and beyond”, *Technovation* 23:809−821.
25. Koren Y, Heisel U, Jovane F, Moriwaki T, Pritschow G, Ulsoy G, Van Brussel H, (1999), “Reconfigurable manufacturing systems-keynote papers”, Ann CIRP 48:1–14