Manufacturing cells: the methods to form them and their limits

TESI DI LAUREA MAGISTRALE IN
XXXXX ENGINEERING-INGEGNERIA XXXXX

Author: **Edoardo Chiapponi**

Student ID:	920509
Advisor:	Antonio Calabrese
Co-advisor:	
Academic Year:	2021-22

Abstract

Manufacturing cell layout is one of the main layout configurations and it is formed by manufacturing cells, groups of machines which are dedicated on manufacturing part families. The problem of forming manufacturing cells is called Cell Formation (CF) problem and it can be solved by means of different methods. The scope of this thesis is in presenting several methods from 2000 to 2020 under the theoretical point of view, with the help of some examples taken by papers coming from the literature, comparing them and indicating their limits in terms of completeness and reliability. In fact, most of these methods are based on mathematical models, which are characterized by constraints and simplifications, moving them away from the real and more complex problem. What will arise from this analysis is a diffused lack of completeness among the methodologies, but in recent years it is evident a positive trend toward the integration of more aspects.

Key-words: Manufacturing cell, cell formation problem, cell formation, cell layout, cell scheduling.

Abstract in lingua italiana

Il layout delle celle di produzione è una delle principali configurazioni di layout ed è formato da celle di produzione, gruppi di macchine dedicate alla produzione di famiglie di parti. Il problema sulla formulazione delle celle di produzione è chiamato problema della formazione delle cellule (CF problem) e può essere risolto con diversi metodi. Lo scopo di questa tesi è presentare diversi metodi dal 2000 al 2020 dal punto di vista teorico, con l'ausilio di alcuni esempi tratti da articoli provenienti dalla letteratura, confrontandoli e indicandone i limiti in termini di completezza e affidabilità. Infatti la maggior parte di questi metodi si basa su modelli matematici, che sono caratterizzati da vincoli e semplificazioni, allontanandoli dal più complesso problema reale. Ciò che emergerà da questa analisi è una diffusa mancanza di completezza tra le metodologie, ma negli ultimi anni è evidente una tendenza positiva verso l'integrazione di più aspetti.

Parole chiave: Cella di produzione, problema della formazione delle cellule, formazione della cellula, layout della cellula, pianificazione della cellula.

Contents

1. Introduction

1.1 Introductory concepts

Group technology (GT) is a manufacturing technique based on the principle of grouping parts into families according to similarities in design or manufacturing process.

Cellular manufacturing (CM) is an application of GT in manufacturing and consists in machining one or more part families in a machine cell, which is formed by a group of machines or workstations. Cellular manufacturing combines the advantages of both the job shop and mass production approaches: in fact, it has the flexibility of the job shop and the efficient flow and high production rate of the mass production; giving it a good compromise between the two procedures.

Especially with respect to the job shop [1]: manufacturing cell permits a lower setup between batches, due to the fact the machines inside a cell have to work similar products, allowing smaller batch size and thus improve flexibility, giving the ability to produce the right amount of parts in the right time span; physically grouping together different machines reduces the transportation time between processes, the part flows are more linear and rational and there are no intersections, which are frequent in a job shop; the machine cell enables the creation of multiskilled workers that are able to work on different machines or workstations, compared to the job shop where each worker is specialized in one type of machine (milling, turning, drilling etc.).

The use of cellular manufacturing improves the flow time and tardiness performance of parts they process with respect to job shops [2,3], thus it is suitable in machining important part families. Other examples can be found in the millwork industry [4,5], where the transition from job shop to manufacturing cell is accompanied by a reduction on the order of 50% of lead time and scrap rate.

The process of determining part families and machine groups is referred to as the Cell Formation (CF) problem. As pointed out by many papers [6–9] given a set of input data like part types, processing requirements, part type demand and available resources (machines, equipment, etc.), the CF problem can be divided in three main steps:

1. Cell formation, machines are grouped together to form cells and part families are assigned to cells;
2. Cell layout, machines are rearranged inside the cell (intercell layout) and cells are repositioned inside the department (intracell layout);
3. Cell scheduling, the production sequence of part families is formulated and jobs are scheduled in order to better exploit the available time of production.

These steps will be better explained afterwards, in Chapter 4. Note that, based on the procedure that is adopted, it can happens that not all the abovementioned steps will be performed.

To make the CF problem clear, let us take as an example the following manufacturing department:

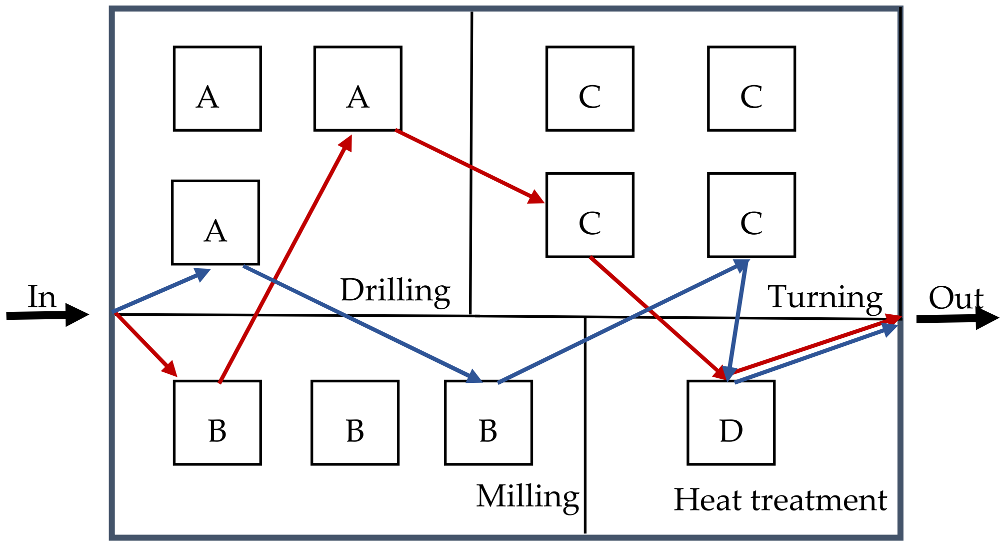


Figure 1.1: Example of a Job Shop department.

In Figure 1.1 it is possible to see a department divided in four shops (Drilling, Turning, Milling, Heat treatment) based on the process involved. Each shop contains machines of the same type: the Drilling shop has three drilling machines called A; the Milling one contains three multi axial CNC machines named B; the Turning shop accommodates four lathe machines labelled C; the Heat treatment shop contains one furnace named D. By means of red and blue arrows the flows of two possible part types are represented: the first part type follows the process path A-B-C-D; the second one the technological route B-A-C-D; it's easy to notice that in this job shop intersections and jams between the part flows are frequent, causing a longer flow time for the products from the input to the output of the department, and an higher material handling cost due to the total distance traveled by parts.

One possible solution to reduce flow time and costs is to rearrange the department in several machine cells, one for each part family. Once machine cells are formed in the cell formation step, cell layout step takes place, in which the designer must rearrange machines inside cells and the cells inside the department. The most used intercell layouts [10,11] are: linear single-row layout, where machines are positioned along a straight line; linear double-row layout, which is similar to the single-row layout, but has two lines of machines in parallel; u-shaped layout, it is similar to the single-row layout, in which machines are located along a circumference of an ellipse, very suited to enhance teamwork and communication among team members. These types of layouts can be seen in Figure 1.3, at the end of the paragraph. The solution will be characterized by more linear paths, with minimum intercellular flows between cells. However, the new department's arrangement could be formed not only by machine cells, but also by a job shop (called remainder cell) that includes the remaining machines that were not assigned to any cell; in the literature this solution is called hybrid layout [2,3,11–14]. The remainder cell has the duty to manufacture the remaining parts that do not belong to any part family and can also contain machines that cannot be duplicated due to their weight/cost (like furnaces, big presses, painting shop).

In Figure 1.2 is represented a feasible solution for the CF problem stated before. In the presented solution one part family is transformed by the machine cell called Cell 1, composed by two machines A, one B and one C; the other part family is machined inside the manufacturing cell labelled Cell 2, formed by two machines B, one A and two C; the remainder cell is simply composed by the remaining machines that are not request by the two cells (i.e. the capacity required by the demand of the two part families is already reached), but still they will be useful in case of demand variations of the two part families, giving an extra capacity to cope with increasing demand, or in case of introduction of new parts in the manufacturing program. In the remainder cell is also present the furnace D, which cannot be moved from its initial location and cannot be duplicated to have one furnace on each cell due to its cost and occupied surface area. From this configuration, cell scheduling can be performed to improve lead time of part families.

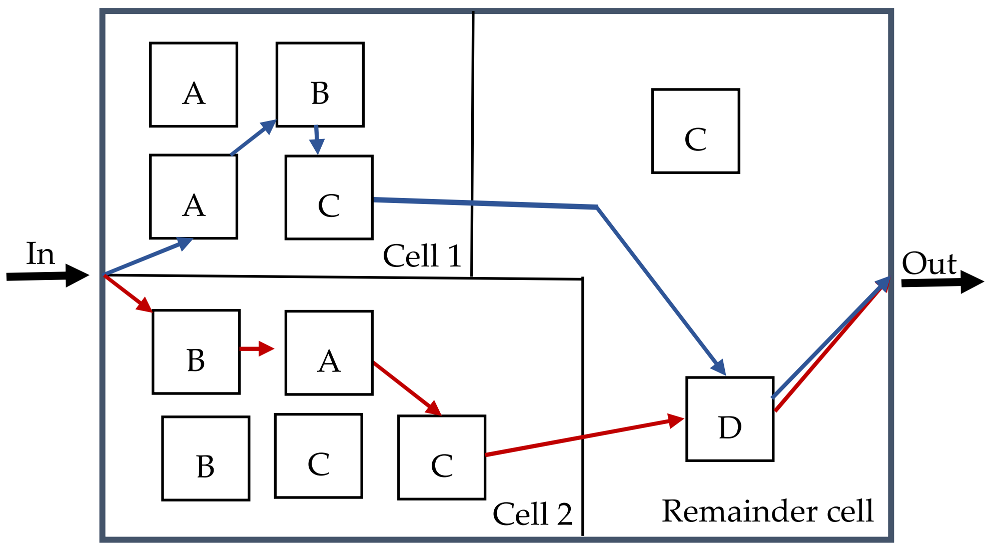
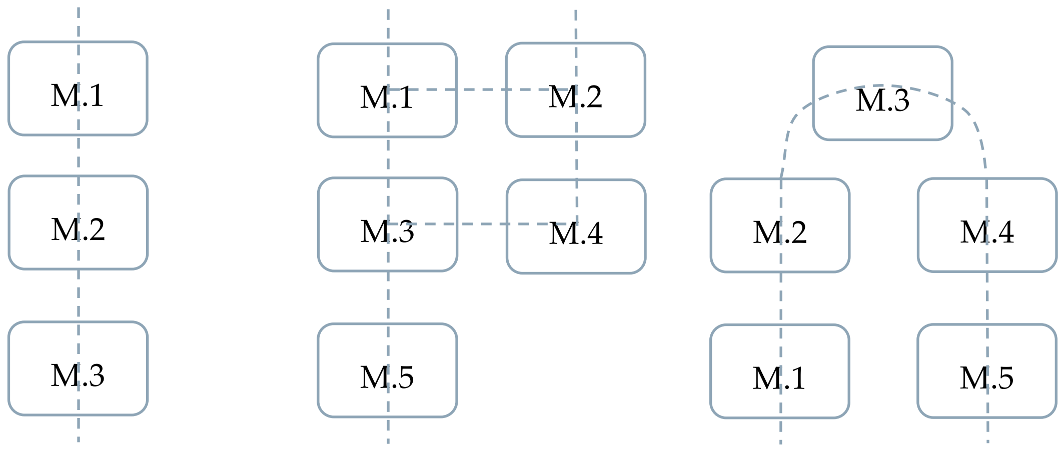


Figure 1.2: Possible solution of the CF problem.



(a) Linear single-row layout (b) Linear double-row layout (c) U-shaped layout

Figure 1.3: Common manufacturing cell layouts.

1.2 Scope of thesis

The scope of the thesis is in trying to:

1. Classify several CF methods, introducing them by means of theory and example sources.
2. Compare them in terms of different criteria like completeness of the solution, complexity of the algorithm, speed (computational time).
3. Discuss the limits of these methods in the real practice.

What is not included in the discussion:

- Supply chain, meaning that flows of parts entering the department and exiting the department are not considered, but only the flows within the single department;
- Material handling system, the procedure of choosing and purchasing material handling systems is not argued;
- Workspace layout, the specific space assigned to workers in not discussed.

Thus, this thesis aims at helping the designer to choose the best method available among all the possible CF techniques to properly design the layout of the single department. The reader will learn the basic theory of several algorithms, their advantages and disadvantages, and their applicability on the real world. About this last point, the main doubt that will rise from the entire explanation is on the reliability of these methods and their solutions, due to the fact they are based on Mathematical Models made by simplifications that, in most of the cases, don't consider data's uncertainties.

The originality of this thesis is in its broad view about this topic, that not only introduces different CF methods, but also tries to compare them and prove their applicability on the field.

1.3 Structure of the thesis

The thesis is divided is several chapters, with the aim of introducing step by step all the information that will be useful in the final discussion.

Chapter 1 constitutes the introduction of the thesis, made by introductory concepts about what it a CF problem.

Chapter 2 is about the research methodology followed during the preparation of the thesis, also presenting the sources employed.

Chapter 3 is the core of the thesis, devoted at classifying the CF methods and explaining them by means of basic theory and examples.

Chapter 4 is characterized by the comparison between CF methods to infer what is the overall best method in terms of quality of the given solution and speed.

Chapter 5 discusses the limits of CF methods in terms of applicability in real cases.

Chapter 6 concludes the thesis with final remarks.

2. Literature review

To explain the topic of CF problem, only the most recent sources are used, in order to be the most updated possible. That is why all the sources come from the range of years 2000-2021. In fact, as it will be possible to appreciate, the complexity and completeness of the methods grows thanks to the continuous evolution of calculators in terms of hardware and software, and also due to the market that requests dynamic and rapid manufacturing systems.

The most used databases websites to gather sources are *Web of Science* (*www.webofscience.com*), the electronic archives of Politecnico di Milano (*biblio.polimi.it*), and *Wikipedia*. The keywords employed to find appropriate sources are "*cell manufacturing*", "*cell formation*", "*cell layout*", "*cell scheduling*", "*review*", all the discussed methods like "*mathematical programming*", "*metaheuristics*", "*Tabu search*", etc.

Sources are classified in few macro categories:

- Books: books are useful to study an argument by the theoretical point of view, in fact they will be employed in the central part of the discussion (chapter 3), where several methods will be presented.
- Review papers: as for books, they can be used to introduce an argument in a general view, but they also provide important statements about the studied literature, therefore their contribution will be important in the concluding part of the thesis (chapter 4).
- Study papers: study papers do research studies in specific topics, comparing for example different layout approaches [2–4], or introducing models to guide designers and managers to cope with worker assignment and worker issues [15–19]. Some of them also do literature review like it happens for review papers.
- Method papers: in this category fall all the papers that explain one or more novel algorithms to solve CF problem. Some of them will be employed as examples to better clarify the CF methods in the core of the discussion

(chapter 3) and, afterwards, together with other research papers, to analyze what still needs to be done (chapter 4).

- Notebook notes/slides: information coming from courses attended at university.

In Figure 2.1, the 104 sources employed in the thesis are arranged based on their year of publication, sources employed for an image only or single images are not included:

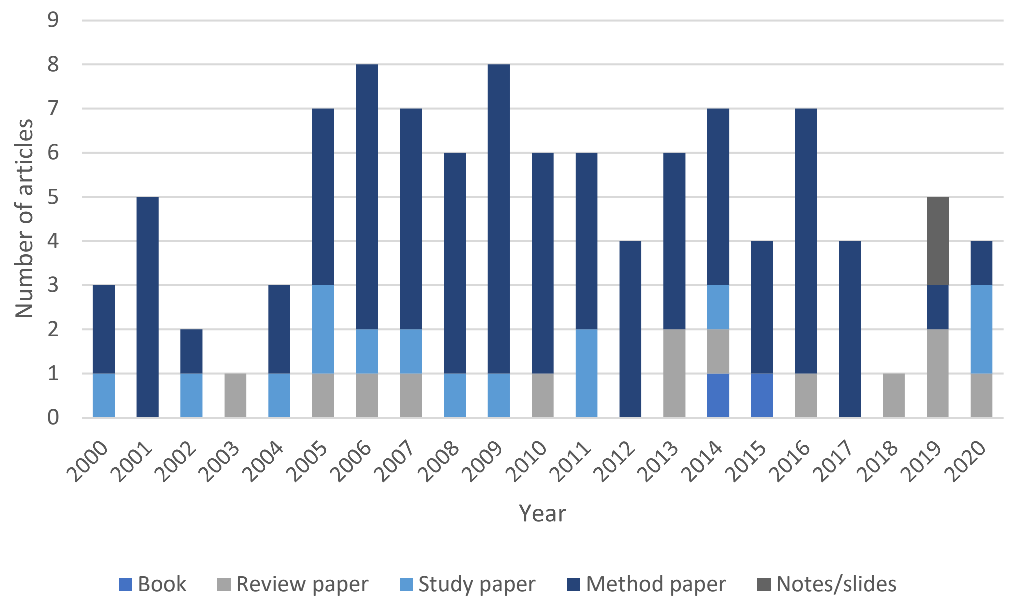


Figure 2.1: The distribution of the sources among the years.

The year of review papers and method papers is given by the software *Mendeley*, and it coincides with the year of publication in a journal. Books are labeled by the year of last published edition. Handbook notes and slides are marked in the year when the courses were attended. From the figure above it can be appreciated the continuous interest of researchers about the topic of cellular manufacturing and CF problem. There are no sources coming from 2021.

Method papers are the most diffused type of source, with 73 entries, followed by study papers and review papers, respectively with 14 and 13 entries, and lastly by books and notebook notes/slides with 2 entries each. Thus, it is of high interest classifying the method papers based on the model they rely on. As it will be explained in the next chapter (Chapter 3), the CF problem can be solved by means of several methods, each of them with their own characteristics. The distribution of proposed methods to solve the CF problem is depicted in Figure 2.2:

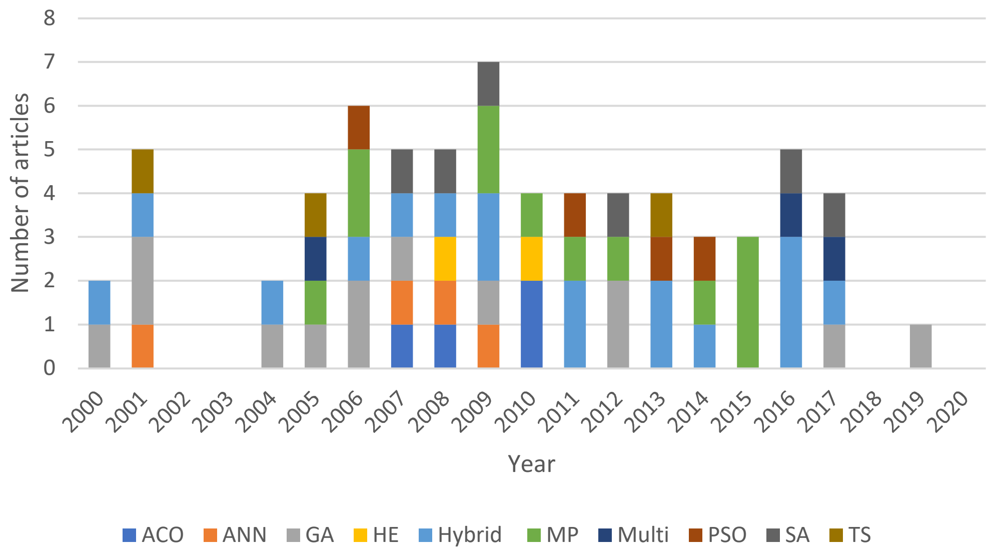


Figure 2.2: Distribution of proposed methods among the years.

Different methodologies are employed to solve the CF problem. Among all the possible models, the algorithms of major interest in the research field are the ones contained in the Production Based Methods. As the name suggests, these methods rely on production data coming from the manufacturing department like the operation sequence of part types, machines' capability (the ability of the machine to do a certain operation) and capacity, demand of part types etc., thus their chance to better meet the firm's request to design a proper MC can be higher with respect to other methods like Visual Methods and Part coding analysis (PCA) that rely on geometrical and technological aspects only. A better explanation of all these approaches will be the focus of Chapter 3.

In the diagram the methodologies of interest are labeled by means of their acronyms, that now will be quickly explained:

- MP stands for Mathematical Programming, this category contains all the papers that propose mathematical models only [11,14,20–29];
- HE means Heuristics, a type of methods that are based on heuristic/practical equations [30,31];
- TS and SA denote respectively Tabu Search and Simulated Annealing, two local search approaches included in the Metaheuristics, that can be seen as structured/defined heuristics. [10,32,33] are Tabu Search models, while [34–39] are Simulated Annealing models;
- GA, PSO and ACO respectively stand for Genetic Algorithm, Particle Swarm Optimization and Ant Colony Optimization, they are three global search

Metaheuristic approaches and, as their names suggest, are based on biological principles. Refer to [7,40–51] for Genetic Algorithm, to [52–55] for Particle Swarm Optimization, and to [56–59] for Ant Colony Optimization;

- ANN means Artificial Neural Network and, together with Fuzzy Logic (FL), it is part of the Artificial Intelligence methodologies, that are techniques trying to mimic human's thinking processing. ANN sources are the following [60–63];
- Multi denotes all the papers than propose more than one model for the CF problem.[64] proposes GA, SA, and TS methods, while [65] GA and SA approaches, [66] proposes GA and PSO methods, ;
- Hybrid stands for Hybrid algorithms, which are methods that couple together two different models in order to enhance their computational capabilities. There is a grand variety of hybrid methods, in fact [67,68] are TS-GA hybrid algorithms, [9,69–71] are SA-GA, [72–74] are GA-Local search (a sort of TS and SA), [75] is a GA-MP, [76] is a GA-ANN, [77] is a GA-FL, [78] is a GA-AUGMECON (Augmented e-constraint method), [79] is a SA-MP, [80] is a PSO-MP, [81] is a FL-MP, [82] is a FL-ANN.

The information contained in Figure 2.2 is rearranged to show the total number of papers per type of method sorted in a descending order as in Table 2.1:

Method type	Total number of papers	
Hybrid	17	
GA	13	
MP	12	
SA	6	
ACO	4	
ANN	4	
PSO	4	
Multi	3	
TS	3	Sum
HE	2	68

Table 2.1: Proposed methods sorted by type

As it is possible to see, most of the papers are Hybrid, GA, or MP methods but, keeping in mind that most of the hybrid approaches proposed contain GA, GA seems to be the most used method. This hypothesis is confirmed by the literature. For example, the review paper [83] observed that 43% of the papers from 1990 to 2013 proposing methods to solve the CF problem were using GA and, in accordance with a more recent paper [84], GA is the most used method in Hybrid approaches and it is very frequent coupled with TS, SA and Local search methods. Other techniques like PSO, ACO and ANN are less common in both papers and in this study. Percentages of found method papers per method type can be seen in Figure 2.3, which has the same data of Table 2.1:

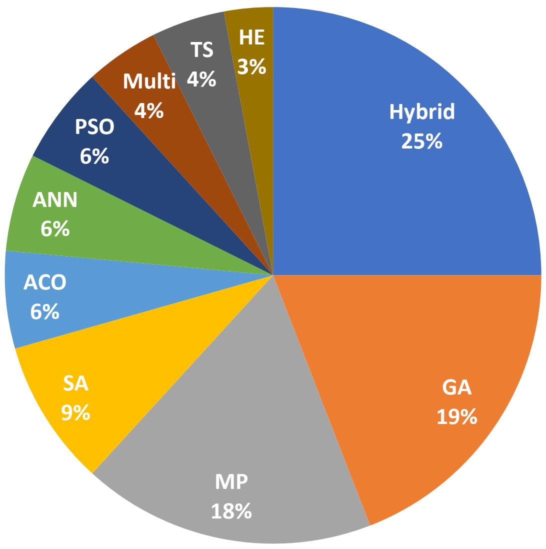


Figure 2.3: Pie chart of proposed methods by type

3. Methods for CF problem

From the concept of GT, introduced in the early 70s, a lot of work has been done in the field of CF problem, resulting in a spread of several methods that try to resolve this problem. Following the literature [6], a first classification of the several methodologies results in the three categories above:

- Visual methods.
- Part coding analysis methods.
- Production based or Production flow analysis methods.

3.1 Visual methods

Visual methods or simply ''eye-balling'' methods rely on the visual inspection of products and identification of the correspondent part families and machine cells. This methodology is effective and easy to apply only when the number of parts and machines is small; otherwise, the identification task becomes impossible.

Figure 3.1: Stainless steel turned parts.

Taking as an example Figure 3.1 [85], it is easy to notice that all the parts, even if they are different at a first sight, are made by turning operations and have similar dimensions and shapes, and they are even made of the same material. Thus, in this case are present both geometrical (dimension, shape) and process (turning) similarities; from this statement is reasonable to group the parts in one single family and physically arrange together the required machines in a machine cell.

3.2 Part coding analysis (PCA)

Part coding analysis (PCA) methods rely on a coding system. The part code results after assigning numerical weights/digits to part characteristics and features according to the defined coding system. The part codes so formulated will be useful to identify part families. With respect to visual methods, PCA methods are standardized (less subjective), reproducible, can be digitalized (to have an archive accessible through a PC), can be used to categorize a larger number of parts; however, the coding task is more time consuming and complex to pursue.

The most widely used codes are: Opitz, Brisch, MICLASS, CAMAC, CIMTEL [86].

As an example, the Opitz code is shown in Figure 3.2:

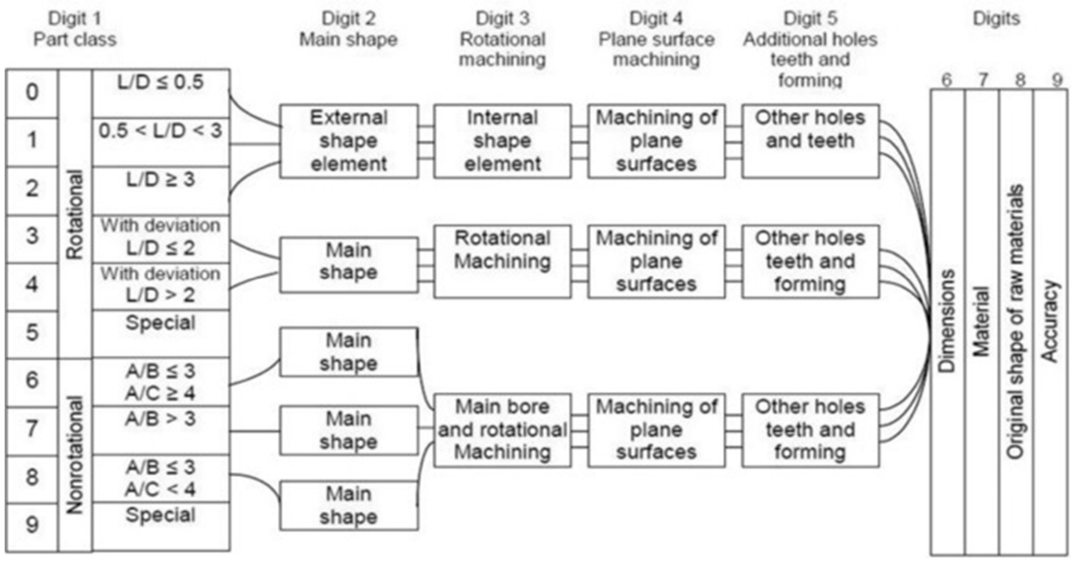


Figure 3.2: The Opitz code's main structure.

The Opitz code includes 9 digits representing a wide range of characteristics. Following this scheme, the user will be able to assign a code to each part. If CAD models of the products are available, this coding task can be done automatically by a

PC by means of a CAD reader [87]: the CAD reader reads the CAD model files, from which, by means of the feature recognition module, several features are identified in a form coherent with the chosen model signature (i.e. numerical code like the Opitz code); then the program can be used to compare different components employing a distance function between two Opitz codes, like the cosine coefficient S_{Cos}:

$$S_{Cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2}\sqrt{Q_i^2}} \tag{3.1}$$

Equation (3.1) computes the cosine of the angle between two vectors (P and Q). The nominator is the scalar product between the two vectors in question, while the denominator is the product of the norms of the two vectors. This function results in a real number between 0 (no similarity) and 1 (high similarity).

3.3 Production based methods

The core classification is production based or production flow methods, which can further be classified as follows [6,88]:

- Cluster analysis.
- Graph partitioning approaches.
- Mathematical programming methods.
- Heuristic and Metaheuristic algorithms.
- Artificial intelligence methodologies.

3.3.1 Cluster analysis

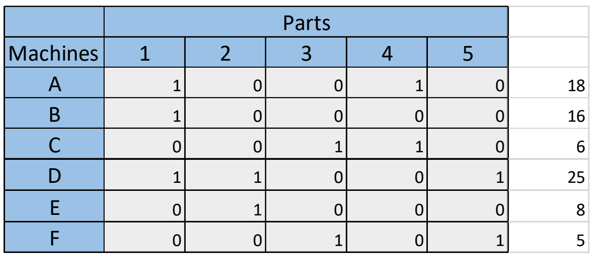
Mainly inspired by [86], the following considerations about Cluster analysis can be presented.

Cluster analysis is a cell formation tool able to form clusters such that elements within a cluster have a high degree of similarity and a very low degree of similarity with elements of different clusters. Clustering techniques can be classified as array-based clustering, hierarchical clustering, and non-hierarchical clustering techniques.

In array-based clustering the technological route of each part is represented by the part/machine matrix formulation. Each element of the matrix (a_{ij}) has zero or one entries. A '1' entry in row *i* and column *j* indicates that component *j* has an operation in machine *i*, instead a '0' entry denotes no operation of part *j* in machine *i*. The objective is to rearrange columns and rows to find clusters (machine cells) characterized by a certain part family and machine group. A well-known array-based clustering technique in the academic environment is the Rank Order Clustering (ROC) introduced by King, J. R. in the '80s. Given an m-by-n (machines-by-parts) matrix, the ROC algorithm has the following steps:

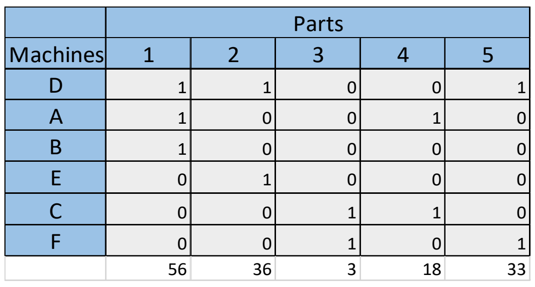
```
Algorithm 1 Rank Order Clustering (ROC)
1:    For each row $i$ compute the number $\sum_{j=1}^{m} a_{ij} * 2^{m-j}$
2:    Order rows according to descending numbers previously computed
3:    For each column $j$ compute the number $\sum_{i=1}^{n} a_{ij} * 2^{n-i}$
4:    Order rows according to descending numbers previously computed
5:    If on steps 2 and 4 no reordering happened go to step 6, otherwise go to step 1
6:    Stop
```

As an example, five parts and their relative routes inside the facility are considered:



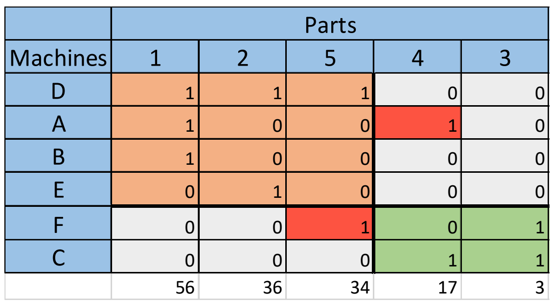
	Parts					
Machines	1	2	3	4	5	
A	1	0	0	1	0	18
B	1	0	0	0	0	16
C	0	0	1	1	0	6
D	1	1	0	0	1	25
E	0	1	0	0	0	8
F	0	0	1	0	1	5

(a) Step 1



	Parts				
Machines	1	2	3	4	5
D	1	1	0	0	1
A	1	0	0	1	0
B	1	0	0	0	0
E	0	1	0	0	0
C	0	0	1	1	0
F	0	0	1	0	1
	56	36	3	18	33

(b) Step 3



	Parts				
Machines	1	2	5	4	3
D	1	1	1	0	0
A	1	0	0	1	0
B	1	0	0	0	0
E	0	1	0	0	0
F	0	0	1	0	1
C	0	0	0	1	1
	56	36	34	17	3

(c) Final configuration

Figure 3.3: Rank Order Clustering example

Two possible manufacturing cells are identified: the orange one (cell 1) with machines A, B, D, E and part family 1, 2, 5, and the green one (cell 2) with machines C, F and part family 4, 3. The red rectangles highlight that for parts 4 and 5 a further choice is needed: duplicate machines F and A, that will respectively go to cell 1 and cell 2, bringing to an higher purchasing cost of the machines; or allow intercellular flows to satisfy the technological routes of parts 4 and 5, causing higher flow time.

Both hierarchical and non-hierarchical clustering use similarity or dissimilarity coefficients between the candidate objects for clustering, like machines in the CF problem; a list of these coefficients used in clustering can be found in [12]. After obtaining the similarity matrix containing few broad cells thanks to a clustering algorithm, by means of a graphical representation like a dendrogram is possible to sort every machine cell basing on its similarity coefficients and decide which manufacturing cells to form.

Starting from a parts/machines matrix like the one in Figure 3.3a, the steps are the following:

Algorithm 2 Clustering method with similarity coefficients

```
1: Compute the similarity coefficients $s_{ij} = \max\left(\frac{n_{ij}}{n_i}; \frac{n_{ij}}{n_j}\right)$
2: Join the couple of machines $i^*$, $j^*$with the highest similarity coefficient, thus forming
   the machine group k
3: Substitute the original rows and columns of machines $i^*$ and $j^*$ with rows and
   columns of machine group k, then compute the similarity coefficient
   $s_{rk} = \max\left(s_{ri^*}; s_{rj^*}\right)$
4: Go to step 2 if the solution complies with the chosen criterion (i.e., single machine
   group, predetermined number of machine groups), otherwise go to step 5
5: Stop
```

The similarity coefficient s_{ij} is the maximum value between the ratios $\frac{n_{ij}}{n_i}$ and $\frac{n_{ij}}{n_j}$, where: n_i, n_j represent the number of part types that require to visit respectively machines i and j ; n_{ij} means the number of part types that require to visit both machines i and j. Thus, the similarity coefficient is a value that goes from zero (no similarity) to one (high similarity). Employing a dendrogram like the one in Figure

3.4 [86] and defining the minimum similarity coefficients amongst the machine groups (like 0,75 in this example) is possible to form the final machine cells.

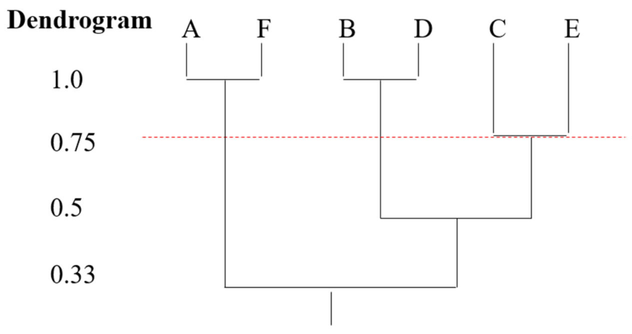


Figure 3.4: Example of a dendrogram

In this case three machine cells/groups are formed: A, F; B, D and C, E.

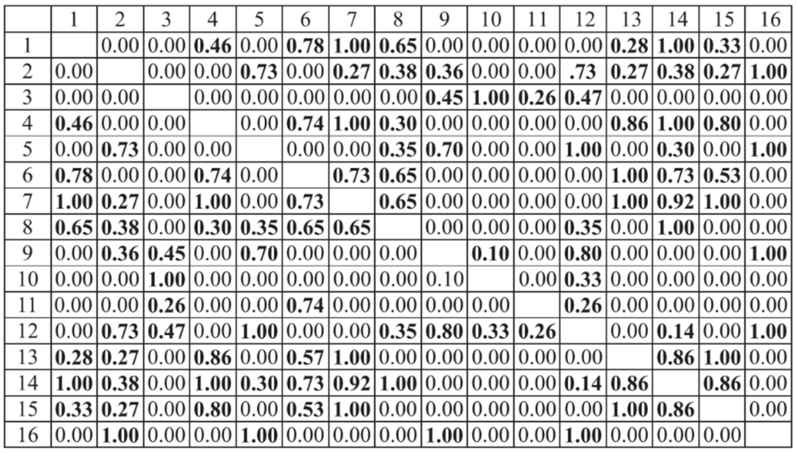
The method described above is known as Single Linkage Clustering Algorithm (SLCA) [46], introduced by Sneath in 1957. The main drawback of SLCA is that it generates the same dendrogram of solutions without considering the specific optimization objective. Thus, using the same similarity coefficient does not provide an optimal solution for the specific problem. That is why recently there are studies about hybrid methods that embed the known SLCA with Genetic Programming (GP), also called GP-SLCA [45,46]. The GP, or Genetic Algorithm (GA), is an evolutionary computation method that, as evolution does, slightly modifies the solutions until an optimal solution is found. In the case of GP-SLCA, the GP employs a randomly generated population of similarity coefficients that generates a respective population of solution dendrograms. By evolving these similarity coefficients by means of typical evolutionary computation mechanisms, where the genetic material is exchanged and only the best performing individual survives, an optimal solution for the problem considered is found. Genetic Algorithms will be better explained afterwards.

Although cluster methodologies are easy to implement and a solution can be found in a reasonable amount of time, they have a main drawback: as saw in the previous examples, the only data used as input is the parts/machines matrix, which gives us a simple but inaccurate idea about the possible route taken by each part type without telling us the operational sequence nor the operational times, therefore the retrieved solution may be valid only in limited/simple situations.

3.3.2 Graph partition approach

Graph partitioning approach is very similar to cluster analysis: also in this case, the objective is to rearrange the part/machine incidence matrix in few broad cells. Following the paper [89], in this case more input data are given: besides the part routing for each part type, are also given the available number of individual machines of each type, the machine processing capabilities (which operations a machine can do), and the part-processing usage for each machine (how much of the utilization rate is used to do a certain operation on a part type). From these data desirability measures are calculated. Desirability measure is a value between zero and one that indicates the level of relationship that is between two individual machines: the higher the value (the closer to one), the higher is the traffic of parts between the two machines, and so the more convenient would be to put the two machines in the same machine cell. Then the desirability measures are used to construct the machine-machine (mc-mc) graph, with desirability measures as arc weights and machine usage as node weights. Giving the mc-mc graph to a heuristic algorithm that has the duty to move machines basing on desirability measures and given number of machine cells set by the user, is finally possible to obtain the desired machine cells.

From the abovementioned paper this example is gathered: there are 16 machines (MC) of 9 different types (type) and 19 different part types. Collecting the data already explained both for machines and parts, a desirability measure matrix is obtained:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		0.00	0.00	**0.46**	0.00	**0.78**	**1.00**	**0.65**	0.00	0.00	0.00	0.00	**0.28**	**1.00**	**0.33**	0.00
2	0.00		0.00	0.00	**0.73**	0.00	**0.27**	**0.38**	**0.36**	0.00	0.00	**.73**	**0.27**	**0.38**	**0.27**	**1.00**
3	0.00	0.00		0.00	0.00	0.00	0.00	0.00	**0.45**	**1.00**	**0.26**	**0.47**	0.00	0.00	0.00	0.00
4	**0.46**	0.00	0.00		0.00	**0.74**	**1.00**	**0.30**	0.00	0.00	0.00	0.00	**0.86**	**1.00**	**0.80**	0.00
5	0.00	**0.73**	0.00	0.00		0.00	0.00	**0.35**	**0.70**	0.00	0.00	**1.00**	0.00	**0.30**	0.00	**1.00**
6	**0.78**	0.00	0.00	**0.74**	0.00		**0.73**	**0.65**	0.00	0.00	0.00	0.00	**1.00**	**0.73**	**0.53**	0.00
7	**1.00**	**0.27**	0.00	**1.00**	0.00	**0.73**		**0.65**	0.00	0.00	0.00	0.00	**1.00**	**0.92**	**1.00**	0.00
8	**0.65**	**0.38**	0.00	**0.30**	**0.35**	**0.65**	**0.65**		0.00	0.00	0.00	**0.35**	0.00	**1.00**	0.00	0.00
9	0.00	**0.36**	**0.45**	0.00	**0.70**	0.00	0.00	0.00		**0.10**	0.00	**0.80**	0.00	0.00	0.00	**1.00**
10	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.10		0.00	**0.33**	0.00	0.00	0.00	0.00
11	0.00	0.00	**0.26**	0.00	0.00	**0.74**	0.00	0.00	0.00	0.00		**0.26**	0.00	0.00	0.00	0.00
12	0.00	**0.73**	**0.47**	0.00	**1.00**	0.00	0.00	**0.35**	**0.80**	**0.33**	**0.26**		0.00	**0.14**	0.00	**1.00**
13	**0.28**	**0.27**	0.00	**0.86**	0.00	**0.57**	**1.00**	0.00	0.00	0.00	0.00	0.00		**0.86**	**1.00**	0.00
14	**1.00**	**0.38**	0.00	**1.00**	**0.30**	**0.73**	**0.92**	**1.00**	0.00	0.00	0.00	**0.14**	**0.86**		**0.86**	0.00
15	**0.33**	**0.27**	0.00	**0.80**	0.00	**0.53**	**1.00**	0.00	0.00	0.00	0.00	0.00	**1.00**	**0.86**		0.00
16	0.00	**1.00**	0.00	0.00	**1.00**	0.00	0.00	0.00	**1.00**	0.00	0.00	**1.00**	0.00	0.00	0.00	

Figure 3.5: Desirability measure matrix.

Then, composing the mc-mc graph from this matrix and giving it to the heuristic algorithm, the final part-machine incidence matrix is represented in Figure 3.6:

MC	1	4	6	7	8	11	13	14	15	2	3	5	9	10	12	16
Type	1	2	3	4	5	6	7	8	9	1	2	3	5	6	7	9
1	1			1				1	1							
2		1	1	1			1	1	1							
3	1	1		1			1	1	1							
4				1			1		1	$\boxed{1}$						
7	1		1	1				1								
8	1	1	1	1	1			1								
9	1		1	1	1			1								
10	1		1	1				1								
11			1			1										
5										1		1	1		1	1
6					$\boxed{1}$			$\boxed{1}$		1		1			1	
12											1			1	1	
13											1		1	1	1	
14											1		1		1	
15											1		1			
16											1			1		
17						$\boxed{1}$					1				1	
18												1	1		1	
19													1		1	

Figure 3.6: Final part-machine incidence matrix.

In this example two machine cells are formed. Notice that machines of the same type can stay in different cells. In this case there are exceptional parts, highlighted by the squares.

In conclusion, graph partitioning approach can manipulate more data with respect to cluster methodologies and give to the user several alternative cellular configurations by setting the number of cells and the desired cell size.

3.3.3 Mathematical programming

In mathematical programming methods the designer must write a mathematical model for the specific problem that could be a nonlinear or linear integer model. The procedure can be explained in this way: the problem of a hypothetical firm is how many machines of each type it has to buy to satisfy a certain (deterministic) demand, knowing the costs and the constraints. A simple mathematical model can be formulated in the following way [90]:

$$\begin{gathered} min\ \mathbf{c}^T \cdot \mathbf{x} \\ \text{s.t. } \mathbf{A} \cdot \mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{gathered} \tag{3.2}$$

Where:

- $\mathbf{x} \in \mathrm{R}^{\mathrm{n}}$ is the vector of decision variables
- $\mathbf{A} \in \mathrm{R}^{\mathrm{m} \times \mathrm{n}}$ is the matrix of technical coefficients

- $\mathbf{c} \in \mathrm{R}^{\mathrm{n}}$ is the vector of cost coefficients
- $\mathbf{b} \in \mathrm{R}^{\mathrm{m}}$ is the vector of right-hand side values

The main objective of this model is to minimize the costs $min\, \mathbf{c}^T \cdot \mathbf{x}$:for costs is possible to have purchasing cost to buy new machines, hiring cost to hire new workers, maintenance costs, material handling costs and many other costs; the more common decision variables are the number and types of machines to buy.

Below the objective there are the constraints (s.t. means subject to) that must be respected to stay in the space of feasible solutions to our problem. In the matrix of technical coefficients **A** there are for example the production rates of each machine (how many parts per hour a machine can produce) and the space occupied by each type of machine, the vector **b** can include the monthly demand for each part type and limitations like maximum area that can be occupied and maximum number of machines of each type. The last constraint $\mathbf{x} \geq 0$ allows only positive and integer number of machines.

In the following some examples of mathematical models.

[23] introduced an integer nonlinear programming model for the design of dynamic cellular manufacturing systems. In a dynamic environment, due to the variability of product mix and demand requirements on each period, what is optimal in a period may not be optimal and efficient for the next period. Thus, machines and workers must be flexible enough to satisfy each period. The objective function is to minimize the costs, which is the sum of different terms: machine maintenance and overhead cost, machine procurement cost, inter-cell travel cost, machine operation and setup cost, tool consumption cost, and system re-configuration cost. Among the 16 constraints written there are the ones checking the capacity of the machines and the number of machines allowed in each cell, the operation sequences and workload balancing among the cells. The model exploits several techniques to generate a dynamic CMS: dynamic reconfiguration of cells by moving machines between cells and/or buying new machines, alternative routings to have a more flexible system, lot splitting by dividing large order into smaller batches providing the opportunity for simultaneous processing of orders to more than one work center. The main limitation

to this model is the computational time, which consequently limits the applicability of the model. In fact, can be used for relatively small problems, the biggest one reported is 2 planning period, 3 cells, 10 machine types and 25 part types; the model can also accommodate 3 periods with 3 cells, 6 machine types and 15 part types. The authors suggested the use of a heuristic method to find an optimal solution for bigger problems.

[26] proposed a novel integer nonlinear programming model for dynamic CMS. This model is very similar to the one already presented in terms of objective function and constraints, but it has a further dimension: worker assignment. Thus, the time available for each worker, salary, hiring and firing are considered. In the model the objective is to minimize the costs, which is the sum of nine terms: holding cost, backorder cost, inter-cell material handling cost, machine cost, machine relocation cost, salary cost, hiring cost and firing cost. Each of these cost items are explained in the paper. In the constraints the demand of each period, capacity and number of machines and workers on each cell and the workload balancing among the cells are considered. The authors explain that the main limitation to the model is computational time to find the optimal solution (i.e., the biggest data set used is 3 machine types, 6 part types, 4 worker types and 3 periods and optimal solution found in 980 minutes) thus other techniques must be applied like metaheuristic approach. Further reasoning about this cubic space of machine–part–worker can be found in [28] by the same authors.

[21] formulated a mathematical model to solve a different kind of problem: starting from an already existing CMS, satisfy the demand avoiding or limiting the redesign of it. Considering exceptional elements within the CMS like bottleneck machines and exceptional parts, the one-period model has as objective function the minimization of the costs related with the exceptional elements. These costs include new equipment purchases by machine duplication, intercellular transfers, and the incremental costs of subcontracting. Minimizing these costs, the model has the duty to retain the current cellular formation. Constraints consider the capacity of machines and the number of machines in each cell; operation sequences and workload balancing

constraints are not included. Even in this case the model is used for a small problem including 9 machine types and 10 part types.

From all the examples above is evident that mathematical programming can be used only for small problems. Due to the NP-hard nature of the CF problem heuristic, metaheuristic and hybrid metaheuristic approaches have been successfully proposed producing acceptable solutions in reasonable time. In computational complexity theory, NP-hard means non-deterministic polynomial-time hard, a class of complex problems that cannot be solved by normal computers in polynomial time. Further, the decision making process in a manufacturing system often involves uncertainties and ambiguities. Under such circumstances, fuzzy methodologies have proved to be effective tools for taking fuzziness into consideration. Moreover, neural networks have been employed successfully for CF due to their robust nature [6].

3.3.4 Heuristic programming

Heuristic algorithms don't give an optimal solution, only sub-optimal solutions, but they are useful in providing an acceptable solution in a reasonable time. As the name suggests, Heuristics are less rigorous with respect to Mathematical Programming and their algorithms are based on practical equations.

[30] developed a heuristic algorithm that takes as input a machine-part incidence matrix containing operation sequence information. Starting from this matrix, by means of 15 steps (also represented by a flow chart), the algorithm has the duty to minimize the intercellular and intracellular movements. In fact, intracellular movements (movements inside a machine cell) must be regulated: from a production planning and control perspective, reverse movements within a cell and skipping of workstations are undesirable, as they tend to increase the complexity of planning and control. Costs are not calculated, but it is obvious that rationalizing part flows has a beneficial effect on both costs (i.e., material handling costs) and scheduling. The algorithm is applied to a problem composed by 25 machine types and 40 part types.

[31] starting from a mathematical model, that has as objective the minimization of intracell and intercell material flow costs, the authors formulated a heuristic algorithm composed by three stages:

1. form the temporary machine group plan according to the alternative process routings of each part: once the expected process routing of each part is found, an adjacency matrix to represent the material flows of all parts between two machines is constructed. This matrix, that has dimension M x M where M is the number of machines, is the input to a code that gives in output the temporary machine group plan G^{λ}.
2. select the appropriate process routing of each part with respect to the over-all material movement cost: starting from G^{λ}, the material flow cost H_a^r of each part a in its own process routing r is calculated, and the appropriate process routing chosen τ for each part a is the one that minimizes the material flow cost $H_a^{\tau} = \min_r H_a^r$.
3. configure the regular manufacturing cells based on the appropriate process routing: similarly to the first step, based on the appropriate process routings found in step 2, an adjacency matrix is formed and finally the final machine groups and part families are determined.

This three-steps heuristic method is applied to an industrial case formed by 41 different types of machines and 27 part types, forming 6 machine cells. This result is obtained in 326.447 s (about 5 minutes and a half). Thus, comparing it for example with [26], explained before, is evident the large difference between mathematical programming and heuristic algorithm: the second one is able to display a larger sub-optimal solution in a very minor time with respect to the first approach.

3.3.5 Metaheuristic programming

Metaheuristics are a group of several optimization techniques. They rely on the following basic principle: the search for optimum actually simulates either the behavior of a biologic system or the evolution of a natural phenomenon, including an intrinsic optimality mechanism [91].

In general, all metaheuristics are using a pseudo-random engine to select some parameters or operations that yield estimation of optimal solution. Thus, the method used to generate pseudo-random numbers is crucial in metaheuristics.

Despite the large number of existing metaheuristic algorithms, they can be divided into two groups: local methods, also called trajectory methods, that choose a global optimal solution among the candidate solutions found by local search in narrow subsets; global methods, also called population-based methods, which aim at finding the optimal solution by performing a search in several zones of the search space.

The two most important local methods are Tabu Search and Simulated Annealing.

Global methods try to simulate a mechanism called natural selection: as it happens with living beings, which evolve towards the fittest individual, it also happens in metaheuristic algorithms, where the search for the optimum in the searching space is performed in parallel by a population of entities. These entities could be chromosomes, particles, bacteria, ants, fireflies, bees, etc., with their corresponding evolution strategy represented by the algorithms.

Among the several global methods existing, basing on [6,84,92], the most widely used are genetic algorithms, particle swarm optimization and ant colony optimization.

In the following, a diagram of the metaheuristics' large population [93] and the metaheuristics this thesis is focusing on are depicted in Figure 3.7 and Figure 3.8 respectively:

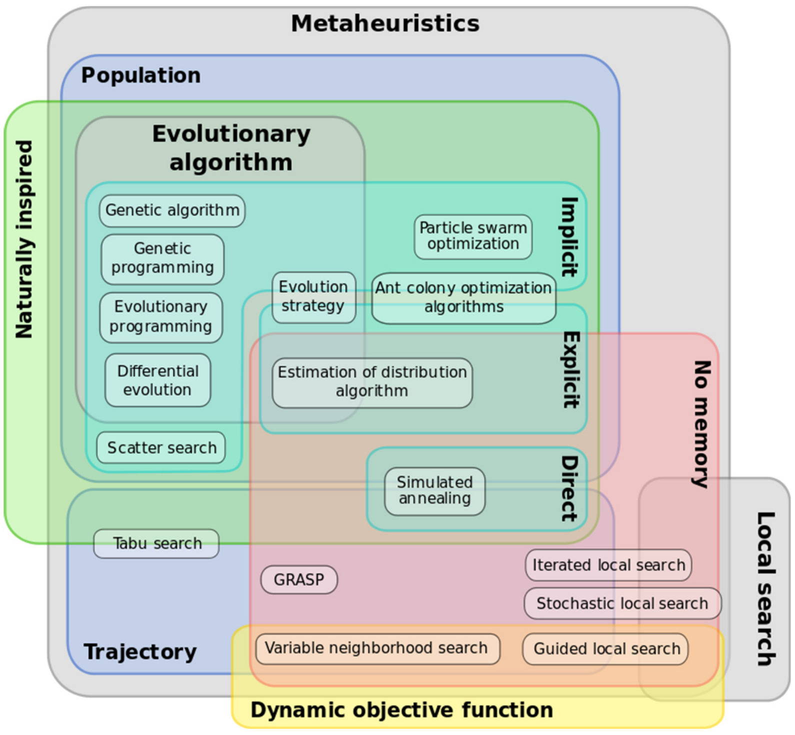


Figure 3.7: Euler diagram of the different classifications of metaheuristics.

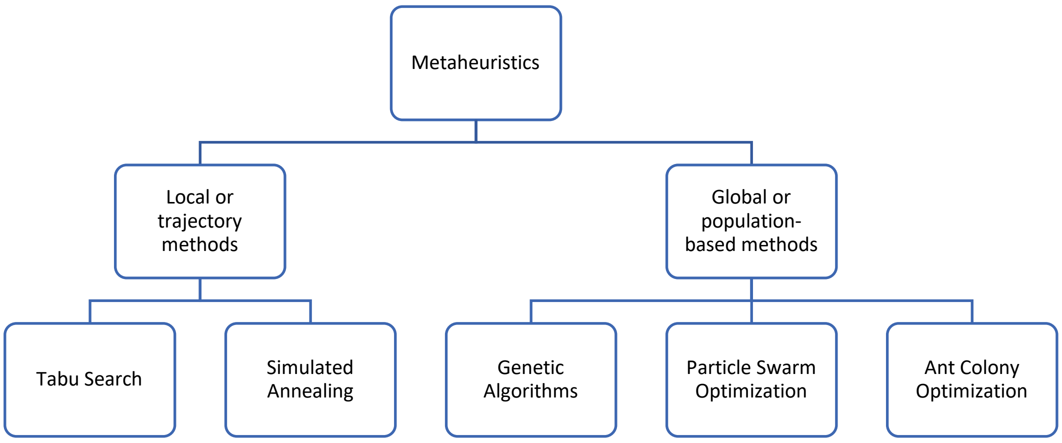

Metaheuristics
Local or trajectory methods
Global or population-based methods
Tabu Search
Simulated Annealing
Genetic Algorithms
Particle Swarm Optimization
Ant Colony Optimization

Figure 3.8: Metaheuristics of interest.

It is important to highlight that most of the theoretical part about Metaheuristics is based on the book [91].

Tabu Search (TS)

Tabu search is based on the Greedy descent algorithm: the search starts from a possible solution x_i of the search space, then the focus is on finding a better solution x_j on a local vicinity $\mathbf{V}(x_i)$ able to decrease the function f. The vicinity is a set of possible solutions that can be reached after a movement from x_i and it is built by a random number generator. The drawback with Greedy descent algorithm is the risk of stopping the search on a local minimum.

Tabu search is an improvement of the previous algorithm because it introduces some further rules to avoid premature search's stops: like before, the algorithm starts from a feasible starting point x_i, but movement to a worse solution x_j in the vicinity $\mathbf{V}(x_i)$ is permitted, even if the function gets worse $f(\mathbf{x}_j) > f(\mathbf{x}_i)$; in this way the algorithm is able to explore more search areas. To avoid infinite search loops, once a point (solution) has been searched, it enters in a taboo list and it is never focused again on the future, it becomes untouchable, "taboo" (that gives the name to the method). Once a solution is in the taboo list, it cannot be accepted in the vicinity of the next iteration.

The size of taboo list is an important parameter of search: if it is too large, there is a bigger risk of avoiding the global optimum (less accuracy); if it is too small, changes are the algorithm is slowed down by loops (more time consuming). Thus, a balance between accuracy and computational time must be found. Usually, the taboo list is of constant length, so once the maximum number of solutions in the taboo list is reached, in the new iteration the older solution goes out of the list and the new solution enters the list, in a last-in-first-out (LIFO) behavior. There are also applications where the taboo list length is modified at each iteration between a minimum N_{min} and a maximum N_{max}.

In a simple, non-exhaustive way, the tabu search algorithm can be written like this:

Algorithm 3 Tabu Search (TS) algorithm

```
1:    Input data:
          ▪ Search space S (space of interest that contains the global optimum).
          ▪ Function $f$ to be optimized.
          ▪ Types of allowed movements starting from a solution.
          ▪ Minimum number of solutions for the current vicinity, $N_v$.
          ▪ Bounds of taboo list length: $N_{min}$ and $N_{max}$.
          ▪ Upper boundaries of counters
          ▪ Maximum number of iterations, $K$.
          ▪ Accuracy threshold, $\varepsilon > 0$.
2:    Initialization:
          a) Select at random the starting point $x_{0,0} \in S$.
          b) Evaluate the starting point performance: $f(x_{0,0})$.
          c) Initialize the optimal solution: $x^{opt} = x_{0,0}$.
          d) Initialize the taboo list: $T_{0,0} = \{x_{0,0}\}$.
          e) Set to zero all the counters.
          f) Set the position of the initial solution: $i = 0$.
3:  At the iteration $k \in (0, K-1)$:
    Starting from the current solution, specify the finite set of all the possible movements.
    Initialize the vicinity of current solution.
    **While** *the vicinity includes less than $N_v$ points* **do**
          Call the random number generator to select a possible movement.
          **If** *the new point belongs to the search space but does not belong to the taboo list* **then**
          add the point to the vicinity.

    **If** *the vicinity includes at least one point* **do**
          Search the better solution among the points of the current vicinity.
          Update the new solution: $x_{k+1,j} = x_{k,j}$, where $x_{k,j}$ is the new solution and $j$ its
          position.
          Add the new solution to the taboo list: $T_{k+1,j} = T_{k,i} \cup \{x_{k+1,j}\}$.
          Update the solution position: $i = j$.
          **If** *the performance of the new solution $f(x_{k+1,j})$ is better than the current optimal
          performance $f(x^{opt})$* **do**:

              Update the optimal solution: $x^{opt} = x_{k+1,j}$.
              **If** *the termination conditions are met (threshold counter)* **go** to final stage
              (no. 4).
              Update the taboo list removing the older solutions

          **Otherwise**, the optimal solution does not change and **then**

              **If** *the termination conditions are met (blocking counter)* **go** to final stage
              (no. 4).

          **Proceed** with the next iteration: $k \leftarrow k + 1$

    **Otherwise**, since the current vicinity is void, the procedure is blocked and the main
    loop has to be broken. **Go** directly to final stage (no. 4).
```

```
4:   Return:
         ▪ The current optimal point: $x_k$.
         ▪ The current optimal performance: $f(x_k)$.
```

The code parts about the counters are represented by the formulation "If the termination conditions are met, go to final stage (no. 4)", a more exhaustive explanation can be found in [91]. However, two counters are used in this algorithm: the threshold counter m, which counts the found solutions that comply with the accuracy threshold (if $\left|f\left(\mathbf{x}_{k+1,i}\right) - f\left(\mathbf{x}^{\text{opt}}\right)\right| < \varepsilon$, increment the threshold counter: $m \leftarrow m + 1$) and stops the search when $m > M$ maximum number; the blocking number n, that counts the number of iterations during which the optimal solution did not change and stops the search when $n > N$.

Notice: as said previously, tabu search can accept solutions that are worse than the previous one, in fact in string 3.4.2. the (new) updated solution could be worse than the old one, but still it is added to the taboo list and the position is updated, allowing the user to search an optimal solution in other regions.

A flow chart can explain in a more effective way how Tabu Search works:

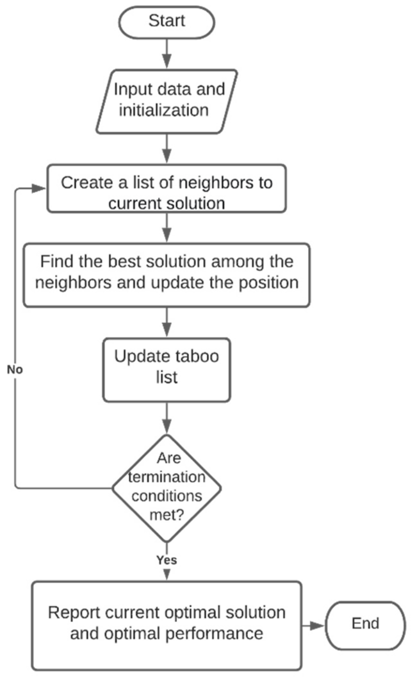

Start
Input data and initialization
Create a list of neighbors to current solution
Find the best solution among the neighbors and update the position
Update taboo list
Are termination conditions met?
No
Yes
Report current optimal solution and optimal performance
End

Figure 3.9: Flow chart of Tabu Search algorithm

[10] the authors present a Pareto-optimality-based multi-objective tabu search (MOTS) algorithm to the machine-part CF problem. In the case of a multi-objective

problem (more than one objective function), an optimal solution can be stated by the Pareto optimality: a solution is optimal if it is non-dominated, meaning the values of its objective functions are better (minimum or maximum depending on the objective) than the ones of any other solution.

The objective functions to minimize are three: the sum of the total investment cost and the inter and intra-cell transportation cost; the intra-cell machine loading unbalance and the inter-cell loading unbalance. The main constraints regard the machine capacity and number, the operation sequence and the workload balancing.

In the case of CF problem, a solution handled by the tabu search algorithm consists of cells containing several machines. The tabu search produces new neighborhood solutions through two kinds of movement operations. One operation is an insertion, in which a machine is randomly chosen from its current cell and reassigned to the other cell. Another is a swap operation, in which randomly chosen machines in two different cells are interchanged with each other. In this paper, only one of the operations is done in iteration. The algorithm exploits a taboo list called external archive *P′* able to sort, add and eliminate solutions based on the values of objective function they give. In the following algorithm, v is the predetermined maximum size of *P′*, s(=(0.1-0.2)v) is the number of the solutions executing a new search in each iteration, The tabu search can be schematized in this way:

Algorithm 4 Pareto-optimality-based multi-objective tabu search (MOTS) algorithm

```
1:  Initialize counters.
2:  Stochastically generate an initial feasible solution satisfying all constraints, reproduce
    the initial solution s-1 times and assign them into P′.
3:  Add all parts into the part list T. Based on counters, do the following on the current
    solution or on a solution of P′:
4:          Randomly move machines into cells to satisfy a constraint about the number
            of machines into cells, according to the allowed movements
5:  Find parts with the lowest inter- and intra-cell transportation cost from the remaining
    parts in the part list T and assign all operations of the part to the corresponding
    machine, compute again the actual capacity of those machines and delete the part
    from the part list. Repeat the procedure until the part list T is empty.
6:  Calculate the objective function vector of the new solution.
7:  Rearrange the archive P′.
8:  If the termination conditions are met stop, otherwise proceed with the next iteration and
    go to step 3.
```

The algorithm finally gives the best result in terms of minimum objective function. The result is represented by a certain number of cells (set by the user) containing machines already in the correct sequence. In the paper, the higher problem is made by 18 machine types, and the method is able to provide different solutions based on the layout (linear single-row layout and linear double-row layout) and number of cells (from 3 to 5). The computational time is very short, a little less than 10 s.

[67] is about the resolution of the CF problem by means of a tabu search based on similarity coefficient combined with a genetic algorithm. The problem takes as main inputs the alternative process routings for each part type and the machine reliability, represented by the mean time between failures (MBTF). The objective function is the minimization of the sum of total intercellular movement cost and the machine breakdown cost.

The proposed tabu search procedure consists of two stages: the initial solution construction and the improvements stage.

In the first step a similarity coefficient-based method is used: for each pair of machines a similarity coefficient S_{ij} can be calculated by means of Equation (3.3):

$$S_{ij} = \frac{N_{ij}}{N_i + N_j - N_{ij}} \tag{3.3}$$

where N_i, N_j and N_{ij} are respectively the production volumes of machine *i*, machine *j* and of the couple between the two machines. With the term production volume of machine *i* and machine *j* is intended the total number of parts, coming from different part types, that cross each machine. Instead N_{ij} is the total number of parts that travel both machine *i* and machine *j*. Then, based on the similarity coefficient of each pair of machines, a similarity matrix is formed and the machines with a higher similarity coefficient are put in the same cell. In the step the alternative routings that result in a lower sum of intercellular movement costs and machine breakdown costs are chosen. Finally, after the formation of part families done in a way equal to cluster analysis, the initial solution is found.

In the second step there is the tabu search combined with genetic algorithm. From the current solution, a neighborhood, that is the set of all feasible solutions reachable by an insertion-move, is generated. The insertion-move is an operation that moves a machine from its current cell to a new cell. All the solutions reached by the algorithm are added to the taboo list, and the best solution among the ones in the neighborhood is selected as the current solution. After a certain number of iterations, if the solution does not improve (i.e., a stagnating area of solutions has been reached; higher risk of local minima) the mechanism of mutation takes place: a machine is reassigned to any cells other than the current one based on a chosen probability. Thus, all machines are probable to change cell when mutation is applied. Notice that the mutation must satisfy the upper and lower limits of cell size. Thanks to mutation is possible to explore more search areas. Finally, the algorithms stop after reaching a maximum number of iterations or if the result does not improve within a certain number of iterations. This method is used in big problems like 30 machine types, 70 part types and 149 alternative routings (more or less 2 alternative routings per part type) with a computational time of around 8 s.

A similar tabu search genetic algorithm method can be found in [33] by the same authors.

Simulated annealing (SA)

Simulated annealing owes its name by the physical phenomenon exploited in metallurgy: annealing. Especially used for steel, annealing is a series of cooling and heating to bring the molten material to its final state, with the right amount of elasticity and stiffness. The solid obtained corresponds to a global minimum of internal energy. Without annealing, if the metal is cooled down too fast, a stiff but fragile metal is obtained, instead if the cooling is too slow, an elastic but too soft solid is produced.

The simulated annealing algorithm reproduces this phenomenon in a simplified way. The internal energy is here the objective function to minimize, whereas the temperature is a parameter directly related to the convergence. As the temperature decreases, the state of minimum of internal energy is wanted, until the global minimum is found.

A diffused model of simulated annealing is Kirkpatrick's model of thermal annealing, introduced by Kirkpatrick Scott in 1983. The objective function to minimize *f*, is referred to as energy. Starting from an optimal solution xi of energy $E_i = f(x_i)$, another neighbor state xj can or cannot be select as the next optimal state depending on its energy, $E_j = f(x_i)$. If the energy decreases $E_j < E_i$, then xj substitute xi as the new optimal solution. Otherwise, if $E_j > E_i$, xj can replace xi only if a requirement is met.

If the energy variation $\Delta \mathrm{E}_{j,i} = \mathrm{E}_j - \mathrm{E}_i$ is positive, then:

$$\mathbf{p}\left(\Delta \mathrm{E}_{j,i}, T\right) = \exp\left(-\frac{\Delta \mathrm{E}_{j,i}}{T}\right) \tag{3.4}$$

is the probability to obtain this shifting from xj to xi at temperature $T > 0$. By normalizing the energy difference and the temperature, the probability surface can be drawn [91]:

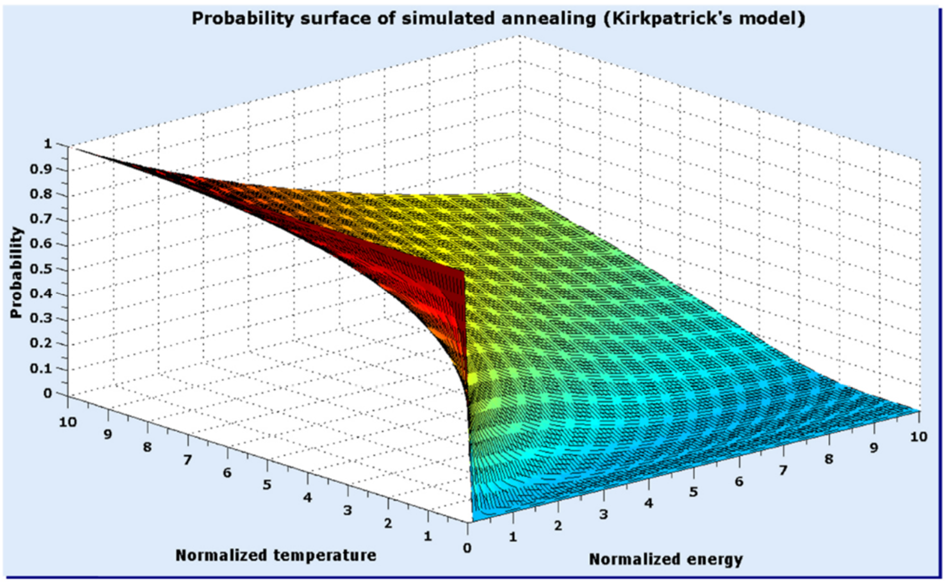


Figure 3.10: Probability surface of Kirkpatrick's model.

From the diagram, it follows that at higher temperatures larger energy variations are more probable that at lower temperatures. In this way, the more the algorithm goes towards the minimum (the more the temperature decreases), the less is the algorithm prone to change area of search, preferring lower energy variations. In practice, a random number generator generates a probability threshold $p_i \in (0,1)$. If

$\mathbf{p}(\Delta E_{j,i}, T) > p_i$ then x_j will replace x_i, otherwise the algorithm will search for a new state in the neighborhood of x_i.

If no improvement is obtained after a certain number of iterations, the search is restarted at a lower temperature T. After some cooling steps, if also decreasing the temperature does not improve the result, the search is stopped.

As previously done for Tabu Search, a simple Simulated Annealing algorithm can be written:

Algorithm 5 Simulated Annealing (SA) algorithm

```
1:    Input data:
            ▪ Search space.
            ▪ Energy f.
            ▪ Accuracy threshold, ε > 0.
            ▪ Upper boundaries of counters.
            ▪ Max temperature to test, T.
2:    Initialization:
            a) Select a random starting state $x$.
            b) Evaluate the initial state energy: $E = f(x)$.
            c) Set the first minimal solution $x^{min} = x$ and $E^{min} = E$.
            d) Set to zero all the counters.
3:    Perform cooling down. **For** the current state $x$:
      Randomly generates a feasible offset and a direction: $\Delta x$.
      Define $x^{rep} = x + \Delta x$ as the state that might replace the current state
      Compute the energy difference with respect to the current minimal state: $\Delta E = E^{rep} - E^{min}$.
      **If** $\Delta E < 0$ **then**
            Replace the current minimal state and energy: $x^{min} = x^{rep}$ and $E^{min} = E^{rep}$.
            Replace the current solution: $x = x^{rep}$ and $E = E^{rep}$.
            **If** *the termination conditions are met (energy counter)* **go** to final stage (no. 4).
      **Otherwise**, the energy of the possible new solution is higher or equal to the energy of the current solution and **then**
            **If** *the immobility counter* $n \leq N$ **do**
                  Compute the energy difference with respect to the current state $\Delta E = E^{rep} - E$.
```

```
                Compute the Kirkpatrick probability $\mathbf{p}(\Delta E, T)$.
                Generate a random number $p \in (0,1)$.
                **If** $\boldsymbol{p}(\Delta E, T) > p$ **replace** the current solution: $x = x^{rep}$ and $E = E^{rep}$.
                **Otherwise**, keep the current solution.
            **Otherwise**, the temperature has to be decreased:
                Randomly generate a new temperature, a smaller one, in the interval
                $(\alpha T, T)$, where α is set by the user.
                **If** *the termination conditions are met (temperature counter)* **go** to final stage
                (no. 4).
        Resume the search from step 3.
4:      Return:
            ▪ The minimal current state: $x^{min}$.
            ▪ The current minimal energy: $E^{min} = f(x^{min})$.
```

In this case the counters that prevent infinite loops are three: the temperature counter k, which counts how many temperature changes are necessary before replacing the current state; the energy counter m, that counts the number of successive states that cannot improve the energy beyond the accuracy threshold; the immobility counter n, which counts the number of attempts to change the minimal state at constant temperature.

[34] is a simulated annealing based on similarity coefficients. The two types of similarity coefficients are: machine chain similarity coefficients (MCS), based on the production volume between two machines; parts similarity coefficients (PS), that consider the similarities in terms of shared machines (by looking at the process routings) between two parts. These coefficients are done for each pair of machines and each pair of part type.

Unlike the general algorithm of Simulated Annealing explained above, the authors introduced a temperature length L, that is a fixed number of iterations that can be done at each temperature level. Thus, the performance of the algorithm is influenced by this parameter, together with other parameters like the initial temperature T_0, that should be high enough to accept in the first iterations also worst solutions (to search more areas of solutions), and the temperature-reducing function, that generally is geometric: $T_i = CT_{i-1}$ and C is typically $0.7 \leq C \leq 0.95$.

The proposed algorithm has three main steps: finding the best alternative routings, grouping machines, and grouping parts. Each step has its specific SA algorithm.

In the first part, taking as input the production volumes and the alternative process routings for each part, an initial solution is generated randomly in the form of a vector $\boldsymbol{A} = [a_1, a_2, \dots, a_N]$, where a_i is a part type, N is the number of part types, and each part type can take an alternative routing $1 \leq a_i \leq p_i$ where p_i is the number of possible alternative routings for each part type. Then, at each iteration the neighborhood solution is generated randomly by inverting the components of vector $\boldsymbol{A}$. The output of this phase is the best combination of the parts routings.

In the second step, the objective function is to maximize the similarity of machines that belong to the same cell. The inputs of this phase are the best parts routings found in the first phase, the maximum number of machines allowed in a single cell, the number of cells, and the production volume of each part. In this way, starting from the MCS matrix, derived from MCS coefficients, a block diagonal form of the MCS matrix is obtained, which shows the machines set belonging to each cell.

In the third phase parts are assigned to cells, found in the previous phase, in a way that the total intracellular movement is minimized.

This SA algorithm was used for a problem made by 17 machine types and 30 part types, and a solution was found in 446s (≅7.5 minutes).

[39] presents a parallel multiple search path Simulated Annealing for the CF problem. Starting from an existing layout made of 25 square-shape locations, the authors have the duty to assign locations to cells and machines to cells, minimizing a weighted sum of costs made by material handling costs between all pairs of locations and intercellular movement costs.

An important aspect of SA is the definition of the solutions, that must be consisted with the algorithm's code. Each solution is represented by a string of values, divided in three segments: the left-hand side segment (LHS-Segment), that contains the information about which cells are assigned to which locations; the middle-segment (MDL-Segment), tells which location machines are installed; the right-hand side segment (RHS-Segment) encodes the part route and size of sublots in each route.

Afterwards, pseudocodes for randomly generate a solution and evaluate its performance are presented. Then, the singularity of this SA algorithm is explained: to find the solution, instead of using a single-search path (like could be the one presented at the beginning of the paragraph), a multiple search path is exploited; in this way, a local minimum is less probable to be the optimal solution, and the algorithm is faster in terms of computational time. In practice, the search is divided in p processes (that is one computing unit, core, or cpu), each of them has S search paths. Thus, each search path starts with its own randomly generated solution, in order to cover more search areas. For better result, the search paths within a process may communicate every Z1 iteration to start the search from the best solution so far known within that process at the current temperature level. Moreover, the processes may communicate every $Z_2 \gg Z_1$ iteration to restart their search paths from the best solution so far known across all the processes. The paper demonstrates the benefits of these interactions in reducing the value of the objective function.

This algorithm is used for a very large problem composed by 192 machine types and 600 part types.

Genetic algorithms (GA)

Genetic algorithms (GA) are the first evolutionary metaheuristics that will be discussed. GA are used to simulate the evolution of microscopic beings like chromosomes. To explain this method, a vocabulary taken from natural genetics is used:

- *Chromosome*: is a long DNA molecule containing all the genetic material of an organism. For the scope of simplicity, a chromosome can be seen as a binary chain composed by zeros and ones.
- *Gene*: is a functional block of a chromosome that encodes a specific protein. Each gene defines a characteristic of the living organism like height, eye color, nose shape etc. The position of a gene is extremely important, modifying its position is possible to have important changes in the basic characteristics of the body. Returning to the binary schematic, the

chromosome can be divided in smaller groups called genes, which are the building blocks of the binary chain.

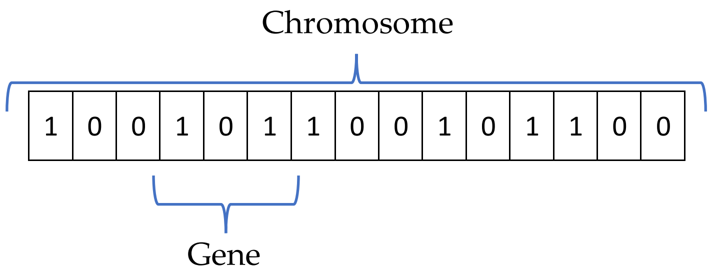


Figure 3.11: Chromosome schematized as a binary chain.

In Figure 3.11 the binary schematization of a chromosome is showed.

During evolution, the chromosomes' group or population develops, generation after generation, by means of reproduction and mutations, until a global optimum (the resolution of the problem) is found. The genetic code is transferred from the older generation to the new one by means of three genetic operations:

- *Crossover*: exchange of genes between two parents. The result are two offspring (children) that have genes from both parents. The genes that are exchanged are chosen by two parameters: the pivot, which indicates the position in the chromosome and the length, which sets the number of positions involved in the crossover. Usually, these two parameters are randomly chosen. This is what happens for example in animals' reproduction. Crossover operation is depicted in Figure 3.12.
- *Mutation*: random change of one or more positions in a chromosome. The outcome is a mutated chromosome (mutant). The aim of this operation is to prevent a local minimum: by generating mutants, almost always worse than the previous individuals, and breed them with more capable individuals by crossover, the resulting offspring would be quite away from the current search zone. In this way, the local minimum trap is avoided. From this operation the initial chromosome and the mutant are obtained. Mutation operation is showed in Figure 3.13a.

- *Inversion*: inversion in the sequence of the genes inside the chromosome's chain. As in crossover, pivot and length parameters must be specified. Inversion can be seen as a type of mutation. From this operation the initial chromosome and the inverted one are get. Inversion operation is displayer in Figure 3.13b.

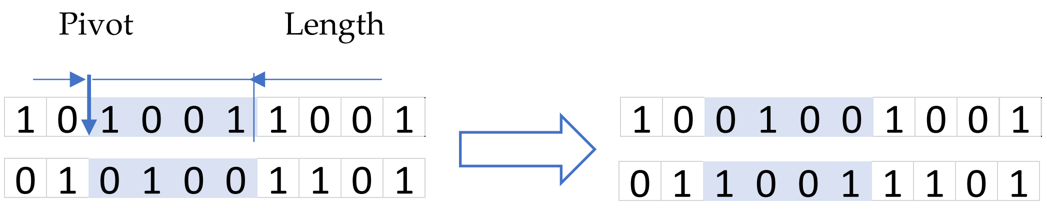


Figure 3.12: Crossover genetic operation.

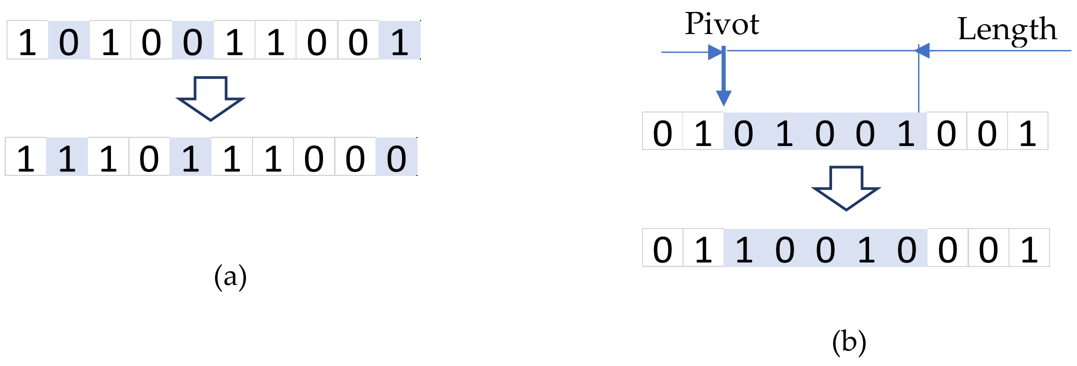


Figure 3.13: a) Mutation and b) Inversion genetic operation

The crossover probability P_c is much larger than the others of mutation P_m and inversion P_i; usually $P_c \in [0.5,0.95]$ and $P_m, P_i \in [0.005,0.1]$. In fact, while the main job in searching the optimum is done by crossover, mutation and inversion are adopted to keep a certain diversity in the population.

After the genetic operations of crossover, mutation and inversion, it must be checked that the offspring fall inside the search space, if not, the quicker operation from a computational point of view is not to remove them but to adjust their form to make them viable.

Another issue to answer is which are the chromosomes available for reproduction. To measure the performance of each individual, a fitness function is used. In fact, the best, more performant chromosomes are the ones having a high value of fitness function which has the duty to represent the objective function that, based on the problem, could be a maximizing or a minimizing one. In nature, usually only the strongest individuals (with a high level of fitness) are capable to reproduce; in GA instead, also a certain percentage of less capable individuals (low level of fitness) are chosen in order to keep a certain diversity in the population, avoiding the local minimum trap, but not too much to avoid the scattering of the population and therefore a too slow search of the optimum. Thus, in GA there must be a trade-off between the diversity of the population (the exploration capacity) to search more areas and the evolution speed (the exploration capacity) to quickly find the optimum solution. There are several selections (rules) to choose from, they can be checked in the book [91].

After reproduction, a too large group of chromosomes is obtained, made by parents, offspring and mutants. In fact, in GA the population (total number of chromosomes) remains equal at every generation, thus a selection rule for survival is necessary. For this purpose, the main rules are: generational selection, where only offspring and mutants survive, with the risk of losing the current global optimum (if it was of the previous generation); elitist selection, where the best individuals are maintained in the population while the other chromosomes are replaced with their offspring and mutants; and the generational elitist selection, that combines the advantages of the previous two.

Another important issue is the selection of the configuring parameters, which are the ones that influence the performance of the GA, in terms of computational time and quality of the solution. Some of these parameters are: population size; number of genetic operations to apply at each generation; probability of crossover, mutation and inversion genetic operation; method for reproducers selection; method for survivals selection; stop tests (termination conditions) selected.

The general GA structure can be depicted in Figure 3.14:

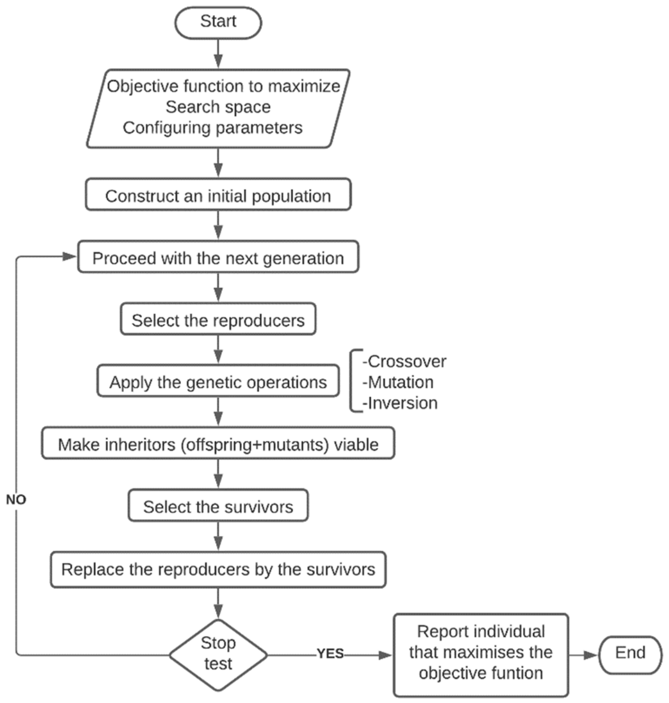

Start
Objective function to maximize
Search space
Configuring parameters
Construct an initial population
Proceed with the next generation
Select the reproducers
Apply the genetic operations
-Crossover
-Mutation
-Inversion
Make inheritors (offspring+mutants) viable
Select the survivors
Replace the reproducers by the survivors
Stop test
NO
YES
Report individual that maximises the objective funtion
End

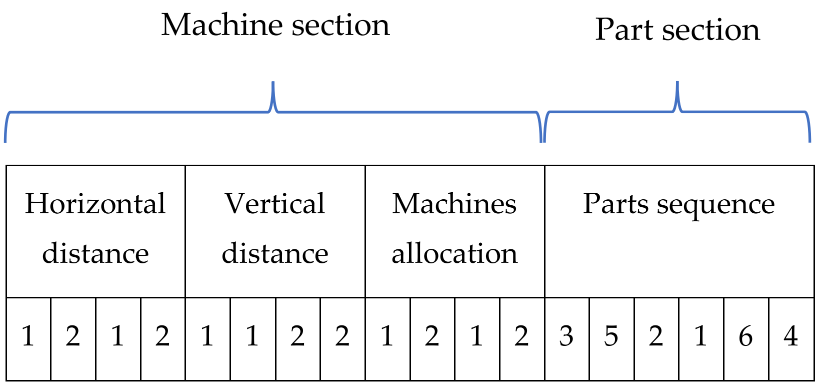
Figure 3.14: Flow chart of Genetic Algorithm.

For more information about the GA consult the book [91].

[41] is a multi-objective GA (MOGA). The aim of this model is to form manufacturing cells, grouping machines, considering the cells inside the layout and operations scheduling in a way that minimizes the total transportation cost of parts and makespan. Makespan is the time that passes from the start to the finish of a job. Minimizing this time, the firm would be more efficient, reducing the work-in-progress (WIP) level, and can satisfy tighter customer request.

The GA works in this way: the population evolves through generations by maximizing a fitness function based on minimizing the first objective function (total transportation cost); when the best solution is found, it is added to the set of Pareto optimal solutions, and its second objective (makespan) is calculated. Then a new era (a sequence of generations) begins, and the fitness function used is based on the makespan calculated from the optimal solution of the previous era. Thus, era after era, the upper bound made by the makespan is more and more restrictive, allowing only solutions that are increasingly more performing in terms of minimum total transportation cost and minimum makespan.

An important step in GA is the definition of the chromosomes. The chromosomes have two main sections: the machine one, and the part one. The machine section, of length 3*m*, where *m* is the number of machines, is further divided in 3 subsections of length *m*, where the first two subsections depict respectively the abscissa and coordinate positions of each machine in a Cartesian reference system, and the third subsection indicates the allocation of machines to cells. The part section has length *n*, where *n* is the number of part types, and represents the scheduling sequence for parts. Directly from the paper, a chromosome following this definition can be depicted:



Horizontal distance				Vertical distance				Machines allocation				Parts sequence					
1	2	1	2	1	1	2	2	1	2	1	2	3	5	2	1	6	4

Figure 3.15: Chromosome representation for the first GA example.

The problem consists of 4 machines disposed in 2 cells and 6 part types. Taking as an example machine two, it is at horizontal distance 2, vertical distance 1, and is allocated in cell two. The parts sequence tells which part must be machined first, in this case part number four. From this description derives that parts are not grouped in part families and sent to the specific cell, they can be seen as jobs. In fact, this parts sequence can be directly translated into a scheduling sequence.

The first two subsections are implemented to avoid overlapping cells and promote rectangular-shaped cells.

The higher problem solved is made by 20 machines, divided in 5 cells, and 25 part types.

[42] is an Adaptive GA (AGA). The term adaptive means that the probabilities of crossover and mutation P_c and P_m are adaptive to achieve a good trade-off between the exploration and the exploitation abilities of the GA without burdening on computational time. The objective function to minimize is the sum cost of three terms: machine utility cost, crossflow cost, and the intercell cost.

As usual, the chromosome's definition is a major aspect of the GA:

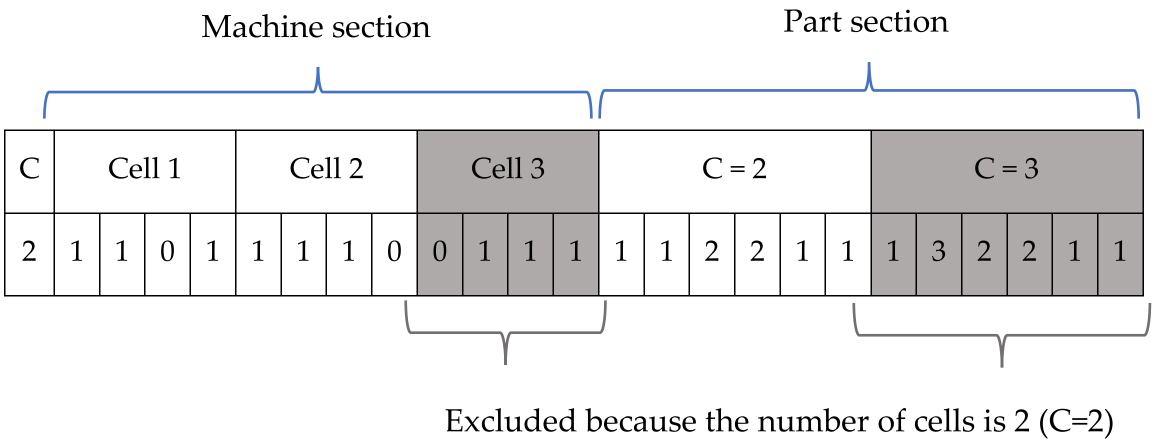


Figure 3.16: Chromosome representation for the second GA example

Looking at Figure 3.16, in this example there are 4 machines, 6 parts, and the maximum number of cells allowed is 3.

C is a random integer between 2 and the maximum number of cells C_{max}, and stands for the allowed number of machine cells in the solution. Thus, from the same run of GA is possible to explore several solutions with different number of cells. C = 1 is not included because it is the trivial solution (all the machines are grouped in one single cell).

In the machine section the machines are randomly assigned to the cells by means of a binary system, where "1" stands for machine m is assigned to cell c, and "0" means the machine m is not assigned to cell c. This section is long $M \times C_{max}$, where M is the number of machines. Each subsection (cell) is long M. From this definition, duplication of machines is allowed, for example machine two is placed in cell 1, 2 and 3.

In the part section, part types are randomly assigned to cells by placing the cell number (that goes from 1 to C_{max}).

The individuals for reproduction are selected based on their fitness value: chromosomes are ranked by fitness value and divided in three equally distributed groups called Best, Medium, and Worst Group. In the Best Group, the most performant individual becomes the elite individual and is preserved for the next generation. Randomly, from all the three groups, individuals for the mating pool are selected, half of which comes from the Best Group and the rest from Medium and Worst Groups. In this way, without excluding entirely the worst solutions a good trade-off between exploration and exploitation is achieved. Then, the inheritors (offspring and mutants) substitute the individuals of the previous generation.

The larger problem managed by this GA is 20 machines divided in 3 cells, and 25 part types.

Other applications of GA besides CF problem can be found in [94,95].

Particle Swarm Optimization (PSO)

In Particle Swarm Optimization (PSO), entities called particles are adopted. These particles, that together form the swarm, are both autonomous and social entities and, communicating with each other, are able to find an optimum (or near-optimum) solution to the problem.

At first, the particles are uniformly distributed inside the search space, that for example could be a hypersphere or a hypercube; each position covered by a particle is a solution characterized by its fitness value. Then, iteration after iteration, each particle moves independently based on the information collected by itself in its own path (points/solutions touched by the particle) and by the particles around him, called *informants*. In fact, these particles have both the abilities to evaluate the quality of their own position and keep in memory their best performance (*cognitive consciousness)*, and to ask to the other particles about their positions and best performances (*social consciousness*). The particles are mutually attracted towards the optimal solution (with high value of fitness) that at the end is found.

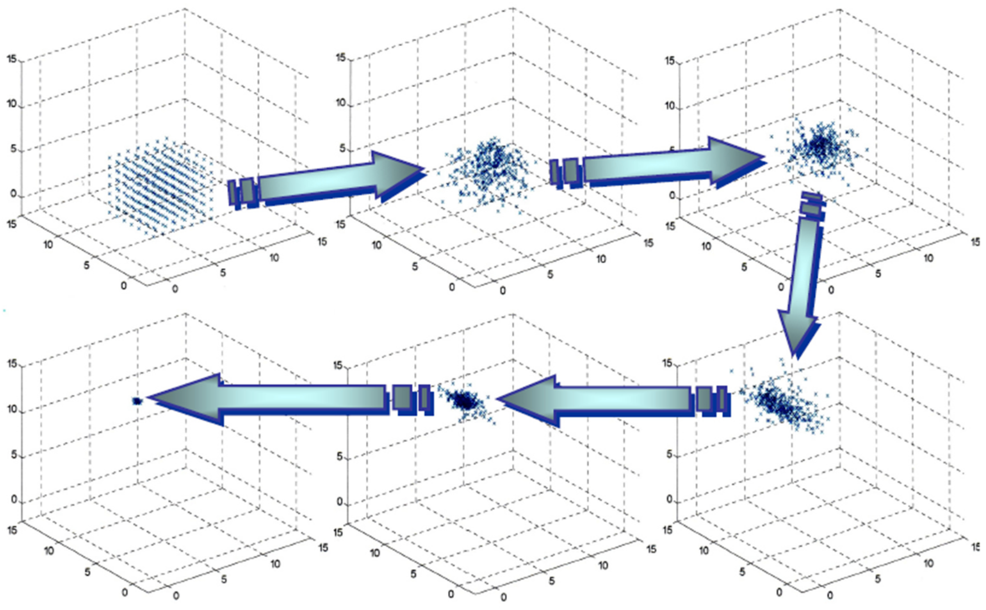


Figure 3.17: Principle of Particle Swarm Optimization.

From the figure above [91], is possible to notice the unique trip that each particle takes and that, in the end, the particles agglomerate around the optimal solution.

How does a particle move? Iteration after iteration, a particle p moves from position x_p^k (k is the iteration) to position x_p^{k+1} with a certain speed v_p^k in a certain time interval ΔT_p^k, as depicted in Equation (3.5):

$$\mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \mathbf{v}_p^k \Delta T_p^k, \qquad \forall k \in \mathbf{N} \tag{3.5}$$

Speed and time interval are two crucial quantities that must be computed at each iteration. The time interval ΔT_p^k can be constant (i.e., unit time $\Delta T_p^k = 1$) or a random number, uniformly distributed in a range $[0, T]$ a priori known, and can be employed to make the new position $\mathbf{x}_p^{k+1}$ viable. In fact, there is the possibility of the new position to be outside the search space, thus the time delay can be reduced to put back the new position inside the allowable space.

The speed v_p^k is composed by three terms: the *adventurous* vector speed v_p^k, that represents the autonomous will of the particle to continue the journey at the current speed; the *conservative* vector speed $v_{p,c}^k$, that stands for the conservative tendency of the particle to go in the direction of the best position found by the particle itself on its own path; the *panurgian* vector speed $v_{p,s}^k$, which takes into account the social consciousness of the particle that blindly follows the direction to the best performance as indicated by the informants. The speed update of each particle can be represented in Equation (3.6):

$$\mathbf{v}_p^{k+1} = \mu_p^k \mathbf{v}_p^k + \lambda_{p,c}^k \mathbf{v}_{p,c}^k + \lambda_{p,s}^k \mathbf{v}_{p,s}^k, \qquad \forall p \in \overline{1,\mathrm{P}}, \forall k \in \mathbb{N} \tag{3.6}$$

Where:

- P is the total number of particles
- μ_p^k, $\lambda_{p,c}^k, \lambda_{p,s}^k$ are scalars representing respectively the adventurous, conservative and panurgian tendencies of the single particle. μ_p^k varies in the interval [0,1], instead $\lambda_{p,c}^k$ and $\lambda_{p,s}^k$ in the interval [0,2]. They can be constant, randomly chosen, or adaptively updated based on the dynamic of the particle swarm, this last version is called Adaptive PSO algorithm.

Thus, a particle's speed is a compromise between its own consciousness and the influence of the other particles. Controlling the speed (especially the scalars μ_p^k, $\lambda_{p,c}^k, \lambda_{p,s}^k$), is possible to decrease the computational time and have a good exploration-exploitation trade-off.

As said, informants influence the particle's movement. At each iteration k, an informant group E_p^k is formed for each particle. There are several strategies to choose the informants. They could be the entire particle swarm but the particle itself, however this choice causes a computational burden for the algorithm; an elite of particles, having a high fitness value, with the drawback of exaggerate the exploitation at the expense of exploration (local minima trap); a trade-off of the two previous strategies, an informant group composed by the best particle, with the higher fitness value, and other particles randomly chosen.

From the original PSO algorithm, different algorithms based on biology were born. Instead of particles fireflies, bats and bees are adopted:

- Fireflies algorithm is based on the mating process of fireflies. Fireflies communicate among them by means of brightness, which is directly related to fitness value. Thus, the fireflies with the higher brightness (higher fitness value) are the more attractive. The movement of the fireflies and the way they perceive light are based on biology.
- Bats algorithm is inspired by the bats’ swarm searching for food. By means of ultrasounds, bats are able to locate and catch the prey by analyzing the amplitude and the rate of the signal bouncing back from obstacles or preys. If the rate decreases, there is the presence of a prey (optimal solution), and the bats fly towards him, otherwise the bats randomly move based on the average amplitude of the ultrasounds sent by the other bats.
- Bees algorithm is based on the cooperative behavior of bees in finding flowers to pollinate. Bees of the same colony are divided in two major groups: scouts, which have the duty to find the best flower beds (solutions with high value of fitness); and foragers, that are hired by the scouts to sample the surroundings of the best flower beds, possibly to find a better solution of the ones suggested by the scouts.

For the sake of simplicity, the three algorithms above are considered PSO algorithms, as opposed to other review papers like [92], that consider them separately.

For more information about PSO, Adaptive PSO, fireflies, bats and bees algorithms consult the book [91].

[54] is a fireflies algorithm applied to CF problem. The objective function to minimize is the number of exceptional elements. In fact, by doing something very similar to the ROC, starting from a parts/machines matrix formulation, the algorithm has the duty

to find the optimal block diagonal matrix depicting the best cells configuration with the minimum number of exceptional elements.

The fireflies used for the purpose are built in the following way: the position covered by the firefly i in iteration k can be written as $Y_i^k = \left(y_{i11}^k, y_{i12}^k, \dots, y_{imc}^k\right)$, where m is the number of machines, c is the number of cells, $y_{ijt}^k = 1$ if machine j of firefly i is placed in the cell t at iteration k and 0 otherwise. Thus, if for example the firefly has $y_{i11}^k = y_{i23}^k = y_{i33}^k = y_{i42}^k = 1$ and the other terms equal to 0, it is representing the following configuration:

		Cell (t)		
		1	2	3
Machine (j)	1	1	0	0
	2	0	0	1
	3	0	0	1
	4	0	1	0

Table 3.1: Possible firefly for the fireflies algorithm.

By looking at the structure of fireflies, it derives that the search space of possible solutions/configurations is made by only machines and cells. In fact, the partition of parts in part families and their distribution among the cells is made by combining a solution with the information coming from the parts/machines matrix, which contains the operations needed by each part. The number of cells is set by the user by specifying the maximum number of machines per cell, so is not possible to search solutions with different number of machine cells with the same run.

The larger problem dealt by the algorithm is made by 20 machines, divided in 6 cells, and by 51 part types.

[80] is an hybrid algorithm composed by PSO and linear programming to solve the integrated CF and worker assignment problem. Linear programming is a mathematical programming based on a mathematical model made by linear relationships.

The objective function is the minimization of the sum cost made by many terms like material handling cost, machine procurement cost, several workers costs, subcontracting cost.

The search space covered by particles is made by two decision variables called x_{krpwc} and z_{rp}.

The first term $x_{krpwc} = 1$ if the kth operation in the r̲th routing of type p part is processed by type w worker in cell c, and 0 otherwise; the second term $z_{rp} = 1$ if the rth routing of type p part is selected for self-manufacturing (as opposed to subcontracting), and 0 otherwise. Thus, by combining these two terms, each particle represents a possible configuration, characterized by part types taking their alternative route inside the machine cells made by possible amounts and types of machines and workers. Then, the linear programming takes the information coming from the particles of the PSO to calculate other important quantities like the quantities of machines of type m and of workers of type w to assign to each cell c, the quantities of type p parts to subcontract or to self-manufacture, and the objective function. Finally, if the termination conditions (i.e., max number of iterations reached, the optimal solution does not change after a certain number of iterations) are satisfied, an optimal solution which minimizes the objective function is found.

The larger problem set tackled by the hybrid algorithm is 30 machines and 15 workers spread between 6 cells, and 40 part types with 89 alternative routings (more or less 2 routings per part type). The problem is solved in 23 minutes and 41 seconds, quite high with respect to other metaheuristics, but it is justified by the complexity of the problem (CF + worker assignment problem) and, as explained by the authors, by the better results with respect to the standard PSO algorithm.

Ant Colony Optimization (ACO)

In Ant Colony Optimization (ACO) the swarm is no more represented by particles like in PSO, but by ants. In nature, ants communicate by means of pheromone to report the shorter route from the nest to the food source. Initially, ants move by random around the nest, each ant releases part of the pheromone on the route it is taking and part at the nest, based on the information found. Finally, the shorter (optimal) route that ants must follow to the food source is the one with the higher pheromone's trail. From this description, is possible to notice that ants' behavior is based on a continuous improvement made possible by the indirect communication between ants by means of pheromones.

Inspired from the ants' natural behavior, the ACO algorithm is organized in the following way. As in PSO, the search space, that could be a hypercube or a hypersphere, is divided in nodes. To construct a route, the ant moves between nodes along arcs, until a predefined number of nodes is not reached. At each iteration, each ant starts from a random point in the search space and, node after node, constructs a possible route. After all the ants have taken a feasible route, these routes are evaluated, based on the distance, on the cost, or on both two. The best route of the iteration in terms of minimum distance and cost will release the higher amount of pheromones along its arcs. Iteration after iteration, the ants will follow with a higher probability the arcs with a higher pheromones' trail, until an optimal route is found.

The components of ACO algorithm and its basic functioning are depicted in the following figure:

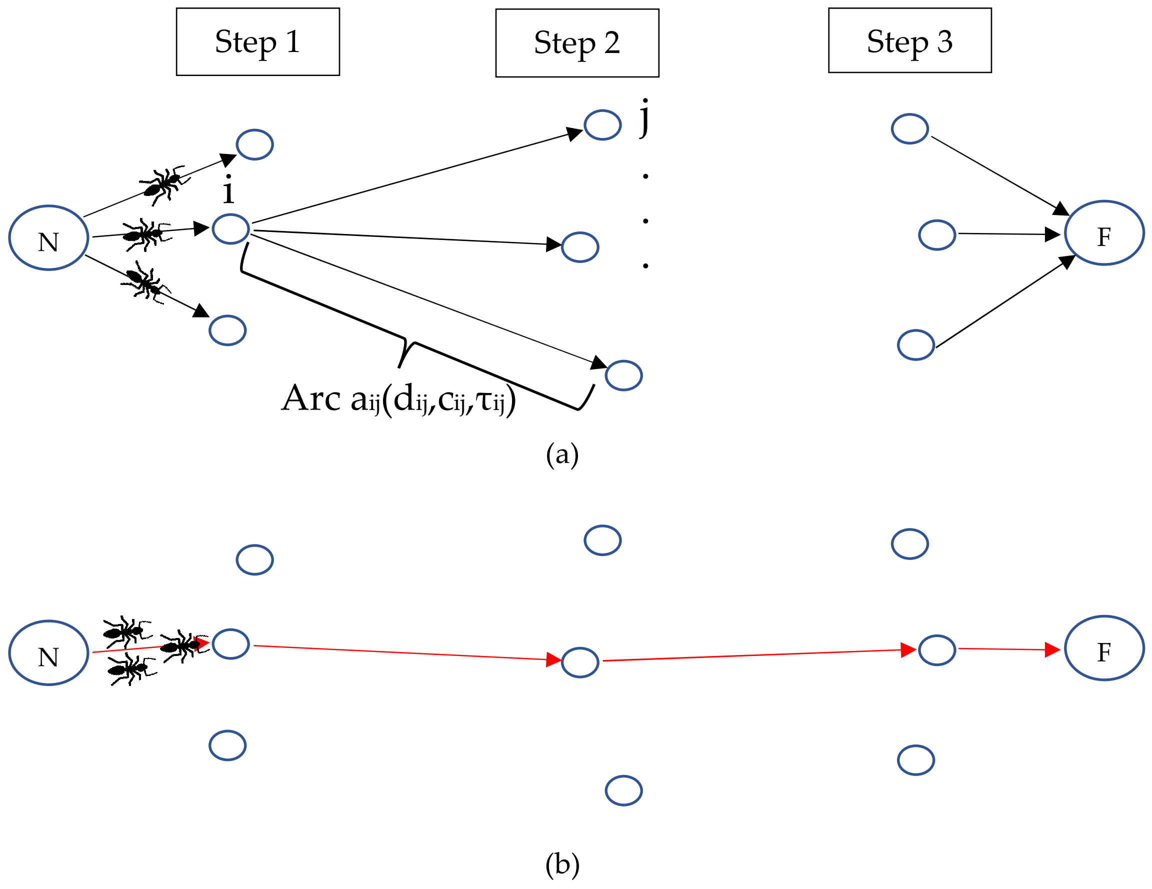


Figure 3.18: General structure of ACO: a) Components of ACO, b) Example's optimal solution.

In this example all the ants start from the same point, the nest ***N***, and cooperate to find the shortest route from the nest to the food ***F***. To do so, the route to take must have the nodes ***N***, ***F***, and three intermediate nodes, one for each *Step*. In this case, the starting point at each iteration remains the same, but in the general case all the ants start from random nodes in the search space.

Nodes are divided in *parents* and *children*. The children are the possible nodes that can be reached from the parent node. In this example, node i of Step 1 is a parent for the children nodes j of Step 2. In fact, due to the structure of the problem, nodes of Step s ($s \in [1,2]$) are parents of nodes of Step s+1. Nodes are connected by means of arcs, depicted by arrows. An arc a_{ij}, which stands for arc from node i to node j, is characterized by its own distance d_{ij}, cost c_{ij}, and pheromones trail τ_{ij}. Thus, to travel from node i to node j using arc a_{ij}, the ant must walk for a certain distance d_{ij} and pay a certain cost c_{ij}, and obviously the objective of ACO is to minimize the total distance and the total cost given by summing the contributions of all the arcs. In the above example, arcs do not have costs since they are not relevant, but only distances. Unlike d_{ij} and c_{ij}, τ_{ij} varies its value iteration after iteration, with an intensity update formula like Equation (3.7):

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{\mathrm{f} \in \mathrm{F}_{\mathrm{ij}}} \Delta\tau_{ij}(\mathrm{f}) \tag{3.7}$$

Where:

- $\rho \in (0,1)$ is the pheromone evaporation factor. As in nature, the pheromone's trail evaporates to allow the exploration of new routes. In the algorithm it is added to have a good trade-off between exploration and exploitation. In fact, without evaporation there would be an excessive accumulation of pheromone's trail in specific arcs, causing a higher exploitation at the expense of exploration, with a high risk of local minimum trap.
- $\Delta\tau_{ij}(\mathrm{f})$ is the amount of pheromone's trail releases by the ant on the arc a_{ij}.
- $\mathbf{F}_{\mathrm{ij}}$ is the set of the ants that traveled in the arc a_{ij} in the same iteration.

From the definitions above, is possible to say that the most prominent arcs, that together will define the optimal route, are the ones with a higher τ_{ij}, in other words the most travelled arcs. The equation above is also called *local* intensity update, it is done at the end of every iteration, after all the ants have travelled a route. In some papers is possible to find also *global* intensity update formulas, it is used after the local one, with the aim of distinguish clearer the optimal route (with its arcs) of the current iteration, adding more pheromone's trail. The optimality of a route could be based on distance, cost, a value of fitness etc. according to the problem.

As said, pheromone's trail τ_{ij}, together with d_{ij} and c_{ij}, influences the ant in the choice of the next node to take by means of a formula like Equation (3.8):

$$\mathbf{P}_{in} = \frac{\tau_{in}^{\alpha}\eta_{in}^{\beta}\varphi_{in}^{\gamma}}{\sum_{j \in \boldsymbol{E}_j} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}\varphi_{ij}^{\gamma}}, \qquad n \in \boldsymbol{E}_j \tag{3.8}$$

Where:

- $\boldsymbol{P}_{in}$ is the probability that the ant chooses the child node n among the set of child nodes $\mathbf{E}_j$ of parent node i.
- $\eta_{in} = \frac{1}{d_{in}}$ is the inverse of the arc's distance d_{in}.
- $\varphi_{in} = \frac{1}{c_{in}}$ is the inverse of the arc's cost c_{in}.
- $\alpha \geq 0,\ \beta \geq 1,\ \gamma \geq 1$ are weights expressing the importance of pheromone's trail, distance, and cost.
- The denominator $\sum_{j \in \boldsymbol{E}_j} \tau_{ij}^{\alpha} \eta_{ij}^{\beta} \varphi_{ij}^{\gamma}$ is added to have a probability $\mathbf{P}_{in} \in (0,1)$.

Thus, the nodes and corresponding arcs with a higher probability to be visit are the ones with low distance d_{in}, low cost c_{in}, and high pheromone's trail τ_{in}.

Once the stopping criterion are met, the optimal route for the example above, with the shorter distance from the nest **N** to the food **F**, is the one depicted by a red line.

[56] is an ACO for scheduling virtual cells. For virtual cell is indicated a set of machines that are not physically grouped together in a manufacturing cell but, as a manufacturing cell does, they produce a part family. Virtual cell turns out to be more flexible with respect to manufacturing cell due to the fact they can be formed at any time a new job/part is available and machines are not moved inside the department, but it will cause a higher material handling costs because the machines can be distant from each other.

The objective function is in fact the minimization of the total travelling distance of materials and component parts for manufacturing. At each iteration, ants start from a random node. Nodes represent operations of jobs. The route that is constructed by each ant it's a possible operation sequence that must be satisfied by the virtual cells. For example, if Job1 has two operations Job1.OP1 and Job1.OP2 and Job2 has three operations Job2.OP1, Job2.OP2, and Job2.OP3, a possible route or operation sequence depicted by an ant could be {Job2.OP1, Job2.OP2, Job1.OP1, Job2.OP3, Job1.OP2}. Then, from each tour, workstations are assigned by means of a simple heuristic code that considers the capability (the ability of the machine to do a certain operation) and the capacity of each machine. From these results, each tour is evaluated in terms of total travelled distance by materials and components, and the best ones are awarded with a higher release of pheromone's trail in their arcs. Finally, an optimal operation sequence is found. Another heuristic code is implemented to properly schedule the production.

The higher problem solved by the algorithm is made by 12 machines and 10 part types produced by means of 10, 30, 60, 90, and 120 jobs. The computational time is in the order of few seconds for each case.

[59] is an ACO for the CF problem. Starting from the part/machine matrix, the objective is to find the block diagonal matrix that maximizes the grouping efficacy:

$$\eta = \frac{A - E}{A + V} \tag{3.9}$$

Where A is the number of operations, E is the number of exceptional elements, and V is the number of voids. For exceptional element is intended when machine i could do an operation on part j (depicted by a "1" in the matrix), but both of two are not assigned to the same cell. There is a void when machine i can't do an operation on part j (depicted by a "0" in the matrix), but anyway they are assigned to the same cell. Thus, grouping efficacy is a way to evaluate the "wasted capacity" of a possible cell configuration.

Routes are constructed in this way: at each iteration, each ant starts from a random node, at each node the ant randomly chooses a cell and, by means of a probability distribution based on part/machine similarities and pheromone's trail, randomly chooses an operation to assign to the cell. The route is built when all the operations are assigned to the cells. Then, by means of a heuristic called Local Search, machines are assigned to operations and the solution found is evaluated. Based on the evaluation, the pheromone's trails are updated, giving an award to the iteration-best route, the one with the higher grouping efficacy. Finally, once the termination conditions are met, the optimal configuration is found.

The higher problem set faced by the algorithm is made by 50 machines and 150 part types and is solved in 20 seconds.

3.3.6 Artificial Intelligence Methodologies

Artificial Intelligence (AI) is the branch of computer science that studies the fundamental theory and applications of computers' hardware and software that mimic human cognitive functions like learning and thinking.

The two most used AI methodologies in CF problem are Artificial Neural Networks (ANN) and Fuzzy Logic (FL).

Artificial Neural Networks (ANN)

ANN functioning is inspired by the biological neural networks of the animal brains. Nodes called artificial neurons are connected by means of synapses. These artificial neurons communicate between them using signals, real numbers computed by nodes by means of non-linear functions. ANN can be classified in three broad categories [96]:

- ART (Adaptive Resonance Theory).
- SOM (Self Organizing Map).
- Other traditional neural network.

ART "is a cognitive and neural theory of how the brain autonomously learns to categorize, recognize, and predict objects and events in a changing world" [97]. Taking as an example one of the five senses, the sight, ART tries to mimic the way the brain of humans and other animals receive the information coming from new objects that are discovered: the bottom-up information coming from the eyes is received by the brain and is categorized based on top-down information (long term memory) coming from experience. In other words, the new object will fall in the category of similar objects already discovered by the animal. The main framework of ART can be simply explained by the two-layer model depicted in Figure 3.19 coming from the paper [98]:

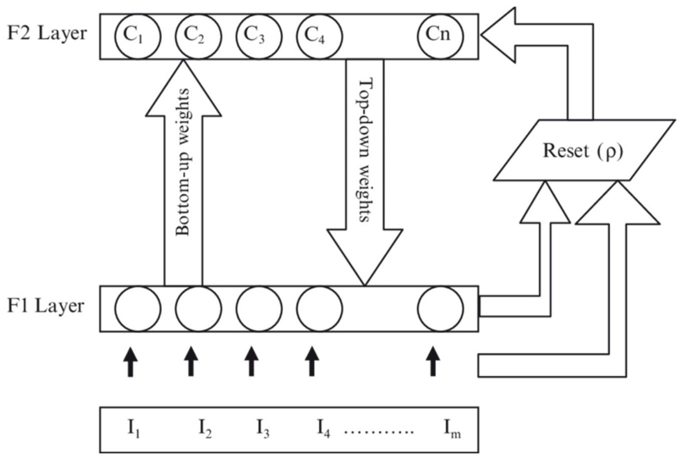


Figure 3.19: Main structure of Adaptive Resonance Theory (ART).

The input vector **I**, that in case of the simplest ART model, ART1, is a binary vector, is read by the first layer of nodes F1 (i.e., the eyes), which elaborates the input to produce a feasible signal, adjusted by the bottom-up weights, that is sent to the nodes of the second layer F2. Some nodes of F2 layer are activated by this signal and interact with the F1 layer nodes by sending another signal modified by the Top-down weights. A Reset unit, characterized by a vigilance parameter ρ, is used to filter out noises coming from these signals. What happens is a cyclic learning process between F1 layer and F2 layer, where in the end the weights are adjusted to better match the input vector and the input vector itself is categorized.

The ART neural network can be trained by using a set of training inputs and then can be exploited to solve problems like the CF problem. In fact, giving as input a part/machine incidence matrix, the ART model can categorize machines and parts basing on similarities and grouping them into cells.

[63] is a modified ART1 algorithm able to solve the CF problem. Taking as input a part/machine incidence matrix containing both process sequence and operational time, this algorithm gives as output the part families (the categories given by the nodes of ART), and then a heuristic code assigns machines to part families to form manufacturing cells. The objective function is the maximization of grouping efficacy, similar to the grouping efficacy already discussed in [59], in a form that takes also into account the operational times. The larger problem solved by the proposed model is made by 35 machines and 90 part types, and it is solved in 1,85 seconds.

[62] is another example of modified ART1. It works similarly to the previous example, with the only difference that takes as input part/machine incidence matrix with operation sequence only. Also in this case the objective is the maximization of grouping efficacy (or efficiency as called in the paper).

ART, and especially ART1, is the simplest and most diffused ANN approach applied to CF problem. To learn more about SOM and other traditional neural networks it is recommended to consult the review article [96].

Fuzzy Logic (FL)

FL is a tool used by decision makers when facing uncertainties and imprecision [99]. Uncertainties can come from internal factors like machines, humans, system issues, and from external factors like varying demand, market forces, competitive behaviors. Exist also data that are not numerically quantifiable as in the case of "linguistic variables" like small/large, low/medium/high. Due to these uncertainties, the decision/plan made by the firm could be outdated, imprecise, not cost effective, and updates along the way (i.e., in terms of capabilities and capacities) will be necessary, with all the related costs. Thanks to FL, the user can employ a tool that mimics the reasoning of human mind when facing uncertainties, giving as response an approximate but realistic solution.

The basic model of fuzzy logic reminds the mathematical formulation already presented in Mathematical programming paragraph:

$$\begin{gathered} min\ \mathbf{c}^T \cdot \mathbf{x} \\ \text{s.t. } \mathbf{A} \cdot \mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{gathered} \tag{3.2}$$

but in the case of fuzzy logic the vector cost coefficients (objective function coefficients) $\boldsymbol{c}$, the vector of right hand-side values (vector of resources) $\boldsymbol{b}$, and the matrix of technical coefficients **A** could be fuzzy, and they are depicted by the apex ~ (tilde): $\tilde{c}, \tilde{b}, \tilde{A}$. Based on the problem, only one or more of these data are fuzzy.

Considering the generic fuzzy number $\tilde{d}$ (d stands for data), it can be represented as a triangular fuzzy number in the following way:

$$\tilde{d} = (a_1, a_2, a_3) \tag{3.10}$$

Where:

- $a_1 = m - \alpha, a_2 = m, a_3 = m + \beta$, where a_1 is the lower bound value, a_2 is the mean value, a_3 is the upper bound value.
- m is the mean value, $\alpha, \beta \geq 0$ are scalars.

Thus, each fuzzy number can be seen as a range of values, due to the fact the decision maker doesn't have enough data to define a precise value. For example, due to poor quality data, the processing time p_{ij} of part i on machine j is given in the form $p_{ij} = (3, 5, 8)$ (in minutes), meaning that the process takes at minimum 3 minutes and 8 minutes at maximum.

How can triangular fuzzy numbers be employed in the mathematical model? They must be defuzzied for example by means of strong probability factor:

$$\tilde{d} = \frac{a_1 + 4a_2 + a_3}{6} \tag{3.11}$$

In this way, from the initial range of values the fuzzy number is reduced to a single number that can be managed by the algorithm.

The main applications of FL are in scheduling, aggregate planning and manufacturing flexibility, and recently in CF problem.

[77] is a fuzzy genetic algorithm (FGA) to solve the dynamic CF problem in a multi-period planning horizon. In a dynamic environment, the cell composition changes at each period, in a manner that follows the variable demand. However, changing the cell configuration would rise several problems related to higher material handling cost and reconfiguration costs, which are costs to purchase new tools and set-up the machine for the new job. That is why the bi-objective of the problem is the minimization of intercellular movements and reconfiguration cost.

The chromosomes of the GA are constructed in a way that consider which machines are in which cells in each period. The GA works as usual (initial population, crossover, mutation, generational replacement etc.), the only difference is that genetic operators (crossover and mutation) are enhanced by fuzzy logic. In fact, each chromosome contains all the configurations in each period, thus is possible to judge is the subsequent configurations are similar or not, based on flow matrices (how many parts flow from one machine to another one) and on reconfiguration cost. By using a linguistic variable *Similarity*, formed by terms {*Weak*, *Medium*, *Great*}, crossover and mutation cut and swap positions accordingly.

The larger problem faced by the algorithm is made by 20 machines, 25 part types and 6 periods. This problem was solved in 817.64 seconds (13 min 38 sec).

3.4 Comparison between proposed methods for CF problem

After the theoretical discourse about methods to resolve CF problem, with the help of some examples, is now possible to discuss the advantages and disadvantages of these methods, extracting also the main dark spots not clearly resolved by this taxonomy.

As initially said, the three major categories of CF methods are Visual methods, Part coding analysis, and Production based methods. Is possible to assert that:

- Visual methods and Part coding analysis are simpler than Production based methods, they can be done "by hand" without the help of a software, or they can be automatized by the help of a CAD-reader [87] or of an AI able to read and categorize the real parts to divide in part families.
- Visual methods and Part coding analysis don't take into consideration any manufacturing information like processing time, process routing, costs, layout constraints etc.; they only care about process requirements, shape and size of the final product, without any reasoning about the routing to take to produce it, and maybe material. Thus, the results obtained by these two methods would be not optimal or feasible in practice.
- Part coding analysis is a more schematic and standardized way to proceed with respect to Visual methods, there is less risk of committing mistakes and the result is more rational and less subjective.
- Visual methods and Part coding analysis are feasible for small scale problems, where the number of parts is small or the variability in shape, size, process requirements is small among the products. Thus, they are a simple and good option for small-medium size companies and for companies with few products.

Due to these reasons, Production based methods seem to be the most promising to solve larger CF problems in a more effective way. They are further divided in five categories: Cluster analysis, Graph partitioning approaches, Mathematical programming methods, Heuristic and Metaheuristic algorithms, and Artificial intelligence methodologies. Other reasonings arise from these:

- Cluster analysis, that is further divided in the simpler array-based clustering (ROC), and in the more sophisticated hierarchical and non-hierarchical clustering (similarity coefficients), is the simpler method to apply among the Production based methods. The main drawback is that it only takes as input the part/machine incidence matrix, too little information with respect to the real-life industrial environment.
- Graph partitioning approach is a more complete technique with respect to Cluster analysis. In fact, as already seen in the example [89], a part from the part/machine incidence matrix it can accept more data like the available number of individual machines of each type, the machine processing capabilities and the part-processing usage for each machine, and it can refine the result in terms of minimum intercell movements by means of an heuristic. However, the still limited amount of data demands for the usage of more complex techniques.

This further refinement leads to the core of the discussion: the search of the most promising method among Mathematical programming methods, Heuristic and Metaheuristic algorithms, and Artificial intelligence methodologies. With the help of the examples already presented, the scope of this section is to gather some key differences among the different methods. Table 3.2 enlists all the example papers used, and Table 3.3, that is largely inspired on a table presented by [6], compares the different methods based on main aspects.

No.	Ref. Source	No.	Ref.Source
1.	Defersha and Chen (2006)	11.	Defersha et al. (2017)
2.	Mahdavi et al. (2010)	12.	Arkat et al. (2012)
3.	Mahdavi et al. (2012)	13.	Jawahar and Subhaa (2017)
4.	Brown (2015)	14.	Sayadi et al. (2013)
5.	Mahadavi et al. (2008)	15.	Feng et al. (2017)
6.	Liu et al. (2010)	16.	Mak et al. (2007)
7.	Lei and Wu (2005)	17.	Li et al. (2010)
8.	Chung et al. (2011)	18.	Pandian and Mahapatra (2009)
9.	Chang et al. (2013)	19.	Yang and Yang (2008)
10.	Arkat et al. (2007)	20.	Boulif and Atif (2008)

Table 3.2: List of used example papers

By means of Table 3.3 some first considerations about the performances can be done:

- Mathematical Programming can manage only small problems, in the order of a few dozen among machines and part types, and the solution (optimal or sub-optimal) is found after many hours. Instead, all the other methods comprehending heuristics, metaheuristics and artificial intelligence approaches can face larger problems and solve them in less than 1h.
- The optimal solution is found only in few papers. In fact, especially for heuristics and metaheuristics, what is obtained is always a sub-optimal solution. Looking at the theory, among the metaheuristics the Global methods are the ones with a better chance to find a lower minimum with respect to Local methods thanks to their ability to probe the entire search space. Instead, Mathematical programming, due to the fact is based on a mathematical model, it is more inclined in finding the optimal solution. However, the check symbol (✓) on the OS column was added on the sources that presented the joint keyword "optimal solution" many times, even without verifying if it really was the optimal solution or not. Given these considerations, the data presented in the OS column are not reliable.
- Most of the authors compare their method with other existing methodologies on a base of reference problems found in literature. All the presented methods performed as or better compared to the older ones. Doing this kind of

Table 3.3: Comparison of example papers

No.	Objectives			Constraints					Method used									Performances				Tool
	Obj1	Obj2	Obj3	C1	C2	C3	C4	C5	MP	H	TS	SA	GA	PSO	ACO	ANN	FL	DS	CT	OS	Comp	
1.	✓			✓	✓	✓	✓	✓	✓									(10 x 25)			N	Lingo
2.	✓			✓	✓	✓		✓	✓									(3 x 6)	16h 20m	✓	N	Lingo 8.0
3.	✓			✓	✓			✓	✓									(7 x 10)			Y	Lingo
4.	✓			✓	✓				✓									(9 x 10)			Y	Lingo
5.		✓					✓			✓								(25 x 40)			Y	
6.	✓				✓		✓		✓	✓								(41 x 27)	5m 26s		Y	Matlab 7
7.	✓		✓	✓	✓		✓	✓	✓		✓							(18 x ?)	9,7s		Y	C++ 6.0
8.	✓				✓		✓		✓		✓							(30 x 70)	8,5s	✓	Y	
9.		✓			✓		✓		✓		✓							(25 x 40)	2,1s	✓	Y	
10.		✓			✓		✓					✓						(17x30)	7m 26s		Y	
11.	✓			✓	✓		✓	✓	✓			✓						(192x600)			Y	C++
12.	✓				✓		✓		✓				✓					(20x40)		✓	Y	C#
13.	✓				✓		✓		✓				✓					(20x25)	16,8s	✓	Y	MatlabR2010
14.			✓		✓				✓					✓				(20x51)			Y	
15.	✓			✓	✓		✓	✓	✓					✓				(30x40)	23m 41s		Y	
16.		✓		✓			✓		✓	✓					✓			(12x10)	1s		Y	C++
17.			✓				✓		✓						✓			(50x150)	20s		Y	
18.			✓				✓									✓		(35x90)	1,9s		Y	C++
19.			✓				✓									✓		(46x105)		✓	Y	
20.	✓	✓		✓	✓	✓	✓		✓	✓			✓				✓	(20x24)	13m 38s		Y	C++

Notes:
(I) Major objectives (column two).
Obj1 = minimum costs.
Obj2 = minimum intracell and/or intercell movement.
Obj3 = maximum grouping efficacy (minimum number of exceptional elements, minimum number of voids).
(II) Major constraints considered by different methods (column three).
C1 = Considering capacity of machines/workers in each cell.
C2 = Considering allowed number of machines/workers in each cell.
C3 = Considering multi-period planning.
C4 = Considering operation sequences.
C5 = Considering workload balancing among cells.
(III) Method used (column four).
MP = Mathematical Programming   H = Heuristic.
TS = Tabu Search   SA = Simulated Annealing.
GA = Genetic Algorithm   PSO = Particle Swarm Optimization.
ACO = Ant Colony Optimization   ANN = Artificial Neural Networks   FL = Fuzzy Logic.
(IV) Performances (column five).
DS = Largest data set used (machines x parts)   CT = Computational time for the previous data set.
OS = provides optimal solution when largest data set used   Comp = is compared to other existing methodologies (Y = Yes, N = No).

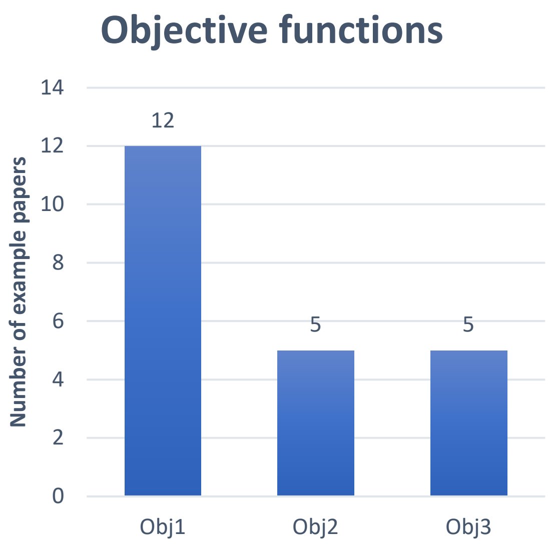
comparison is a good way to validate its own method and to track the evolution of CF methodologies towards the optimal solution.

Instead looking at the first three columns (Objectives, Constraints, Method used) is possible to assert that:

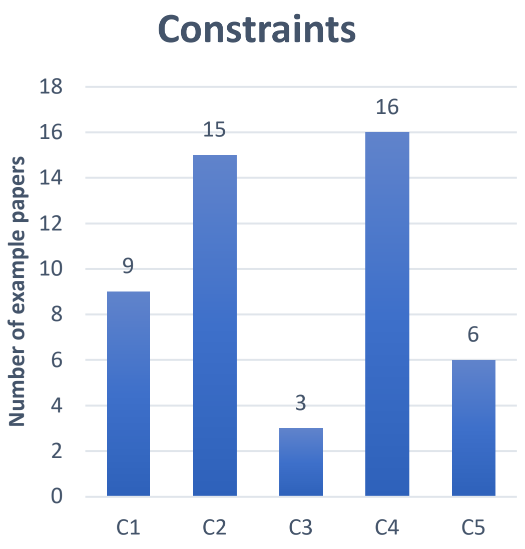
- There is a grand variety of models, that can be mono-objective, bi-objective, multi-objective, and considering from one to many constraints. Obviously, the higher is the number of objectives and the number of constraints, the more complete and reliable but also more complex is the model.
- The most frequent *objective function* is the minimum costs (Obj1). This cost could be a single term, like material handling cost, or the sum cost of several terms like in the examples [23,26]. Obviously, costs are often the leading factor in the design process of many manufacturing fields, and they must be minimized to save money. The other two objectives, that are the minimization of intracell and/or intercell movements (Obj2), and the maximization of grouping efficacy (Obj3) are less frequent. Obj2 and Obj3 are both related to layout and scheduling issues and, in an implicit way, to costs. Frequencies of appearance of Obj1, Obj2 and Obj3 in all the methods can be seen in Figure 3.20a.

  The most frequent *major constraints* are the ones considering allowed number of machines/workers in each cell (C2) and the ones considering operation sequences (C4). These two kinds of constraints can be seen as the bare minimum to formulate a comprehending all the machines or the presence of more than one machine type on a cell. Then, the other two types of constraints that consider the capacity of machines/workers (C1) and the workload balancing among the cells (C5) are less frequent and both can be seen as an improvement of C2 and C4, and thus they are more complex to model. Only few methods consider constraints about multi-period planning (C3). This means that most of the solutions found are static solutions, which don’t consider the dynamic environment that the Manufacturing sector is. Thus, the configurations found would be not enough flexible to accept for example a changing product mix or new products. Frequencies of appearance of C1, C2, C3, C4, C5 in all the methods can be seen in Figure 3.20b.

- Most of the papers start their explanation by introducing a mathematical model (MP checks are put on the methods with a mathematical model) that, as it is known, is made by the formulation of objective function and constraints. In fact, a mathematical model is a schematic and effective way to properly define the problem. The mathematical model is used as the foundation to properly build a method that for example could be a heuristic or metaheuristic method. In case of ANN, the two presented papers exploiting ART1 technique don't introduce a mathematical model because ANN is based on a comparison algorithm that categorizes the input problem in classes to find the optimal solution depicted by the block diagonal matrix.



(a) Frequency of Obj. Fun. in ex. papers



(b) Frequency of constraints in ex. papers

Figure 3.20: Frequencies of a) Obj. fun. and b) Constraints in the ex. papers

At this point of the discussion is possible to state that, even if Mathematical Programming is the only one theoretically able to find the optimal solution to the CF problem, it underperforms from a computational point of view with respect to the other techniques, which are heuristics, metaheuristics, and AI methodologies.

However, both the theoretical introduction and the analysis above don't give any hint about the best method to adopt. As noticed by both this research and the review paper [83], few research papers provide a clear reason in choosing one method with respect to another one. Following the article cited above, due to historical reasons the most used algorithm is the GA, one of the first metaheuristics to be adopted for the CF problem, followed by SA thanks to its simplicity of implementation. Another variable to add is the utilization of hybrid algorithms. Obviously, hybrid techniques combine the best characteristics of two methods to form a more performant and effective algorithm.

In conclusion, the most promising methods are:

1. *Metaheuristics*, which range from simpler algorithms like TS and SA to more complex methods like GA, PSO, and ACO. Heuristics can be seen as simple codes used to help larger algorithms for example in scheduling, like defining the Gantt chart. However, Metaheuristics, especially the ones deriving from biology like PSO and ACO, suffer for a more complicated definition which requires a preliminary study of the algorithms themselves. In particular PSO, that can be further divided in Fireflies, Bat, Bees, etc. algorithms, it is a forest of methods from which the designer can choose. Among the Metaheuristics, global methods are the most preferable, especially GA that seems to be the more immediate to understand.
2. *AI methodologies*, composed by ANN and Fuzzy Logic. These two methods have the peculiarity to mimic the cognitive functions of animals' brain. Among all the ANN techniques only ART1 was analyzed. What is striking about it is the absence of a Mathematical model thanks to its different approach to the problem, based on comparing the input data with a list of training data registered in the long term memory. Thus, further studies on ANN must be pursued. About Fuzzy Logic, it introduces in a Mathematical model fuzzy, uncertain data. In this way, the solution found can be a more

flexible and realistic solution that considers all the uncertainties coming from inside and outside the firm.

3. *Hybrid algorithms*, which combine together two different methods. In this way, more articulated and useful algorithms can be employed. The most combined methods are GA and FL [83]. However, the high variety of combinations don't permit a further investigation on this topic.

In the next chapter, a discussion about the feasibility of the solutions found with these methods will be addressed. In fact, up to now the implementation of manufacturing cells on the field was not considered as well as the presence of workers (human factor) inside the cell.

4. Theory vs reality

In this chapter the issue about the feasibility of the solutions given by the CF methods is discussed. In fact, as it is possible to infer from the previous analysis in chapter 3, most of the results are based on methodologies characterized by a single objective function, by a mono-period approach (constraint C3 is the least considered) and by a lack of other constraints (like C1 and C5), making the reader suspect a lack of completeness and realism of these methods.

Anticipating the content of the chapter, the three steps composing the CF problem (cell formation, cell layout, cell scheduling) are better explained; then, further topics quite recurrent in the methodologies like human factor, multi-period and multi-objective are presented and added to the discussion. Finally, some important conclusions are made based on presented papers and literature.

4.1 CF problem criteria

CF problem is composed by three major steps:

1) Cell formation: starting from manufacturing data like the Part/machine incidence matrix, machine cells are formed by grouping together machines and by constitute part families. Even if this seems enough to solve the CF problem, in reality there are other aspects to consider like layout and scheduling. In fact, by doing cell formation only, the only information gathered are the types of machines and part families composing cells, with no hint about the intercell and intracell fluxes of parts inside the department
2) Cell layout: by solving the cell formation, cells are formed without knowing the exact position of each machine inside the cell (intercell layout) and of each

cell inside the department (intracell layout). As already seen in the introductory part of the thesis (chapter 1), there are several possible intercell layouts to choose from, like linear single-row layout, linear double-row layout, u-shaped layout, and each of these requires the exact sequence of machines, from the input to the output of the cell. About intracell layout there could be many goals like creating rectangular, non-overlapping cells [7,41], that must be mostly independent (minimum intercell movements) and balanced on their workload. Thus, by solving the cell layout step the designer will place machines and cells in the optimal positions and he will be more aware about the part flows inside the facility and, as a consequence, more reliable decisions can be taken about for example the material handling system to adopt or the ubication of possible warehouses. Even in this case, the solution found is not complete, there are no hints about job scheduling and lead time for each part type.

3) Cell scheduling: the last but not least aspect to formulate a more complete solution is scheduling. Without scheduling, the firm would manufacture parts "at random", with a high probability of delivering the final product to the customer too late (after the deadline), due to a non-optimal exploitation of the available time, causing delay cost (legal costs set by the job's contract) and, in the worst scenario, loss of the customer. It is quite understandable that even a perfect cellular layout, able to minimize all the possible costs, which is not able to satisfy customer's demand is useless. Scheduling must not be confused with production planning: while scheduling is a tactic decision managed every day, where the main focus is which part to produce in which cell, production planning is a strategic decision managed every week or month, with the purpose of planning how many parts to produce for a customer and how much raw material to purchase.

Besides cell formation, cell layout, and cell scheduling, other aspects to formulate an effective cell configuration can be studied. To keep the analysis simple enough, only other three criteria are taken into account, and they are human factor, multi-period planning and multi-objective:

- Human factor: humans and machines work together to build the final product. Thus, it is of crucial importance to assign workers to machine cells based for example on their skill level, while assuring a healthy and safe environment in the workplace.
- Multi-period planning: to formulate a reliable and flexible configuration, a multi-period planning is necessary. In fact, by means of multi-period planning, an "average" configuration that can accept small changes between periods is found. Without it, a new CF problem and the related configuration must be formulated for each new period and theoretically for each new job, causing troubles in the case machines and/or material handling systems cannot be reconfigured in a simple way.
- Multi-objective: by taking a holistic point of view, the business' performance will be affected by the interaction of several terms like humans, machines, parts, cells, material handling systems, and so on. These interactions would be hard to quantify in terms of importance and related costs and benefits (economies of scale). Thus, an algorithm that considers Pareto-optimality has better chances to provide a more complete and reliable solution, due to the fact it takes into account various aspects of the manufacturing world. Among the possible objective functions, the ones already presented, and the most common, are minimum costs, minimum intracell/intercell movement, and maximum grouping efficacy. Other less common objective functions faced by the sources are minimum intercell/intracell workload unbalancing and minimum makespan.

These other three aspect were added because they seem to be quite recurrent among the CF methods. In the next paragraphs, these six criteria will be further discussed, and by means of the method papers presented in the literature review (chapter 2) and of the remaining literature about the topic conclusions will be stated.

4.2 Cell formation, cell layout and cell scheduling

The first three criteria, which form the three main steps of CF problem, are analyzed together. It is quite obvious to state that the more steps are addressed, the more complete and reliable will be the configuration found. Thus, integrated methods (methods that consider more than one step) are preferable with respect to simpler CF models that address cell formation only.

By means of method papers it is possible to construct a table to get an idea about the level of integration of these three steps in the models. To search the presence or not of these steps in the papers, the keywords *"formation"*, *"layout"*, and *"scheduling"* are checked.

Model type	Total number of papers	%
F	47	69
F+L	10	15
F+S	7	10
F+L+S	3	4
L	1	2
Sum	68	100

Table 4.1: Level of integration of method papers.

As it is possible to see from

Model type	Total number of papers	%
F	47	69
F+L	10	15
F+S	7	10
F+L+S	3	4
L	1	2
Sum	68	100

Table 4.1, most of the proposed methods consider cell formation only (F means cell formation), followed by integration models of cell formation and cell layout (F+L), cell formation and scheduling (F+S), and finally the complete integration of the three steps (F+L+S) is addressed by three method papers only. Only one method paper addresses cell layout only (L) [36] because it is about a layout improvement method of an already existing manufacturing cell.

Due to the relative low number of papers considered, other sources coming from the literature are taken into account. Many method papers and review papers [6–

9,38,50,83,100] confirm the hypothesis already supposed: in the literature there is a greater attention towards cell formation, which is the first, obligatory step to take in order to formulate a possible manufacturing cell configuration; still there is a lack in the literature for what concerns integration procedures of two or three steps.

A common procedure to cope with this lack of completeness is to apply other algorithms after the completion of the cell formation, in order to find the proper cell layout and cell scheduling [8]. In this case is not possible to speak of integration, because layout and scheduling steps are done successively and are based on an incomplete cell formation algorithm.

4.3 Human factor

Human factor is a crucial aspect in MC, especially in case of labor-intensive MC, where the presence of humans takes an important role in manufacturing parts, as opposed to capital intensive MC, where the main contribution is done by fixed assets like machines. In fact, the performance of MC improves over time, from its implementation to its full capacity, by solving both technical issues (i.e., allocating parts and machines to cells) and human issues [5,16]. The three major human issues are communication, teamwork and training [15], indeed it is quite understandable that the major role in a CM is not assigned to the single individual but to the whole team: the group's members must cooperate among them (teamwork), dividing the work based on a skill level continuously improving (training), resolving conflicts using a clear communication (communication). As it is possible to realize, human factor is a complex topic, that considers qualitative/non-quantifiable aspects of the human behavior that evolve over time.

Even in this case, method papers are filtered by means of keywords like "*worker*" and "*human*" to find methods that account for human factor aspect. A pie chart like the one in Figure 4.1 is constructed. Papers that contain workers' issues fall in the category "Yes", otherwise in the "No" category.

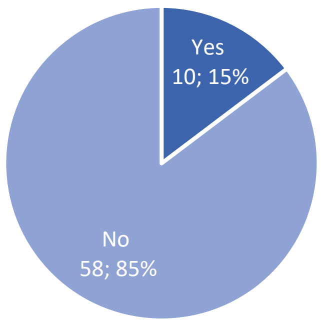


Figure 4.1: Presence of human factor among method papers.

Some integration methods that combine together cell formation with worker assignment exist and they are based only on one or more quantifiable measures like skill of workers, number of workers, available time of a single worker, related costs (salary, hiring, firing, training costs) [18,24,26,28,38,66,68,71,78,80,81,101]. Talking about human skill, it means the capability of a worker to do a certain task on a specific machine, thus the skillful operator is the one able to perform the higher number of tasks inside the cell. In a more sophisticated model, human skill accounts also for the processing time of a worker to do a specific job, composed by mean and variance [18], and so the most skillful operator is the one able to complete a certain task in the shortest time possible (following a stochastic distribution). The common path to deal with human factor is by means of algorithms subsequent at the cell formation, meaning at cells already formed. Workers are assigned to cells based on their skill level, and training programs are scheduled to follow the technological evolution in Manufacturing [19,102]. Apart from skill level, human relationships among team members (team leader and other workers) must be monitored to solve possible conflicts, as well as controlling the compliance with work standards in order to assure a minimum quality of production [17]. For what concerns Occupational Safety and Health (OSH) issues, each country has its own set of standards established by public organizations, like could be the Occupational Safety and Health Administration (OSHA) in the United States, to which are also added other standards provided by the International Labour Organization (ILO).

4.4 Multi-period

Historically, designers opted a static approach to cell formation [6,13,71], meaning that cells once designed need not be redesigned for a considerable length of time. However, in the last decade, it can be appreciated an increasing interest about Reconfigurable Manufacturing Systems (RMS) [14,83], comprehending also CM systems (CMS). An RMS is a dynamic manufacturing system that can quickly change and respond to the variable demand by changing its production capacity and capability. In case of CMS, in response to a change in the part families demand, the system must be able to rapidly move machines and workers from one cell to another one, purchase new machines, hire new workers etc.

It is of great interest to couple multi-period criteria with the three steps of CF problem and show the trend of proposed method papers over the years like in Figure 4.2:

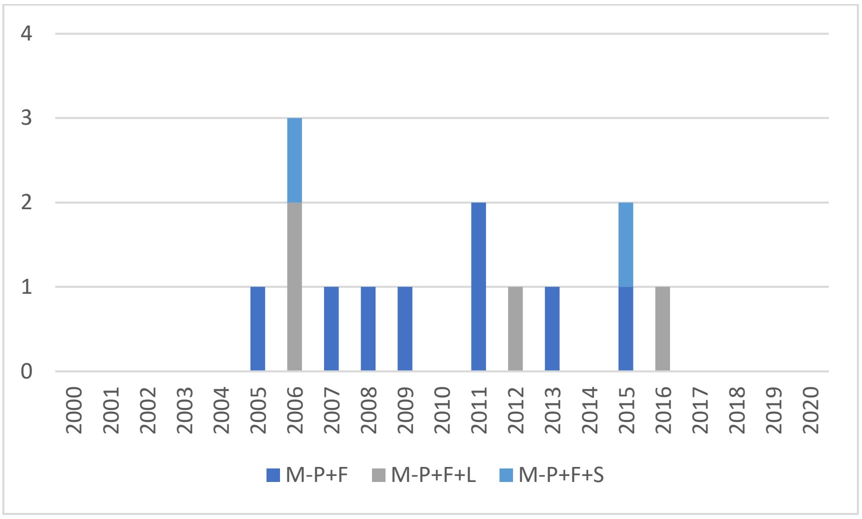


Figure 4.2: Integration of multi-period and CF problem steps among method papers.

The integration of multi period with cell formation (M-P+F) is the most searched, with 8 papers, followed by the integration with layout (M-P+L) with 4 entries and by integration with scheduling (M-P+S) with only 2 entries.

4.5 Multi-objective criteria

Multi-objective feature can be seen as adding further dimensions to the stated problem: while the other aspects refine the solution in a vertical way, concentrating on only one objective, adopting a multi-objective approach helps in expanding the view about CF problem in a horizontal way. Thus, by means of a multi-objective method is possible to better account reality, that is composed by a series of relationships between different elements, more or less deterministic. In fact, the optimal solutions, that can be more than one for the problem, follow the Pareto-optimality, which in practice states that the optimal solutions are a trade-off among the objective functions.

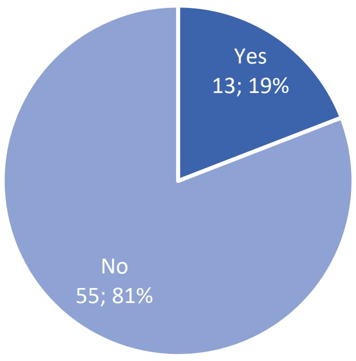


Figure 4.3: Presence of multi-objective among method papers.

The majority of proposed method papers are mono-objective. The bi-objective and multi-objective papers are only 13 out of 68.

Multi-objective algorithms are more complex to formulate, and this is why historically researchers are more inclined in introducing single objective methods, both in static and dynamic (multi-period) configurations [6,71]. However, it can be seen an increasing interest towards multi-period methods for what concerned dynamic facility layout [84].

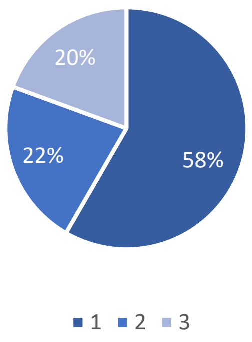
4.6 Final discussion about applicability of CF problem methods

From the analysis made in the previous sections, some important statements can be exposed:

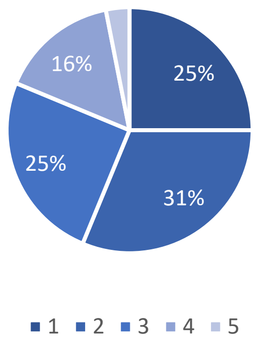
- The most searched aspect about CF problem is the cell formation step, followed by cell layout and cell scheduling topics. Integrating algorithms considering more than one step exist, but in a lower amount;
- Most of the methods are mono-period, mono-objective models. This would limit their applicability in the case of a dynamic environment like today's market.
- Human factor is a quite complex topic and it is ignored by the majority of studies, even if is quite evident the important contribution of the worker in CM performance.
- The common practice to enhance the solution is to use different methods at manufacturing cells already formed, this is the case of cell layout, cell scheduling and worker reassignment (human factor). In particular for human factor aspect, teamwork and good practices to improve the cell's performance can be pursued by means of Lean Production (LP) concepts [17,103].

However, from the sources analyzed is possible to state that a positive trend towards the integration of more aspects of the problem is in place, especially more importance is given to the multi-period aspect [14,83], which is crucial in the design of dynamic systems. An improvement of these methods can be appreciated starting from the last decade (2010-2019), thanks to the evolving knowledge about the topic and to the impressive improvement of computers in terms of hardware and software and the utilization of new architectures like quantum computing, which can be used for optimization problems [104].

To demonstrate the improvement over the years of CF methods, one point is assigned to each method paper every time one criteria among the six discussed is accounted to a total of six points (full integration). The comparison is done between method papers published in 2000-2009 and 2010-2019. The results are showed in Figure 4.4 and Table 4.2:



(a) 2000-2009



(b) 2010-2019

Figure 4.4: Improvement of methods complexity

Points	2000-2009	2010-2019
1	21	8
2	8	10
3	7	8
4	0	5
5	0	1

Sum	36	32

Table 4.2: Counter of points among method papers.

If in the past only cell formation was accounted, in recent years more and more integration attempts has been formulated.

5. Conclusions and future development

This thesis presented and discussed the CF problem and the main methodologies to solve it. The CF problem consists of three main steps:

- Cell formation, where machines and part families are chosen to form the manufacturing cells;
- Cell layout, in which machines are positioned in the optimal way inside the cell, and cells are distributed inside the department;
- Cell scheduling, where decisions about the sequence of part families to manufacture are taken.

After a quick introduction about the manufacturing cells, their characteristics and their advantages with respect to other manufacturing layouts, the core of the discussion is about the classification of CF methods and the general theory behind them.

The main classification of CF methodologies is:

- Visual methods;
- Part-coding analysis methods, like the Opitz code;
- Production based or Production flow analysis methods, further divided in:
    - Cluster analysis, array-based clustering (ROC), hierarchical clustering, and non-hierarchical clustering;
    - Graph partitioning approaches.
    - Mathematical programming methods.
    - Heuristic and Metaheuristic algorithms, the second one composed by local methods like TS and SA, and by global methods like GA, PSO, and ACO.
    - Artificial intelligence methodologies, formed by ANN and FL.

As the reader noticed, there is a grand variety of methods to adopt and for many of them, like Metaheuristics, it is not quite clear the reason to choose one among the others. Then, the methodologies are classified based on performance like computational time and data set dimension (machines x parts). From this comparison some important statements are highlighted:

- Visual methods and part-coding analysis methods are simple but unreliable because they don’t use important data like processing time, process routing, costs, layout constraints etc., and unfeasible for large problems:
- Cluster analysis and Graph partitioning approaches suffer of a lack of data in input, they mostly rely on the machine/part incidence matrix, thus they cannot provide complete solutions in terms of layout and scheduling;
- Mathematical programming underperforms the other methods in terms of computational time and size of data set managed;
- Methaeuristics, especially the global ones, AI approaches and Hybrid algorithms are the most prominent methodologies thanks to their better performances and the continuous research aiming at better integrate/complete solutions.

In the last part of the discussion, limits of the CF methods in the real world and possible improvements by means of other procedures are pointed out. In particular:

- There is still a big gap between what is requested by the market and what is offered by these methods, in fact most of them investigate cell formation only, are mono-period and mono-objective, resulting in partial and static solutions to a problem that is complex and dynamic. Moreover, human factor is quite ignored by most methods.
- The common practice to improve the solution is to apply further algorithms at cells already formed and follow good practices like the ones given by Lean Production.

However, there is a positive trend toward integration of more aspects of CF problem.

Possible future developments of this work could be:

- Increasing the number of analyzed papers, to have a wider consciousness about the topic and to rely less on review papers;
- Increasing the pool of presented methods, introducing for example *Scatter Search* (SS) and other ANN;
- Extending the analysis by adding other criteria to the ones already considered, like for example maintenance and production planning;
- Extending the discussion by better explaining how the manufacturing cells evolve once implemented, or how the material handling system is chosen and organized inside and between cells.

Bibliography

[1] Tompkins JA, Greasley A, Burke R, Vincoli JW. Industrial Plants. John Wiley & Sons, Inc; 2015.

[2] Shambu G, Suresh NC. Performance of hybrid cellular manufacturing systems: a computer simulation investigation. Eur J Oper Res 2000;120:436–58. https://doi.org/10.1016/S0377-2217(98)00371-3.

[3] Kher H V., Jensen JB. Shop performance implications of using cells, partial cells, and remainder cells. Decis Sci 2002;33:161–90. https://doi.org/10.1111/j.1540-5915.2002.tb01641.x.

[4] Chakravorty SS, Hales DN. Implications of cell design implementation: A case study and analysis. Eur J Oper Res 2004;152:602–14. https://doi.org/10.1016/S0377-2217(03)00060-2.

[5] Chakravorty SS, Hales DN. The evolution of manufacturing cells: An action research study. Eur J Oper Res 2008;188:153–68. https://doi.org/10.1016/j.ejor.2007.04.017.

[6] Papaioannou G, Wilson JM. The evolution of cell formation problem methodologies based on recent studies (1997-2008): Review and directions for future research. Eur J Oper Res 2010;206:509–21. https://doi.org/10.1016/j.ejor.2009.10.020.

[7] Arkat J, Farahani MH, Hosseini L. Integrating cell formation with cellular layout and operations scheduling. Int J Adv Manuf Technol 2012;61:637–47. https://doi.org/10.1007/s00170-011-3733-4.

[8] Neufeld JS, Gupta JND, Buscher U. A comprehensive review of flowshop group scheduling literature. Comput Oper Res 2016;70:56–74. https://doi.org/10.1016/j.cor.2015.12.006.

[9] Rafiei H, Rabbani M, Gholizadeh H, Dashti H. A novel hybrid SA/GA algorithm for solving an integrated cell formation–job scheduling problem with sequence-dependent set-up times. Int J Manag Sci Eng Manag

2016;11:134–42. https://doi.org/10.1080/17509653.2014.1003109.

[10] Lei D, Wu Z. Tabu search-based approach to multi-objective machine-part cell formation. Int J Prod Res 2005;43:5241–52. https://doi.org/10.1080/00207540500216516.

[11] Ioannou G. Time-phased creation of hybrid manufacturing systems. Int J Prod Econ 2006;102:183–98. https://doi.org/10.1016/j.ijpe.2005.03.005.

[12] Yin Y, Yasuda K. Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. Int J Prod Econ 2006;101:329–52. https://doi.org/10.1016/j.ijpe.2005.01.014.

[13] Balakrishnan J, Cheng CH. Multi-period planning and uncertainty issues in cellular manufacturing: A review and future directions. Eur J Oper Res 2007;177:281–309. https://doi.org/10.1016/j.ejor.2005.08.027.

[14] Renna P, Ambrico M. Design and reconfiguration models for dynamic cellular manufacturing to handle market changes. Int J Comput Integr Manuf 2015;28:170–86. https://doi.org/10.1080/0951192X.2013.874590.

[15] Bidanda B, Ariyawongrat P, Needy KL, Norman BA, Tharmmaphornphilas W. Human related issues in manufacturing cell design, implementation, and operation: A review and survey. Comput Ind Eng 2005;48:507–23. https://doi.org/10.1016/j.cie.2003.03.002.

[16] Molleman E, Slomp J. The impact of team and work characteristics on team functioning. Hum Factors Ergon Manuf 2006;16:1–15. https://doi.org/10.1002/hfm.20041.

[17] Saurin TA, Marodin GA, Ribeiro JLD. A framework for assessing the use of lean production practices in manufacturing cells. Int J Prod Res 2011;49:3211–30. https://doi.org/10.1080/00207543.2010.482567.

[18] Egilmez G, Erenay B, Süer GA. Stochastic skill-based manpower allocation in a cellular manufacturing system. J Manuf Syst 2014;33:578–88. https://doi.org/10.1016/j.jmsy.2014.05.005.

[19] Hashemi-Petroodi SE, Dolgui A, Kovalev S, Kovalyov MY, Thevenin S. Workforce reconfiguration strategies in manufacturing systems: a state of the art. Int J Prod Res 2020;0:1–25. https://doi.org/10.1080/00207543.2020.1823028.

[20] Paydar MM, Saidi-Mehrabad M. Revised multi-choice goal programming for integrated supply chain design and dynamic virtual cell formation with fuzzy parameters. Int J Comput Integr Manuf 2015;28:251–65.

https://doi.org/10.1080/0951192X.2013.874596.

[21] Brown JR. A capacity constrained mathematical programming model for cellular manufacturing with exceptional elements. J Manuf Syst 2015;37:227–32. https://doi.org/10.1016/j.jmsy.2014.09.005.

[22] Albadawi Z, Bashir HA, Chen M. A mathematical approach for the formation of manufacturing cells. Comput Ind Eng 2005;48:3–21. https://doi.org/10.1016/j.cie.2004.06.008.

[23] Defersha FM, Chen M. A comprehensive mathematical model for the design of cellular manufacturing systems. Int J Prod Econ 2006;103:767–83. https://doi.org/10.1016/j.ijpe.2005.10.008.

[24] Aryanezhad MB, Deljoo V, Mirzapour Al-E-Hashem SMJ. Dynamic cell formation and the worker assignment problem: A new model. Int J Adv Manuf Technol 2009;41:329–42. https://doi.org/10.1007/s00170-008-1479-4.

[25] Safaei N, Tavakkoli-Moghaddam R. Integrated multi-period cell formation and subcontracting production planning in dynamic cellular manufacturing systems. Int J Prod Econ 2009;120:301–14. https://doi.org/10.1016/j.ijpe.2008.12.013.

[26] Mahdavi I, Aalaei A, Paydar MM, Solimanpur M. Designing a mathematical model for dynamic cellular manufacturing systems considering production planning and worker assignment. Comput Math with Appl 2010;60:1014–25. https://doi.org/10.1016/j.camwa.2010.03.044.

[27] Saxena LK, Jain PK. Dynamic cellular manufacturing systems design - A comprehensive model. Int J Adv Manuf Technol 2011;53:11–34. https://doi.org/10.1007/s00170-010-2842-9.

[28] Mahdavi I new mathematical model for integrating all incidence matrices in multi-dimensional cellular manufacturing system, Aalaei A, Paydar MM, Solimanpur M. A new mathematical model for integrating all incidence matrices in multi-dimensional cellular manufacturing system. J Manuf Syst 2012;31:214–23. https://doi.org/10.1016/j.jmsy.2011.07.007.

[29] Egilmez G, Süer G. The impact of risk on the integrated cellular design and control. Int J Prod Res 2014;52:1455–78. https://doi.org/10.1080/00207543.2013.844375.

[30] Mahdavi I, Shirazi B, Paydar MM. A flow matrix-based heuristic algorithm for cell formation and layout design in cellular manufacturing system. Int J Adv

Manuf Technol 2008;39:943–53. https://doi.org/10.1007/s00170-007-1274-7.

[31] Liu C, Yin Y, Yasuda K, Lian J. A heuristic algorithm for cell formation problems with consideration of multiple production factors. Int J Adv Manuf Technol 2010;46:1201–13. https://doi.org/10.1007/s00170-009-2170-0.

[32] Adenso-Díaz B, Lozano S, Racero J, Guerrero F. Machine cell formation in generalized group technology. Comput Ind Eng 2001;41:227–40. https://doi.org/10.1016/S0360-8352(01)00056-0.

[33] Chang CC, Wu TH, Wu CW. An efficient approach to determine cell formation, cell layout and intracellular machine sequence in cellular manufacturing systems. Comput Ind Eng 2013;66:438–50. https://doi.org/10.1016/j.cie.2013.07.009.

[34] Arkat J, Saidi M, Abbasi B. Applying simulated annealing to cellular manufacturing system design. Int J Adv Manuf Technol 2007;32:531–6. https://doi.org/10.1007/s00170-005-0358-5.

[35] Wu TH, Chang CC, Chung SH. A simulated annealing algorithm for manufacturing cell formation problems. Expert Syst Appl 2008;34:1609–17. https://doi.org/10.1016/j.eswa.2007.01.012.

[36] Ariafar S, Ismail N. An improved algorithm for layout design in cellular manufacturing systems. J Manuf Syst 2009;28:132–9. https://doi.org/10.1016/j.jmsy.2010.06.003.

[37] Kia R, Baboli A, Javadian N, Tavakkoli-Moghaddam R, Kazemi M, Khorrami J. Solving a group layout design model of a dynamic cellular manufacturing system with alternative process routings, lot splitting and flexible reconfiguration by simulated annealing. Comput Oper Res 2012;39:2642–58. https://doi.org/10.1016/j.cor.2012.01.012.

[38] Mehdizadeh E, Rahimi V. An integrated mathematical model for solving dynamic cell formation problem considering operator assignment and inter/intra cell layouts. Appl Soft Comput J 2016;42:325–41. https://doi.org/10.1016/j.asoc.2016.01.012.

[39] Defersha FM, Hodiya A. A mathematical model and a parallel multiple search path simulated annealing for an integrated distributed layout design and machine cell formation. J Manuf Syst 2017;43:195–212. https://doi.org/10.1016/j.jmsy.2017.04.001.

[40] Mak KL, Wong YS, Wang XX. Adaptive genetic algorithm for manufacturing

cell formation. Int J Adv Manuf Technol 2000;16:491–7. https://doi.org/10.1007/s001700070057.

[41] Arkat J, Farahani MH, Ahmadizar F. Multi-objective genetic algorithm for cell formation problem considering cellular layout and operations scheduling. Int J Comput Integr Manuf 2012;25:625–35. https://doi.org/10.1080/0951192X.2012.665182.

[42] Jawahar N, Subhaa R. An adjustable grouping genetic algorithm for the design of cellular manufacturing system integrating structural and operational parameters. J Manuf Syst 2017;44:115–42. https://doi.org/10.1016/j.jmsy.2017.04.017.

[43] Feng H, Xia T, Da W, Xi L, Pan E. Concurrent design of cell formation and scheduling with consideration of duplicate machines and alternative process routings. J Intell Manuf 2019;30:275–89. https://doi.org/10.1007/s10845-016-1245-7.

[44] Onwubolu GC, Mutingi M. Genetic algorithm approach to cellular manufacturing systems. Comput Ind Eng 2001;39:125–44. https://doi.org/10.1016/S0360-8352(00)00074-7.

[45] Dimopoulos C, Mort N. A hierarchical clustering methodology based on genetic programming for the solution of simple cell-formation problems. Int J Prod Res 2001;39:1–19.

[46] Dimopoulos C, Mort N. Evolving knowledge for the solution of clustering problems in cellular manufacturing. Int J Prod Res 2004;42:4119–33. https://doi.org/10.1080/00207540410001711845.

[47] Yasuda K, Hu L, Yin Y. A grouping genetic algorithm for the multi-objective cell formation problem. Int J Prod Res 2005;43:829–53. https://doi.org/10.1080/00207540512331311859.

[48] Chan FTS, Lau KW, Chan PLY, Choy KL. Two-stage approach for machine-part grouping and cell layout problems. Robot Comput Integr Manuf 2006;22:217–38. https://doi.org/10.1016/j.rcim.2005.04.002.

[49] Jeon G, Leep HR. Forming part families by using genetic algorithm and designing machine cells under demand changes. Comput Oper Res 2006;33:263–83. https://doi.org/10.1016/j.cor.2005.03.033.

[50] Wu X, Chu CH, Wang Y, Yue D. Genetic algorithms for integrating cell formation with machine layout and scheduling. Comput Ind Eng 2007;53:277–

89. https://doi.org/10.1016/j.cie.2007.06.021.

[51] Mahdavi I, Paydar MM, Solimanpur M, Heidarzade A. Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. Expert Syst Appl 2009;36:6598–604. https://doi.org/10.1016/j.eswa.2008.07.054.

[52] Andrés C, Lozano S. A particle swarm optimization algorithm for part-machine grouping. Robot Comput Integr Manuf 2006;22:468–74. https://doi.org/10.1016/j.rcim.2005.11.013.

[53] Rafiee K, Rabbani M, Rafiei H, Rahimi-Vahed A. A new approach towards integrated cell formation and inventory lot sizing in an unreliable cellular manufacturing system. Appl Math Model 2011;35:1810–9. https://doi.org/10.1016/j.apm.2010.10.011.

[54] Sayadi MK, Hafezalkotob A, Naini SGJ. Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation. J Manuf Syst 2013;32:78. https://doi.org/10.1016/j.jmsy.2012.06.004.

[55] Husseinzadeh Kashan A, Karimi B, Noktehdan A. A novel discrete particle swarm optimization algorithm for the manufacturing cell formation problem. Int J Adv Manuf Technol 2014;73:1543–56. https://doi.org/10.1007/s00170-014-5906-4.

[56] Mak KL, Peng P, Wang XX, Lau TL. An ant colony optimization algorithm for scheduling virtual cellular manufacturing systems. Int J Comput Integr Manuf 2007;20:524–37. https://doi.org/10.1080/09511920600596821.

[57] Spiliopoulos K, Sofianopoulou S. An efficient ant colony optimization system for the manufacturing cells formation problem. Int J Adv Manuf Technol 2008;36:589–97. https://doi.org/10.1007/s00170-006-0862-2.

[58] Solimanpur M, Saeedi S, Mahdavi I. Solving cell formation problem in cellular manufacturing using ant-colony-based optimization. Int J Adv Manuf Technol 2010;50:1135–44. https://doi.org/10.1007/s00170-010-2587-5.

[59] Li X, Baki MF, Aneja YP. An ant colony optimization metaheuristic for machinepart cell formation problems. Comput Oper Res 2010;37:2071–81. https://doi.org/10.1016/j.cor.2010.02.007.

[60] Lozano S, Canca D, Guerrero F, García JM. Machine grouping using sequence-based similarity coefficients and neural networks. Robot Comput Integr Manuf 2001;17:399–404. https://doi.org/10.1016/S0736-5845(01)00015-1.

[61] Saidi-Mehrabad M, Safaei N. A new model of dynamic cell formation by a

neural approach. Int J Adv Manuf Technol 2007;33:1001–9. https://doi.org/10.1007/s00170-006-0518-2.

[62] Yang MS, Yang JH. Machine-part cell formation in group technology using a modified ART1 method. Eur J Oper Res 2008;188:140–52. https://doi.org/10.1016/j.ejor.2007.03.047.

[63] Sudhakara Pandian R, Mahapatra SS. Manufacturing cell formation with production data using neural networks. Comput Ind Eng 2009;56:1340–7. https://doi.org/10.1016/j.cie.2008.08.003.

[64] Tavakkoli-Moghaddam R, Aryanezhad MB, Safaei N, Azaron A. Solving a dynamic cell formation problem using metaheuristics. Appl Math Comput 2005;170:761–80. https://doi.org/10.1016/j.amc.2004.12.021.

[65] Bayram H, Şahin R. A comprehensive mathematical model for dynamic cellular manufacturing system design and Linear Programming embedded hybrid solution techniques. Comput Ind Eng 2016;91:10–29. https://doi.org/10.1016/j.cie.2015.10.014.

[66] Azadeh A, Ravanbakhsh M, Rezaei-Malek M, Sheikhalishahi M, Taheri-Moghaddam A. Unique NSGA-II and MOPSO algorithms for improved dynamic cellular manufacturing systems considering human factors. Appl Math Model 2017;48:655–72. https://doi.org/10.1016/j.apm.2017.02.026.

[67] Chung SH, Wu TH, Chang CC. An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. Comput Ind Eng 2011;60:7–15. https://doi.org/10.1016/j.cie.2010.08.016.

[68] Zohrevand AM, Rafiei H, Zohrevand AH. Multi-objective dynamic cell formation problem: A stochastic programming approach. Comput Ind Eng 2016;98:323–32. https://doi.org/10.1016/j.cie.2016.03.026.

[69] Wu TH, Chang CC, Yeh JY. A hybrid heuristic algorithm adopting both Boltzmann function and mutation operator for manufacturing cell formation problems. Int J Prod Econ 2009;120:669–88. https://doi.org/10.1016/j.ijpe.2009.04.015.

[70] Wu TH, Chung SH, Chang CC. Hybrid simulated annealing algorithm with mutation operator to the cell formation problem with alternative process routings. Expert Syst Appl 2009;36:3652–61. https://doi.org/10.1016/j.eswa.2008.02.060.

[71] Niakan F, Baboli A, Moyaux T, Botta-Genoulaz V. A bi-objective model in sustainable dynamic cell formation problem with skill-based worker assignment. J Manuf Syst 2016;38:46–62. https://doi.org/10.1016/j.jmsy.2015.11.001.

[72] Gonçalves JF, Resende MGC. An evolutionary algorithm for manufacturing cell formation. Comput Ind Eng 2004;47:247–73. https://doi.org/10.1016/j.cie.2004.07.003.

[73] James TL, Brown EC, Keeling KB. A hybrid grouping genetic algorithm for the cell formation problem. Comput Oper Res 2007;34:2059–79. https://doi.org/10.1016/j.cor.2005.08.010.

[74] Paydar MM, Saidi-Mehrabad M. A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. Comput Oper Res 2013;40:980–90. https://doi.org/10.1016/j.cor.2012.10.016.

[75] Boulif M, Atif K. A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem. Comput Oper Res 2006;33:2219–45. https://doi.org/10.1016/j.cor.2005.02.005.

[76] Zeidi JR, Javadian N, Tavakkoli-Moghaddam R, Jolai F. A hybrid multi-objective approach based on the genetic algorithm and neural network to design an incremental cellular manufacturing system. Comput Ind Eng 2013;66:1004–14. https://doi.org/10.1016/j.cie.2013.08.015.

[77] Boulif M, Atif K. A new fuzzy genetic algorithm for the dynamic bi-objective cell formation problem considering passive and active strategies. Int J Approx Reason 2008;47:141–65. https://doi.org/10.1016/j.ijar.2007.03.003.

[78] Bootaki B, Mahdavi I, Paydar MM. A hybrid GA-AUGMECON method to solve a cubic cell formation problem considering different worker skills. Comput Ind Eng 2014;75:31–40. https://doi.org/10.1016/j.cie.2014.05.022.

[79] Caux C, Bruniaux R, Pierreval H. Cell formation with alternative process plans and machine capacity constraints: A new combined approach. Int J Prod Econ 2000;64:279–84. https://doi.org/10.1016/S0925-5273(99)00065-1.

[80] Feng H, Da W, Xi L, Pan E, Xia T. Solving the integrated cell formation and worker assignment problem using particle swarm optimization and linear programming. Comput Ind Eng 2017;110:126–37. https://doi.org/10.1016/j.cie.2017.05.038.

[81] Mahdavi I, Aalaei A, Paydar MM, Solimanpur M. Multi-objective cell

formation and production planning in dynamic virtual cellular manufacturing systems. Int J Prod Res 2011;49:6517–37. https://doi.org/10.1080/00207543.2010.524902.

[82] Kuo RJ, Chi SC, Teng PW. Generalized part family formation through fuzzy self-organizing feature map neural network. Comput Ind Eng 2001;40:79–100. https://doi.org/10.1016/S0360-8352(00)00073-5.

[83] Renzi C, Leali F, Cavazzuti M, Andrisano AO. A review on artificial intelligence applications to the optimal design of dedicated and reconfigurable manufacturing systems. Int J Adv Manuf Technol 2014;72:403–18. https://doi.org/10.1007/s00170-014-5674-1.

[84] Zhu T, Balakrishnan J, Cheng CH. Recent advances in dynamic facility layout research. INFOR 2018;56:428–56. https://doi.org/10.1080/03155986.2017.1363591.

[85] Manometal web site- Turned parts by Manometal - Stainless steel n.d. https://www.manometal.com/en/services/turning-parts (accessed April 20, 2021).

[86] Terzi S. Design and Management of Production Systems Slides 2019:Design of Production Systems – Manufacturing Cells.

[87] Zehtaban L, Elazhary O, Roller D. A framework for similarity recognition of CAD models. J Comput Des Eng 2016;3:274–85. https://doi.org/10.1016/j.jcde.2016.04.002.

[88] Galan R, Racero J, Eguia I, Garcia JM. A systematic approach for product families formation in Reconfigurable Manufacturing Systems. Robot Comput Integr Manuf 2007;23:489–502. https://doi.org/10.1016/j.rcim.2006.06.001.

[89] Selim HM. Manufacturing cell formation problem: A graph partitioning approach. Ind Manag Data Syst 2002;102:341–52. https://doi.org/10.1108/02635570210432046.

[90] Tolio T. Manufacturing systems engineering slides 2019:Stochastic programming Part 1.

[91] Dan Stefanoiu, Pierre Borne, Dumitru Popescu, Florin Gheorghe Filip, Abdelkader El Kamel FGF. Optimization in Engineering Sciences : Metaheuristic, Stochastic Methods and Decision Support. 1st ed. John Wiley & Sons, Incorporated; 2014.

[92] Pérez-Gosende P, Mula J, Díaz-Madroñero M. Overview of dynamic facility layout planning as a sustainability strategy. Sustain 2020;12:13–5.

https://doi.org/10.3390/su12198277.

[93] Metaheuristics - Wikipedia n.d. https://en.wikipedia.org/wiki/Metaheuristic (accessed September 23, 2021).

[94] Pierreval H, Caux C, Paris JL, Viguier F. Evolutionary approaches to the design and organization of manufacturing systems. Comput Ind Eng 2003;44:339–64. https://doi.org/10.1016/S0360-8352(02)00195-X.

[95] Chaudhry SS, Luo W. Application of genetic algorithms in production and operations management: A review. Int J Prod Res 2005;43:4083–101. https://doi.org/10.1080/00207540500143199.

[96] Chattopadhyay M, Sengupta S, Ghosh T, Dan PK, Mazumdar S. Neuro-genetic impact on cell formation methods of Cellular Manufacturing System design: A quantitative review and analysis. Comput Ind Eng 2013;64:256–72. https://doi.org/10.1016/j.cie.2012.09.016.

[97] Grossberg S. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. Neural Networks 2013;37:1–47. https://doi.org/10.1016/j.neunet.2012.09.017.

[98] Dutta Gupta S, VSS P. Matrix Supported Liquid Culture and Machine Vision Analysis of Regenerated Shoots of Gladiolus. Methods Mol Biol 2010;589:97–107. https://doi.org/10.1007/978-1-60327-114-1_10.

[99] Azadegan A, Porobic L, Ghazinoory S, Samouei P, Saman Kheirkhah A. Fuzzy logic in manufacturing: A review of literature and a specialized application. Int J Prod Econ 2011;132:258–70. https://doi.org/10.1016/j.ijpe.2011.04.018.

[100] Bagheri M, Bashiri M. A new mathematical model towards the integration of cell formation with operator assignment and inter-cell layout problems in a dynamic environment. Appl Math Model 2014;38:1237–54. https://doi.org/10.1016/j.apm.2013.08.026.

[101] Bhatnagar R, Saddikuti V. Models for cellular manufacturing systems design: Matching processing requirements and operator capabilities. J Oper Res Soc 2010;61:827–39. https://doi.org/10.1057/jors.2008.181.

[102] Delgoshaei A, Delgoshaei A, Ali A. Review evolution of cellular manufacturing system's approaches: Human resource planning method. J Proj Manag 2019;4:31–42. https://doi.org/10.5267/j.jpm.2018.7.001.

[103] Pattanaik LN, Sharma BP. Implementing lean manufacturing with cellular layout: A case study. Int J Adv Manuf Technol 2009;42:772–9.

https://doi.org/10.1007/s00170-008-1629-8.

[104] Ajagekar A, Humble T, You F. Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems. Comput Chem Eng 2020;132:106630. https://doi.org/10.1016/j.compchemeng.2019.106630.

A. Appendix A

List of employed method papers

				Human	Multi-	Multi-		
No.	Formation	Layout	Scheduling	f	per.	obj		
1	1		1				2007	ACO
2	1						2008	ACO
3	1						2010	ACO
4	1						2010	ACO
5	1						2001	ANN
6	1				1		2007	ANN
7	1						2008	ANN
8	1						2009	ANN
9	1						2000	GA
10	1					1	2001	GA
11	1						2001	GA
12	1						2004	GA
13	1					1	2005	GA
14	1	1					2006	GA
15	1		1		1		2006	GA
16	1	1	1				2007	GA
17	1						2009	GA
18	1	1	1			1	2012	GA
19	1	1	1				2012	GA
20	1		1				2017	GA
21	1		1				2019	GA
22	1	1					2008	HE
23	1						2010	HE
24	1						2001	hybrid FL+ANN
25	1			1	1	1	2011	hybrid FL+MP
26	1			1		1	2014	hybrid GA
27	1				1	1	2013	hybrid GA+ANN
28	1				1	1	2008	hybrid GA+FL
29	1						2004	hybrid GA+Local

								search
30	1						2007	hybrid GA+Local search
31	1						2013	hybrid GA+Local search
32	1						2006	hybrid GA+MP
33	1			1			2017	hybrid PSO+MP
34	1						2009	hybrid SA+GA
35	1						2009	hybrid SA+GA
36	1		1				2016	hybrid SA+GA
37	1			1	1	1	2016	hybrid SA+GA
38	1						2000	hybrid SA+MP
39	1						2011	hybrid TS+GA
40	1			1	1	1	2016	hybrid TS+GA
41	1						2005	MP
42	1	1			1		2006	MP
43	1	1			1		2006	MP
44	1			1	1		2009	MP
45	1				1		2009	MP
46	1			1	1		2010	MP
47	1				1		2011	MP
48	1		1				2014	MP
49	1				1	1	2015	MP
50	1		1		1		2015	MP
51	1						2015	MP
52	1			1			2012	MP
53	1			1	1	1	2017	multi GA, PSO
54	1	1			1		2016	multi GA, SA
55	1				1		2005	multi GA, SA, TS
56	1						2006	PSO
57	1				1		2011	PSO
58	1						2013	PSO
59	1						2014	PSO
60	1						2007	SA
61	1						2008	SA
62		1					2009	SA
63	1	1			1		2012	SA
64	1	1		1	1	1	2016	SA
65	1	1					2017	SA
66	1						2001	TS
67	1	1				1	2005	TS
68	1	1					2013	TS

List of Figures

List of Tables