A hybrid genetic algorithm for manufacturing cell formation[1]

José Fernando Gonçalves
Faculdade de Engenharia do Porto
Departamento de Engenharia Mecânica e Gestão Industrial
Rua Dr. Roberto Frias
4200-465 Porto, Portugal
jfgoncal@fe.up.pt

Mauricio G.C. Resende [2]
Internet and Network Systems Research
AT&T Labs Research
180 Park Avenue, Bldg. 103, Room C241
Florham Park, NJ 07932, USA
mgcr@research.att.com

Abstract

Cellular manufacturing emerged as a production strategy capable of solving the problems of complexity and long manufacturing lead times in batch production. The fundamental problem in cellular manufacturing is the formation of product families and machine cells. This paper presents a new approach for obtaining machine cells and product families. The approach combines a local search heuristic with a genetic algorithm. Computational experience with the algorithm on a set of group technology problems available in the literature is also presented. The approach produced solutions with a grouping efficacy that is at least as good as any results previously reported in literature and improved the grouping efficacy for 59 % of the problems.

Keywords: Cellular Manufacturing; Group Technology; Genetic Algorithms; Random Keys

---

[1] AT&T Labs Research Technical Report TD-5FE6RN, October 29, 2002.
[2] Corresponding author. Send comments and requests to mgcr@research.att.com.

1. **Introduction**

Cellular manufacturing emerged as a production strategy capable of solving the problem of complexity and long manufacturing lead times in batch production systems in the beginning of the 1960s. Burbidge (1979) defined group technology (GT) as an approach to the optimization of work in which the organizational production units are relatively independent groups, each responsible for the production of a given family of products.

The fundamental problem in cellular manufacturing is the formation of product families and machine cells. The objective of this product-machine grouping problem is to form perfect (i.e. disjoint) groups in which products do not have to move from one cell to the other for processing.

The most common algorithms for GT found in the literature can be classified into the following four method categories: array-based, clustering, mathematical programming-based, and graph theoretic.

Array-based clustering methods perform a series of column and row permutations to form product and machine cells simultaneously. King (1980) and later King and Nakornchai (1982) developed the earliest array-based methods. King and Nakornchai (1982), Chandrasekharan and Rajagopalan (1987), Khator and Irani (1987), and Kusiak and Chow (1987) proposed other algorithms. A comprehensive comparison of three array-based clustering techniques is given in Chu and Tsai (1990). The quality of the solution given by these methods depends on the initial configuration of the zero-one matrix.

McAuley (1972) and Carrie (1973) developed the first algorithms using clustering and similarity coefficients. Since then, Mosier and Taube (1985), Seifoddini (1989), Gupta and Seifoddini (1990), Khan et al. (2000), Yamada and Yin (2001), and Dimopoulos and Mort (2001) proposed hierarchical methods. These methods have the disadvantage of not forming product and machine cells simultaneously, so additional methods must be employed to complete the design of the system.

GRAFICS, developed by Srinivasan and Narendran (1991), and ZODIAC, which is a modular version of MacQueen's clustering method, developed by Chandrasekharan and Rajagopalan (1987), are examples of non-hierarchical methods. Miltenburg and Zhang (1991) present a comprehensive comparison of nine clustering methods where non-hierarchical methods outperform both array-based and hierarchical methods.

Mathematical programming methods treat the clustering problem as a mathematical programming optimization problem. Different objective models have been used. Kusiak (1987) suggested the *p*-median model for GT, where it minimizes the total sum of distances between each product/machine pair. Shtub (1989) modeled the grouping problem as a generalized assignment problem. Choobineh (1988) formulated an integer programming problem which first determines product families and then assigns product families to cells with an objective of minimizing costs. Co and Araar (1988) developed a three-stage procedure to form cells and solved an

assignment problem to assign jobs to machines. Gunasingh and Lashkari (1989) formulated an integer programming problem to group machines and products for cellular manufacturing systems. Srinivasan et al. (1990) modelled the problem as an assignment problem to obtain product and machine cells. Joines et al. (1996) developed an integer program that is solved using a genetic algorithm. Cheng et al. (1998) formulate the problem as a travelling salesman problem and solve the model using a genetic algorithm. Chen and Heragu (1999) present two stepwise decomposition approaches to solve large-scale industrial problems. Won (2000) presents a two-phase methodology based on an efficient *p*-median approach. Akturk and Turkcan (2000) propose an integrated algorithm that solves the machine/product grouping problem by simultaneously considering the within-cell layout problem. Plaquin and Pierreval (2000) propose an evolutionary algorithm for cell formation taking into account specific constraints. Zhao and Wu (2000) present a genetic algorithm for cell formation with multiple routes and objectives. Caux et al. (2000) address the cell formation problem with multiple process plans and capacity constraints using a simulated annealing approach. Onwubolu and Mutingi (2001) develop a genetic algorithm approach taking into account cell-load variation. Uddin and Shanker (2002) address a generalized grouping problem, where each part has more than one process route. The problem is formulated as an integer programming problem and a procedure based on a genetic algorithm is suggested as a solution methodology.

Rajagopalan and Batra (1975) were the first to use graph theory to solve the grouping problem. They developed a machine graph with as many vertices as the number of machines. Two vertices were connected by an edge if there were parts requiring processing on both the machines. Cliques obtained from the graph were used to determine machine cells. The limitation of this method is that machine cells and part families are not formed simultaneously. Kumar et al. (1986) solved a graph decomposition problem to determine machine cells and part families for a fixed number of groups and with bounds on cell size. Their algorithm for grouping in flexible manufacturing systems is also applicable in the context of GT.

Vannelli and Kumar (1986) developed graph theoretic models to determine machines to be duplicated so that a perfect block diagonal structure can, be obtained**.** Kumar and Vannelli (1987) developed a similar procedure for determining parts to be subcontracted in order to obtain a perfect block diagonal structure. These methods are found to depend on the initial pivot element choice.

Vohra et al. (1990) suggested a network-based approach to solve the grouping problem. They used a modified form of the Gomory-Hu algorithm to decompose the part-machine graph. Askin et al. (1991) proposed a Hamiltonian-path algorithm for the grouping problem. The algorithm heuristically solves the distance matrix for machines as a TSP and finds a Hamiltonian path that gives the rearranged rows in the block diagonal structure. The disadvantage of this approach is that actual machine groups are not evident from its solution. Lee and Garcia-Diaz (1993) transformed the cell formation problem into a network flow formulation and used a primal-dual algorithm developed by Bertsekas and Tseng (1988) to determine the machine cells. Other graph approaches include the heuristic graph partitioning approach of Askin and Chiu (1990) and the minimum spanning tree approach of Ng (1993, 1996).

Selim et al. (1998) provide a comprehensive mathematical formulation of the cell formation problem and present a methodology-based classification of prior research.

To the best of our knowledge, no algorithm to optimally solve the product-families and machine-cells problem has yet been proposed in the literature.

The objective of this paper is to present a procedure for obtaining manufacturing cells. The approach combines a genetic algorithm with a local search heuristic. The genetic algorithm is responsible for generating sets of machines cells. The local search heuristic is applied on the set of machines cells with the objective of constructing sets of machine/product groups and improving their quality.

In Section 2, we present the problem in terms of a block diagonalization problem. Some measures of grouping quality are discussed in Section 3. Section 4 presents the local search procedure and the genetic algorithm. The performance of the approach, on a set of 35 GT problems available in the literature, is shown in Section 5. In Section 6, concluding remarks are made.

2. The block diagonalization problem

In this paper, we attempt to solve the machine and product-grouping problem as a zero one block diagonalization problem (BDP), to minimize inter-cellular movement and maximize the utilization of the machines within a cell.

The machine component incidence matrix [**A**], which is a zero-one matrix of order $\boldsymbol{N_p \times N_m}$, is the usual input data to the BDP. Element $\mathbf{a}_{p,m} = 1$ indicates the visit of product p to machine m and $\mathbf{a}_{p,m} = 0$ indicates otherwise.

Figure 2 presents an example of the block diagonalization process of a 15×12 matrix (the zero values were replaced by spaces in order to make the figure more readable). In this case (see Figure 2a), there are 15 products (p = 1, 2, …, 15) to be produced in a set of 12 machines ($m = M_1, M_2, \ldots, M_{12}$). The objective of the diagonalization problem is to produce a matrix such as the one in Figure 2b.

As can be observed in Figure 1, the following four product/machine groups were formed:

Table 1 – Resulting product/machine groups for the example in Figure 1.

Cells	Machines	Products
1	M_3, M_6, M_8	3, 5, 7, 9
2	M_5, M_7, M_{10}, M_{12}	10, 14, 15
3	M_1, M_4, M_{11}	1, 4, 6, 12, 13
4	M_2, M_9	2, 8, 11

a) Initial Matrix – (one cell containing all the machines and products)

	Machine											
Product	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
1	1			1								
2		1							1			
3			1			1		1				
4	1			1							1	
5			1			1		1				
6	1			1							1	
7			1					1				
8		1							1			
9			1			1		1				
10					1					1		1
11		1							1			
12				1							1	
13	1										1	
14					1		1					1
15					1		1			1		1

Block-Diagonalization Process

b) Final Matrix

	Machine											
Product	M_4	M_8	M_3	M_5	M_7	M_{10}	M_{12}	M_1	M_4	M_{11}	M_2	M_9
3	1	1	1									
5	1	1	1									
7		1	1									
9	1	1	1									
10				1		1	1					
14				1	1		1					
15				1	1	1	1					
1								1	1			
4								1	1	1		
6								1	1	1		
12									1	1		
13								1		1		
2											1	1
8											1	1
11											1	1

Figure 1 - Block diagonalization example.

3. Measure of performance

Several measures of goodness of machine-product groups in cellular manufacturing have been proposed. Sarker and Mondal (1999) present a simulation study of the effects of several factors on the efficiency measures. Sarker (2001) introduces a new rule and presents a survey of existing measures. The *grouping efficiency* and *grouping efficacy* are two popular grouping measures because they are simple to implement and generates block diagonal matrices. Grouping efficiency was first proposed by Chandrasekharan and Rajagopolan (1986b). It incorporates both machine utilization and inter-cell movement and is defined as the weighted sum of two functions η_1 and η_2 :

$$\text{Grouping Efficiency} = \eta = \boldsymbol{q}\eta_1 + (\boldsymbol{1} - \boldsymbol{q})\eta_2$$

where

η_1 = ratio of the number of 1´s in the diagonal blocks to the total number of elements in the diagonal blocks of the final matrix;

η_2 = ratio of the number of 0´s in the off-diagonal blocks to the total number of elements in the off-diagonal blocks of the final matrix;

$\boldsymbol{q}$ = weight factor.

One drawback of grouping efficiency is the low discriminating capability (i.e. the ability to distinguish good quality grouping from bad). For example, a bad solution with many 1´s in the off-diagonal blocks often shows efficiency figures around 75%. When the matrix size increases, the effect of 1´s in the off-diagonal blocks becomes smaller, and in some cases, the effect of inter-cell moves is not reflected in grouping efficiency. To overcome the low discriminating power of grouping efficiency between well-structured and ill-structured incidence matrices, Kumar and Chandrasekharan (1990) proposed another measure, which they call *grouping efficacy*. Unlike grouping efficiency, grouping efficacy is not affected by the size of the matrix.

The grouping efficacy can be defined as

$$\text{Grouping Efficacy} = \mu = \frac{N_1 - N_1^{Out}}{N_1 + N_0^{In}}, \qquad (1)$$

where

$\boldsymbol{N}_1$ = total number of 1´s in matrix A;
$\boldsymbol{N}_1^{Out}$ = total number of 1´s outside the diagonal blocks;
$\boldsymbol{N}_0^{In}$ = total number of 0´s inside the diagonal blocks.

The closer the grouping efficacy is to 1, the better will be the grouping. The grouping efficacy for the matrices in Figures 1a (one group containing all the machines and products) and 1b are, respectively,

$$\boldsymbol{\mu}_a = \frac{39 - 0}{39 + 141} = 21.67\%$$

$$\boldsymbol{\mu}_b = \frac{39 - 0}{39 + 6} = 86.67\%$$

As expected, the matrix in Figure 1b has a much higher grouping efficacy than the one in Figure 1a.

We chose grouping efficacy as the measure of performance for the hybrid genetic algorithm proposed in this paper for several reasons. First, it is able to incorporate both the within-cell machine utilization and the inter-cell movement. It has a high capability to differentiate between well-structured and ill-structured matrices (high discriminating power). It generates block diagonal matrices which are attractive in practice. Finally, it does not require a weight factor.

4. The new approach

The approach presented in this paper combines a genetic algorithm with a local search heuristic. The genetic algorithm is used to generate sets of machine cells. The evolutionary process, embedded in the genetic algorithm, is responsible for improving the grouping quality of the sets of machine cells generated. The local search heuristic is applied to the sets of machines cells generated by the genetic algorithm. The objective of the heuristic is to construct a set of machine/product groups and improve it, if possible. The heuristic feeds back to the genetic algorithm the grouping efficacy of the set of machine/product groups it constructs.

The remainder of this section describes in detail the genetic algorithm and the local search heuristic.

4.1. Genetic algorithm

Genetic algorithms (GA) were introduced by Holland (1975) and have been applied in a number of fields, e.g. mathematics, engineering, biology, and social science (Goldberg, 1989). GAs are search algorithms based on the mechanics of natural selection and natural genetics. They combine the concept of survival of the fittest with structured, yet randomized, information exchange to form robust search algorithms.

The concept of genetic algorithms is based on the evolution process that occurs in natural biology. An initial population of possible solutions (referred to as *individuals* or *chromosomes*) is generated. Some individuals are selected to be parents to produce offspring via a *crossover operator*. All the individuals are then evaluated and selected based on Darwin's concept of survival of the fittest. The process of reproduction, evaluation, and selection is repeated until a termination criterion is reached. In addition, a *mutation operator* with certain probability is applied to the individuals to change their genetic makeup. The objective of this mutation process is to increase the diversity of the population and ensure an extensive search.

Each *iteration* (also referred to as *generation* or *family of solutions*) is made up of chromosomes. Each chromosome is in turn made up of individual *genes*. These genes are encodings of the design variables that are used to evaluate the function being optimized. In each iteration of the search process, the system has a fixed population of chromosomes that represent the current solutions to the problem. Figure

2 represents a pseudo-code for a standard genetic algorithm.

Genetic algorithm
{
    Generate initial population P_t
    Evaluate population P_t
    while stopping criteria not satisfied **repeat**
    {
        Select elements from P_t to put into P_{t+1}
        Crossover elements of P_t and put into P_{t+1}
        Mutate elements of P_t and put into P_{t+1}
        Evaluate new population P_{t+1}
        $P_t = P_{t+1}$
    }
}

Figure 2 – A standard genetic algorithm.

The GA calls a subroutine to compute the fitness value (the quality) for each chromosome in the population. This fitness value is the only feedback to the GA.

The genetic algorithm presented in this paper uses a random key alphabet $U(0,1)$ and an evolutionary strategy (see Figure 5) identical to the one proposed by Bean (1994). An important feature of random keys is that all offspring formed by crossover are feasible solutions. This is accomplished by moving much of the feasibility issue into the fitness evaluation procedure. If any random key vector can be interpreted as a feasible solution, then any crossover is feasible. Through the dynamics of the genetic algorithm, the system learns the relationship between random key vectors and solutions with good objective values.

As mentioned earlier, the fitness function used is grouping efficacy. The other important aspects of genetic algorithms: chromosomal representation and decoding, parent selection, crossover, and mutation will be discussed next

4.1.1. Chromosomal representation and decoding

A chromosome represents a solution to the problem and is encoded as a vector of random keys (random numbers). Each chromosome is made of m+1 genes where m is the number of machines:

$$\text{Chromosome} = (\, gene_1,\ gene_2,\ \ldots,\ gene_m,\ gene_{m+1}).$$

The m+1st gene is used to determine the number of machine cells and uses the following decoding expression

$$nCells = \lceil gene_{m+1} \times m \rceil,$$

where $\lceil x \rceil$ is the smallest integer larger than x. Genes 1 through m are used to determine the assignment of machines to machine cells and use the decoding expression

$$Cell_i = \lceil gene_i \times nCells \rceil, \; i = 1, \ldots, m.$$

Figure 3 presents an example of the decoding of a chromosome.

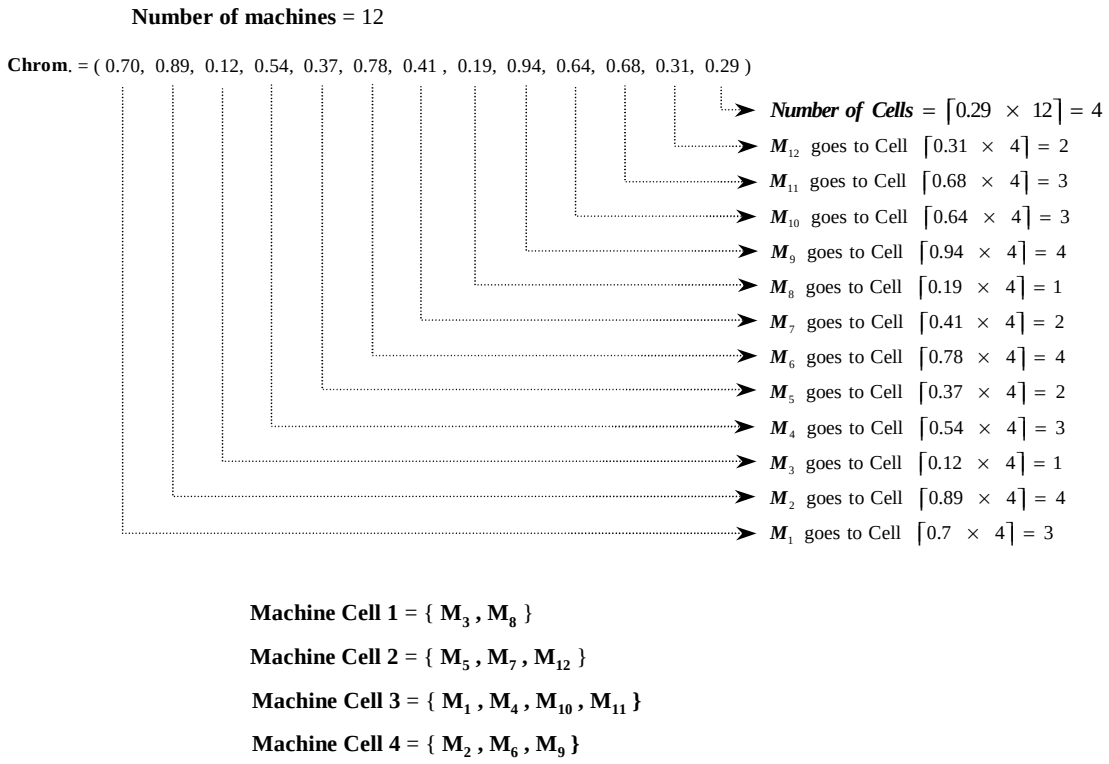


Figure 3 - Example of the decoding of a chromosome.

4.1.2. Reproduction, crossover, and mutation

Many variants of genetic algorithms are formed by varying the reproduction, crossover, and mutation operators. The reproduction and crossover operators determine which parents will have offspring, and how the genetic material is exchanged between the parents to create those offspring. Mutation allows for random alteration of genetic material. Reproduction and crossover operators tend to increase the quality of the populations and force convergence. Mutation opposes convergence and replaces genetic material lost during reproduction and crossover.

Reproduction is accomplished by copying the best individuals from one generation to the next, in what is often called an elitist strategy (Goldberg, 1989). The advantage of an elitist strategy over traditional probabilistic reproduction is that the best solution is monotonically improving from one generation to the next. The potential downside is population convergence. This can however be overcome by high mutation rates described below.

Parameterized uniform crossovers (Spears and DeJong, 1991) are employed in place of the traditional one-point or two-point crossover. After two parents are chosen randomly from the full old population (including chromosomes copied to the next generation in the elitist pass), at each gene a biased coin is tossed to select which parent will contribute the offspring. Figure 4 presents an example of the crossover operator. It assumes that a coin toss of heads selects the gene from the first parent, a tails chooses the gene from the second parent, and that the probability of tossing a heads is 0.7. Below is one potential crossover outcome:

Coin toss	***H***	***H***	***T***	***H***	***T***
Parent 1	***0.57***	***0.93***	***0.36***	***0.12***	***0.78***
Parent 2	***0.46***	***0.35***	***0.59***	***0.89***	***0.23***
Offspring	***0.57***	***0.93***	***0.59***	***0.12***	***0.23***

Figure 4 - Example of uniform crossover.

To prevent premature convergence of the population, at each generation one or more new members of the population are randomly generated from the same distribution as the original population. This process has the same effect as applying at each generation the traditional gene-by-gene mutation with small probability.

All chromosomes of the first generation are randomly generated. Figure 5 depicts the evolutionary process.

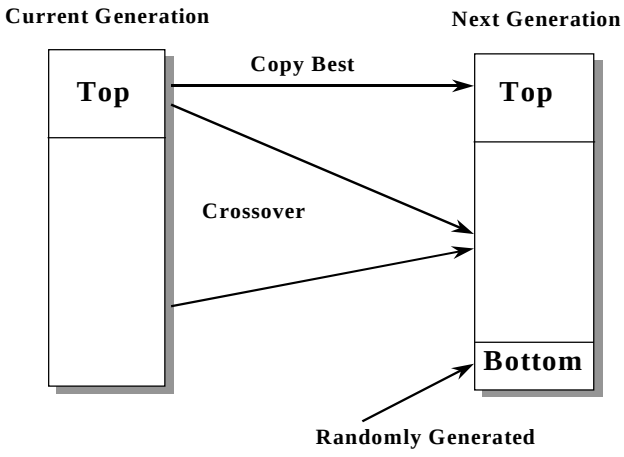


Figure 5 - Evolutionary process.

4.2 Local search heuristic

The local search heuristic is applied to the sets of machine cells generated by the genetic algorithm. When the machine cells are known, it is customary to assign a product to the cell where it visits the maximum number of machines. This is optimal to minimize inter-cell movement. However, it does not guarantee good utilization of the machines within a cell. To overcome this problem, a local search heuristic, which takes into consideration both inter-cell movement and machine utilization, is developed. The local search heuristic has proven, in simulations, to produce better assignments than the customary rule mentioned above (see Figure 8).

The local search heuristic consists of an improvement procedure that is repeatedly applied. Each iteration *k* of the procedure starts with a given initial set of machine cells $\boldsymbol{M}_k^{INITIAL}$, and produces a set of product families $\boldsymbol{P}_k^{FINAL}$, and a set of machine cells $\boldsymbol{M}_k^{FINAL}$. Two block-diagonal matrices can be obtained by combining $\boldsymbol{M}_k^{INITIAL}$ with $\boldsymbol{P}_k^{FINAL}$ and $\boldsymbol{M}_k^{FINAL}$ with $\boldsymbol{P}_k^{FINAL}$. From these two matrices, the one with the highest grouping efficacy is chosen as the resulting block-diagonal matrix of the iteration *k*. The procedure stops if $\boldsymbol{M}_k^{FINAL} = \boldsymbol{M}_k^{INITIAL}$ or if the grouping efficacy of the block-diagonal matrix resulting from iteration *k* is not greater than the grouping efficacy of the block-diagonal matrix resulting from the previous iteration *k*-1, (for *k>2*). Otherwise, the procedure sets $\boldsymbol{M}_{k+1}^{INITIAL} = \boldsymbol{M}_k^{FINAL}$ and continues to iteration *k*+1. Each iteration *k* of the local search heuristic consists of following two steps:

1) Assignment of products to the initial set of machine cells $\boldsymbol{M}_k^{INITIAL}$. (Note that the initial the set of machine cells of iteration 1, $\boldsymbol{M}_1^{INITIAL}$, is supplied by the genetic algorithm). Products are assigned to machine cells one at a time (in any order). A product is assigned to the cell that maximizes the grouping efficacy, that is, a product is assigned to the machine cell $\boldsymbol{C}^*$, given by

$$C^* = \arg\max_C \left\{ \frac{N_1 - N_{1,C}^{Out}}{N_1 + N_{0,C}^{In}} \right\},$$

where

$\boldsymbol{N}_1$ = total number of 1´s in matrix A;

$\boldsymbol{N}_1^{Out}$ = total number of 1´s outside the diagonal blocks if the product is assigned to cell $\boldsymbol{C}$;

$\boldsymbol{N}_0^{In}$ = total number of 0´s inside the diagonal blocks if the product is assigned to cell $\boldsymbol{C}$.

In this step, the heuristic generates a set of product families $\boldsymbol{P}_k^{FINAL}$. Let $\boldsymbol{\mu}_k^1$ be the efficacy of the block-diagonal matrix defined by $\boldsymbol{M}_k^{INITIAL}$ and $\boldsymbol{P}_k^{FINAL}$.

2) Assignment of machines to the set of product families $\boldsymbol{P_k^{FINAL}}$ obtained in step 1). Machines are assigned to product families, one at a time (in any order). A machine is assigned to the product family that maximises the grouping efficacy, that is, a machine is assigned to the product family $\boldsymbol{F^*}$, given by

$$F^* = \arg\max_F \left\{ \frac{N_1 - N_{1,F}^{Out}}{N_1 + N_{0,F}^{In}} \right\},$$

where

$\boldsymbol{N_1}$ = total number of 1´s in matrix A;

$\boldsymbol{N_1^{Out}}$ = total number of 1´s outside the diagonal blocks if the product is assigned to cell $\boldsymbol{F}$;

$\boldsymbol{N_0^{In}}$ = total number of 0´s inside the diagonal blocks if the product is assigned to cell $\boldsymbol{F}$.

In this step, the local search heuristic generates a new set of machine cells $\boldsymbol{M_k^{FINAL}}$. Let $\boldsymbol{\mu_k^2}$ be the efficacy of the block-diagonal matrix defined by $\boldsymbol{M_k^{FINAL}}$ and $\boldsymbol{P_k^{FINAL}}$.

The block-diagonal matrix resulting from the iteration has a grouping efficacy given by $\boldsymbol{\mu_k = \max\left(\mu_k^1, \mu_k^2\right)}$. If $\boldsymbol{M_k^{FINAL} = M_k^{INITIAL}}$ or $\boldsymbol{\mu_k \leq \mu_{k-1}}$ $\boldsymbol{(k \geq 2)}$, then the iterative process stops and the block-diagonal matrix of iteration *k*-1 is the result. Otherwise, the procedure sets $\boldsymbol{M_{k+1}^{INITIAL} = M_k^{FINAL}}$ and continues to step 1) of iteration *k*+1 .

4.2.1. An example

Suppose we start with the initial set of machine cells given by the genetic algorithm and shown in Table 2:

Table 2 – Initial set of machine cells.

Cells	Machines
1	M_3, M_8
2	M_5, M_7, M_{12}
3	M_1, M_4, M_{10}, M_{11}
4	M_2, M_6, M_9

Thus, $M_1^{INITIAL} = \{(M_3, M_8), (M_5, M_7, M_{12}), (M_1, M_4, M_{10}, M_{11}), (M_2, M_6, M_9)\}$.

Step 1 – Determining a set of product families

Table 3 presents the value of $\mu_C = \left\{\frac{N_1 - N_{1,C}^{Out}}{N_1 + N_{0,C}^{In}}\right\}$, for each product and each machine cell. A product is assigned to the cell with the highest value of the grouping efficacy (the cells in grey in Table 3).

Table 3 – Computations for step 1 of the local search heuristic.

		Machine Cells			
		(M_3, M_8)	**(M_5, M_7, M_{12})**	**(M_1, M_4, M_{10}, M_{11})**	**(M_2, M_6, M_9)**
Products	**Product Machines**	μ_C	μ_C	μ_C	μ_C
1	M_1, M_4	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	**(39-0)/(39+2) =95.1%**	(39-2)/(39+3) =88.1%
2	M_2, M_9	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	(39-2)/(39+4) =86.0%	**(39-0)/(39+1) =97.5%**
3	M_3, M_6, M_8	**(39-1)/(39+0) =97.4%**	(39-3)/(39+3) =85.7%	(39-3)/(39+4) =83.7%	(39-2)/(39+2) =90.2%
4	M_1, M_4, M_{11}	(39-3)/(39+2) =87.8%	(39-3)/(39+3) =85.7%	**(39-0)/(39+1) =97.5%**	(39-3)/(39+3) =85.7%
5	M_3, M_6, M_8	**(39-1)/(39+0) =97.4%**	(39-3)/(39+3) =85.7%	(39-3)/(39+4) =83.7%	(39-2)/(39+2) =90.2%
6	M_1, M_4, M_{11}	(39-3)/(39+2) =87.8%	(39-3)/(39+3) =85.7%	**(39-0)/(39+1) =97.5%**	(39-3)/(39+3) =85.7%
7	M_3, M_8	**(39-0)/(39+0) =100.0%**	(39-2)/(39+3) =88.1%	(39-2)/(39+4) =86.0%	(39-2)/(39+3) =88.1%
8	M_2, M_9	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	(39-2)/(39+4) =86.0%	**(39-0)/(39+1) =97.5%**
9	M_3, M_6, M_8	**(39-1)/(39+0) =97.4%**	(39-3)/(39+3) =85.7%	(39-3)/(39+4) =83.7%	(39-2)/(39+2) =90.2%
10	M_5, M_{10}, M_{12}	(39-3)/(39+2) =87.8%	**(39-1)/(39+1) =95.0%**	(39-2)/(39+3) =88.1%	(39-3)/(39+3) =85.7%
11	M_2, M_9	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	(39-2)/(39+4) =86.0%	**(39-0)/(39+1) =97.5%**
12	M_4, M_{11}	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	**(39-0)/(39+2) =95.1%**	(39-2)/(39+3) =88.1%
13	M_1, M_{11}	(39-2)/(39+2) =90.2%	(39-2)/(39+3) =88.1%	**(39-0)/(39+2) =95.1%**	(39-2)/(39+3) =88.1%
14	M_5, M_7, M_{12}	(39-3)/(39+2) =87.8%	**(39-0)/(39+0) =100.0%**	(39-3)/(39+4) =83.7%	(39-3)/(39+3) =85.7%
15	M_5, M_7, M_{10}, M_{12}	(39-4)/(39+2) =85.4%	**(39-1)/(39+0) =97.4%**	(39-3)/(39+3) =85.7%	(39-4)/(39+3) =83.3%

Thus, $P_1^{FINAL} = \{(3,5,7,9), (10,14,15), (1,4,6,12,13), (2,8,11)\}$.

Table 4 - Set of machine/product groups obtained after step 1.

Group	Machines	Products
1	M_3, M_8	3, 5, 7, 9
2	M_5, M_7, M_{12}	10, 14, 15
3	M_1, M_4, M_{10}, M_{11}	1, 4, 6, 12, 13
4	M_2, M_6, M_9	2, 8, 11

The resulting grouping combining $M_1^{INITIAL}$ and P_1^{FINAL} is given in Table 4 and the corresponding block-diagonal matrix is given in Figure 6.

Table 5 – Computations for step 2 of the local search heuristic.

		Product Families			
		(3, 5, 7, 9)	**(10, 14, 15)**	**(1, 4, 6, 12, 13)**	**(2, 8, 11)**
Machines	**Machine Products**	**μ_F**	**μ_F**	**μ_F**	**μ_F**
M_1	1, 4, 6, 13	(39-4)/(39+4) =81.4%	(39-4)/(39+3) =83.3%	**(39-0)/(39+1) =97.5%**	(39-4)/(39+3) =83.3%
M_2	2, 8, 11	(39-3)/(39+4) =83.7%	(39-3)/(39+3) =85.7%	(39-3)/(39+5) =81.8%	**(39-0)/(39+0) =100.0%**
M_3	3, 5, 7, 9	**(39-0)/(39+0) =100.0%**	(39-4)/(39+3) =83.3%	(39-4)/(39+5) =79.5%	(39-4)/(39+3) =83.3%
M_4	1, 4, 6, 12	(39-4)/(39+4) =81.4%	(39-4)/(39+3) =83.3%	**(39-0)/(39+1) =97.5%**	(39-4)/(39+3) =83.3%
M_5	10, 14, 15	(39-3)/(39+4) =83.7%	**(39-0)/(39+0) =100.0%**	(39-3)/(39+5) =81.8%	(39-3)/(39+3) =85.7%
M_6	3, 5, 9	**(39-0)/(39+1) =97.5%**	(39-3)/(39+3) =85.7%	(39-3)/(39+5) =81.8%	(39-3)/(39+3) =85.7%
M_7	14, 15	(39-2)/(39+3) =88.1%	**(39-0)/(39+1) =97.5%**	(39-2)/(39+5) =84.1%	(39-2)/(39+3) =88.1%
M_8	3, 5, 7, 9	**(39-0)/(39+0) =100.0%**	(39-4)/(39+3) =83.3%	(39-4)/(39+5) =79.5%	(39-4)/(39+3) =83.3%
M_9	2, 8, 11	(39-3)/(39+4) =83.7%	(39-3)/(39+3) =85.7%	(39-3)/(39+5) =81.8%	**(39-0)/(39+0) =100.0%**
M_{10}	10, 15	(39-2)/(39+4) =86.0%	**(39-0)/(39+1) =97.5%**	(39-2)/(39+5) =84.1%	(39-2)/(39+3) =88.1%
M_{11}	4, 6, 12, 13	(39-4)/(39+4) =81.4%	(39-4)/(39+3) =83.3%	**(39-0)/(39+1) =97.5%**	(39-4)/(39+3) =83.3%
M_{12}	10, 14, 15	(39-3)/(39+4) =83.7%	**(39-0)/(39+0) =100.0%**	(39-3)/(39+5) =81.8%	(39-3)/(39+3) =85.7%

	Machine											
Product	M_8	M_3	M_5	M_7	M_{12}	M_1	M_4	M_{10}	M_{11}	M_2	M_6	M_9
3	**1**	**1**									**1**	
5	**1**	**1**									**1**	
7	**1**	**1**										
9	**1**	**1**									**1**	
10			**1**		**1**			**1**				
14			**1**	**1**	**1**							
15			**1**	**1**	**1**			**1**				
1						**1**	**1**					
4						**1**	**1**		**1**			
6						**1**	**1**		**1**			
12							**1**		**1**			
13						**1**			**1**			
2										**1**		**1**
8										**1**		**1**
11										**1**		**1**

Figure 6 - Block diagonal matrix corresponding to the product/machine cells in Table 4.

The grouping efficacy after step 1 is

$$\mu_1^{\,1} = \frac{39 - 5}{39 + 12} = 66.67\%$$

Step 2 – Determining a set of machine cells

Table 5 presents the value of the grouping efficacy, $\mu_F = \left\{\frac{N_1 - N_{1,F}^{Out}}{N_1 + N_{0,F}^{In}}\right\}$, for each product and each machine cell. A machine is assigned to the product family with the highest grouping efficacy value (the cells in grey in Table 5).

Thus, $M_1^{FINAL} = \{(M_3, M_6, M_8), (M_5, M_7, M_{10}, M_{12}), (M_1, M_4, M_{11}), (M_2, M_9)\}$.

The resulting grouping combining P_1^{FINAL} and M_1^{FINAL} is given in Table 6.

Table 6 - Set of machine/product groups obtained after 2.

Group	Machines	Products
1	M_3, M_6, M_8	3, 5, 7, 9
2	M_5, M_7, M_{10}, M_{12}	10, 14, 15
3	M_1, M_4, M_{11}	1, 4, 6, 12, 13
4	M_2, M_9	2, 8, 11

The corresponding block-diagonal matrix is given in Figure 7.

	Machine											
Product	M_6	M_8	M_3	M_5	M_7	M_{10}	M_{12}	M_1	M_4	M_{11}	M_2	M_9
3	**1**	**1**	**1**									
5	**1**	**1**	**1**									
7		**1**	**1**									
9	**1**	**1**	**1**									
10				**1**		**1**	**1**					
14				**1**	**1**		**1**					
15				**1**	**1**	**1**	**1**					
1								**1**	**1**			
4								**1**	**1**	**1**		
6								**1**	**1**	**1**		
12									**1**	**1**		
13								**1**		**1**		
2											**1**	**1**
8											**1**	**1**
11											**1**	**1**

Figure 7 - Block diagonal matrix corresponding to the product/machine groups in Table 6.

The grouping efficacy after step 2 is

$$\mu_1^{\,2} = \frac{39 - 0}{39 + 6} = 86.67\%.$$

The resulting block-diagonal matrix obtained at the end of step 2 has a grouping efficacy of $\mu_1 = \max\left(\mu_1^1, \mu_1^2\right) = \max\left(66.67\%,\ 86.67\%\right) = 86.67\%$. Since the set of machine cells obtained at the end of this step is different from the initial set of machine cells and has greater grouping efficacy we set $\boldsymbol{M}_2^{INITIAL} = \boldsymbol{M}_1^{FINAL}$, i.e.

$$\boldsymbol{M}_2^{INITIAL} = \left\{(\boldsymbol{M}_3, \boldsymbol{M}_6, \boldsymbol{M}_8), (\boldsymbol{M}_5, \boldsymbol{M}_7, \boldsymbol{M}_{10}, \boldsymbol{M}_{12}), (\boldsymbol{M}_1, \boldsymbol{M}_4, \boldsymbol{M}_{11}), (\boldsymbol{M}_2, \boldsymbol{M}_9)\right\}$$

and proceed to iteration 2 to repeat steps 1 and 2. At the end of step 2 of the second iteration, we obtain a set of machine cells that is equal to the initial set (i.e. $\boldsymbol{M}_2^{INITIAL} = \boldsymbol{M}_2^{FINAL}$), and so we stop. The final block-diagonal matrix is the one shown in Figure 7 and has a grouping efficacy of 86.67%.

5. Computational results

To demonstrate the performance of the proposed algorithm, we tested the hybrid genetic algorithm on 35 GT instances collected from the literature. The selected matrices range from dimension 5×7 to 40×100 and comprise well-structured, as well as unstructured matrices. The matrix sizes and their sources are presented in Table 7. The smallest dimension of each matrix was considered to be the number of rows.

We compare the grouping efficacy obtained by the proposed algorithm with the grouping efficacies obtained by the following four approaches:

- ZODIAC (Chandrasekharan and Rajagopalan, 1987);
- GRAFICS (Srinivasan and Narendran, 1991);
- Clustering algorithm (Srinivasan, 1994);
- Genetic algorithm (Cheng et al., 1998).

These four approaches provide the best results, found in the literature, for the 35 problems used for comparison.

The grouping efficacy resulting from the application of ZODIAC to above problems is reported in Srinivasan and Narendran (1991), in Srinivasan (1994), and in Cheng et al. (1998). The paper by Srinivasan and Narendran (1991) includes five problems not included in Table 7. Two of these problems were from an unpublished master's thesis, and were unavailable. One of the instances was from Seifodini and Wolfe (1986), but the referenced source does not have any problem of the same dimension 12×12 and with the same number of 1´s. The remaining two other problems, with matrices of sizes 12×10 and 10×20 were from McAuley (1972) and Badarinarayna (1987). However, even after much effort to obtain these instances, we were unable to find them.

The test was run on a personal computer having a MS Windows Me PC with an AMD Thunderbird 1.333 GHz processor. The algorithm was coded in Visual Objects 2.0b-1 from CA-Computer Associates.

The present state-of-the-art practice on genetic algorithms does not provide information on how to configure them. Therefore, a small pilot study was conducted in order to obtain a reasonable configuration. The algorithm was configured as follows and the configuration was held constant for all problems. The number of chromosomes in the population equals 3 times the number of rows in the problem. The probability of tossing heads during crossover was made equal to 0.7. The elitist strategy copies to the next generation the top 20% of the previous population chromosomes. Mutation substitutes with randomly generated chromosomes the bottom 30% of the population chromosomes. The genetic algorithm stops after 150 generations.

The local search procedure presented in §4.2 can produce machine cells or product families that are empty or that have only one machine or product. We address

these cases by penalizing their grouping efficacy, i.e., we consider them to have a grouping efficacy of zero. By doing this we make sure that the evolutionary process of the genetic algorithm will remove the corresponding chromosomes from the population.

The test results are presented in Table 8. In Appendix A, we present the block-diagonal matrices, found by the hybrid genetic algorithm, for each of the 35 problems mentioned in Table 8.

Table 7 - Selected GT problems from the literature.

Num.	Source	Size
1	King and Nakornchai (1982)	5×7
2	Waghodekar and Sahu (1984)	5×7
3	Seiffodini (1989)	5×18
4	Kusiak (1992)	6×8
5	Kusiak and Chow (1987)	7×11
6	Boctor (1991)	7×11
7	Seiffodini and Wolfe (1986)	8×12
8	Chandrasekharan and Rajagopalan (1986a)	8×20
9	Chandrasekharan and Rajagopalan (1986b)	8×20
10	Mosier and Taube (1985a)	10×10
11	Chan and Milner (1982)	10×15
12	Askin and Subrammania (1987)	14×24
13	Stanfel (1985)	14×24
14	McCormick et al. (1972)	16×24
15	Srinivasan et al. (1990)	16×30
16	King (1980)	16×43
17	Carrie (1973)	18×24
18	Mosier and Taube (1985b)	20×20
19	Kumar et al. (1986)	20×23
20	Carrie (1973)	20×35
21	Boe and Cheng (1991)	20×35
22	Chandrasekaran and Rajagopalan (1989) – 1	24×40
23	Chandrasekaran and Rajagopalan (1989) – 2	24×40
24	Chandrasekaran and Rajagopalan (1989) – 3	24×40
25	Chandrasekaran and Rajagopalan (1989) – 5	24×40
26	Chandrasekaran and Rajagopalan (1989) – 6	24×40
27	Chandrasekaran and Rajagopalan (1989) – 7	24×40
28	McCormick et al. (1972)	27×27
29	Carrie (1973)	28×46
30	Kumar and Vannelli (1987)	30×41
31	Stanfel (1985)	30×50
32	Stanfel (1985)	30×50
33	King and Nakornchai (1982)	36×90
34	McCormick et al. (1972)	37×53
35	Chandrasekaran and Rajagopalan (1987)	40×100

As can be seen in Table 8, the algorithm proposed in this paper obtained machine/product groupings, which, with one exception, have a grouping efficacy that is never smaller than any of the best reported results. More specifically, the algorithm obtains, for 14 (40%) problems, values of the grouping efficacy that are equal to the best ones found in the literature and improves the values of the grouping efficacy for 20 (57%) problems. In 8 (23%) problems, the percentage improvement is higher than 5%. For 16 (46%) problems, the solution was obtained in the first generation, showing the good quality and power of the local search heuristic.

Table 8 – Test Results.

Nº	ZODIAC	GRAFICS	CA	GA	Sol.	Gen.	Time (s)	% Improv
1	73.68	73.68	-		73.68	1	0.28	0.00%
2	56.52	60.87	-	68.00	62.50	1	0.35	-8.09%
3	77.36	-	-	77.36	79.59	1	0.47	2.88%
4	76.92	-	-	76.92	76.92	1	0.41	0.00%
5	39.13	53.12	-	46.88	53.13	6	0.73	0.02%
6	70.37	-	-	70.37	70.37	1	0.73	0.00%
7	68.30	68.30			68.30	1	0.96	0.00%
8	85.24	85.24	85.24	85.24	85.25	1	1.28	0.01%
9	58.33	58.13	58.72	58.33	58.72	2	1.27	0.00%
10	70.59	70.59	70.59	70.59	70.59	1	1.36	0.00%
11	92.00	92.00		92.00	92.00	1	1.46	0.00%
12	64.36	64.36	64.36	-	69.86	10	4.60	8.55%
13	65.55	65.55	-	67.44	69.33	1	4.67	2.80%
14	32.09	45.52	48.70	-	51.96	21	6.53	6.69%
15	67.83	67.83	67.83	-	67.83	1	7.51	0.00%
16	53.76	54.39	54.44	53.89	54.86	1	10.50	0.77%
17	41.84	48.91	44.20	-	54.46	32	8.28	11.35%
18	21.63	38.26	-	37.12	42.96	50	9.12	12.28%
19	38.66	49.36	43.01	46.62	49.65	78	9.97	0.59%
20	75.14	75.14	75.14	75.28	76.14	1	12.09	1.14%
21	51.13	-	-	55.14	58.07	2	13.44	5.31%
22	100.00	100.00	100.00	100.00	100.00	1	14.65	0.00%
23	85.11	85.11	85.11	85.11	85.11	1	20.04	0.00%
24	73.51	73.51	73.51	73.03	73.51	1	21.50	0.00%
25	20.42	43.27	51.81	49.37	51.85	114	22.81	0.08%
26	18.23	44.51	44.72	44.67	46.50	117	23.05	3.98%
27	17.61	41.67	44.17	42.50	44.85	75	22.56	1.54%
28	52.14	41.37	51.00	-	54.27	8	19.90	4.09%
29	33.01	32.86	40.00	-	43.85	117	34.12	9.63%
30	33.46	55.43	55.29	53.80	57.69	111	34.19	4.08%
31	46.06	56.32	58.7	56.61	59.43	113	40.95	1.24%
32	21.11	47.96	46.30	45.93	50.51	93	41.68	5.32%
33	32.73	39.41	40.05	-	41.71	45	68.99	4.14%
34	52.21	52.21	-	-	56.14	1	58.35	7.53%
35	83.66	83.92	83.92	84.03	84.03	3	125.33	0.00%

ZODIAC - Grouping efficacy obtained by ZODIAC, Srinivasan and Narendran (1991)
GRAFICS - Grouping efficacy obtained by GRAFICS, Srinivasan and Narendran (1991)
CA - Grouping efficacy obtained by clustering algorithm in Srinivasan (1994)
GA - Grouping efficacy obtained by genetic algorithm in Cheng et al. (1998)
Sol. –Grouping efficacy obtained by proposed algorithm
Gen. - Generation number where best grouping efficacy was obtained
Time (s) – CPU time in seconds for 150 generations
% Improv. - Percentage improvement of the proposed algorithm against the best of the other four approaches.

For Problem 2, the algorithm produces a grouping efficacy that is lower that the one reported by Cheng et al. (1998). Having discussed this matter with C. H. Cheng, we learned that the solutions obtained by the approach presented in Cheng et al. include singleton groups. We modified our algorithm to also allow singletons and obtained a grouping efficacy of 69.57% for Problem 2, better than the solution found by Cheng et al. Figure 7 presents both solutions.

To further evaluate the performance of the local search heuristic, another test was run. In this test, the proposed algorithm was run with the local search heuristic

```
   14 235
  --------
1 |XX|   |
7 |X |   |
  --------
2 | X|X X|
3 | X|XX |
4 | X|XXX|
5 |X |XXX|
6 |X | XX|
  --------
```

a) – Solution of problem 2 without singletons, grouping efficacy = 62.50 %.

```
    1 2345
   --------
1 |X|  X |
6 |X| X X|
7 |X|    |
   --------
2 | |X XX|
3 | |XXX |
4 | |XXXX|
5 |X|XX X|
   --------
```

b) – Solution of problem 2 with singletons, grouping efficacy = 69.57 %.

replaced by the customary allocation rule, i.e. products are allocated to the cell where it visits the maximum number of machines (since the machine cells are known). Figure 8 shows a graph with the percentage improvement of the grouping efficacy obtained by local heuristic over the customary allocation rule.

Figure 7 - Solution of problem 2 with and without singletons**.**

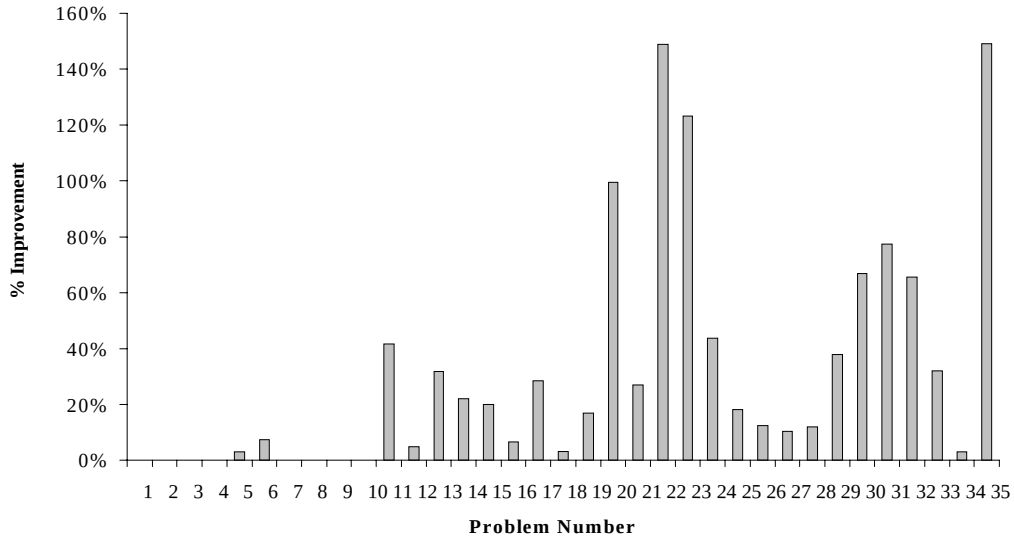


Figure 8 – % improvement of the **Local Search Heuristic** w.r.t. the **Customary Allocation Rule.**

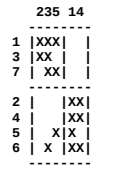
6. Conclusion

A new approach for obtaining machine cells and product families has been presented. The approach combines a local search heuristic with a genetic algorithm. The genetic algorithm uses a random keys alphabet, an elitist selection strategy, and a parameterised uniform crossover. Computational experience with the algorithm, on a set of 35 GT problems from the literature, has shown that it performs remarkably well. The algorithm obtained solutions that are at least as good as the ones found the literature. For 57% of the problems, the algorithm improved the previous solutions, in some cases by as much as 12%.

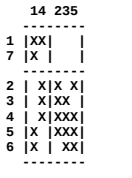
Acknowledgements

The authors would like thank (in alphabetical order): A. S. Carrie, C. H. Cheng, Y. Gupta, B. Mahadevan, G. Srinivasan, and L. Stanfel, for providing and helping to collect the problem data sets.

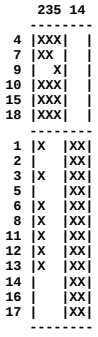
Appendix A



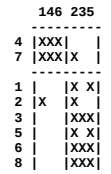
King and Nakornchai (1982) – 5×7



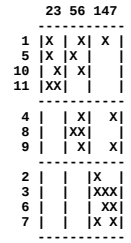
Waghodekar and Sahu (1984) - 5×7



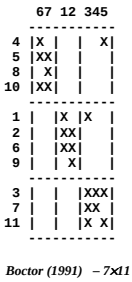
Seiffodini (1989) – 5×18



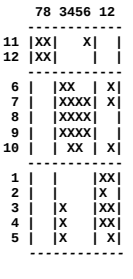
Kusiak (1992) – 6×81



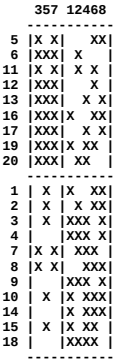
Kusiak and Chow (1987) – 7×11



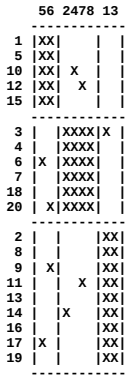
Boctor (1991) – 7×11



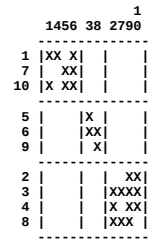
Seiffodini and Wolfe (1986) – 8×12



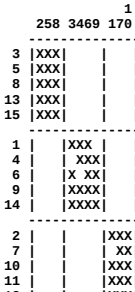
Chandrasekaran and Rajagopalan (1986a) - 8×20



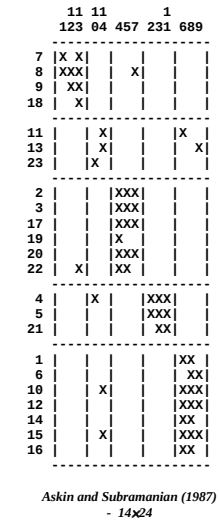
Chandrasekaran and Rajagopalan (1986b) - 8×20



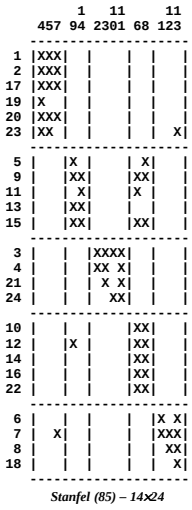
Mosier and Taube (1985a) – 10×10



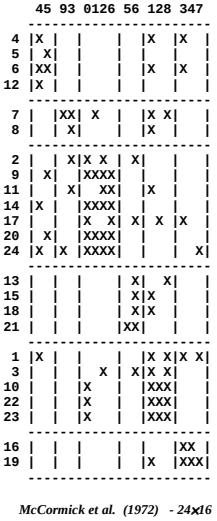
Chan and Milner (1982) – 10×15



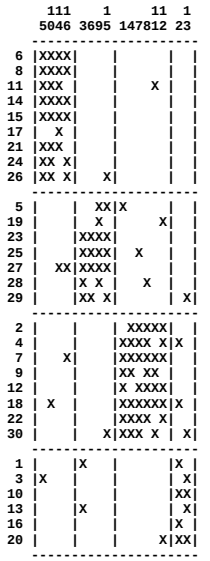
Askin and Subramanian (1987) - 14×24



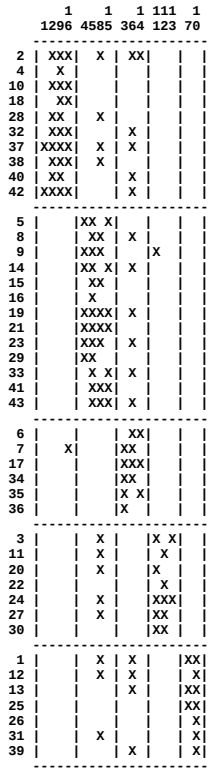
Stanfel (85) – 14×24



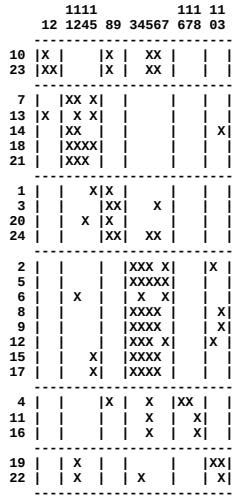
McCormick et al. (1972) - 24×16



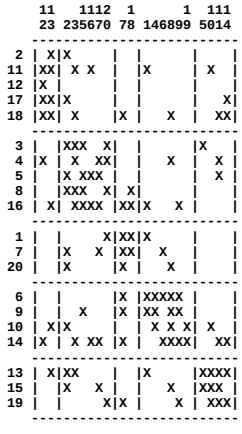
Srinivasan et al.(1990)
-16×30



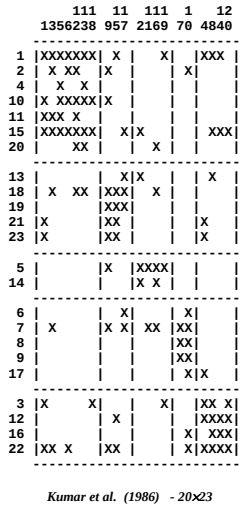
King (1980) - 16×43



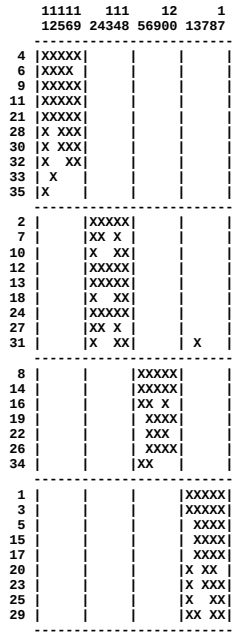
Carrie (1973)- 18×24



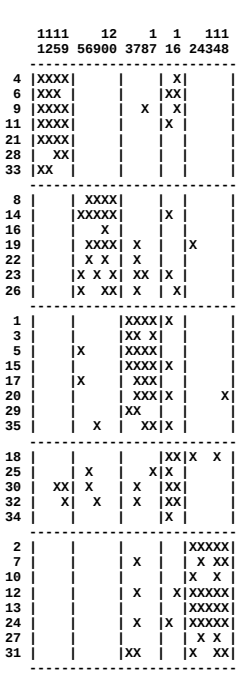
Mosier and Taube (85b) – 20×20



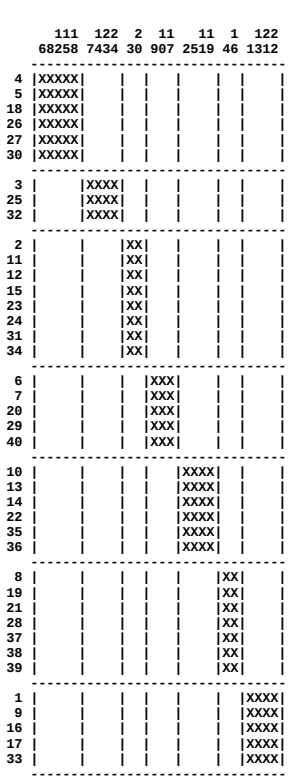
Kumar et al. (1986) - 20×23



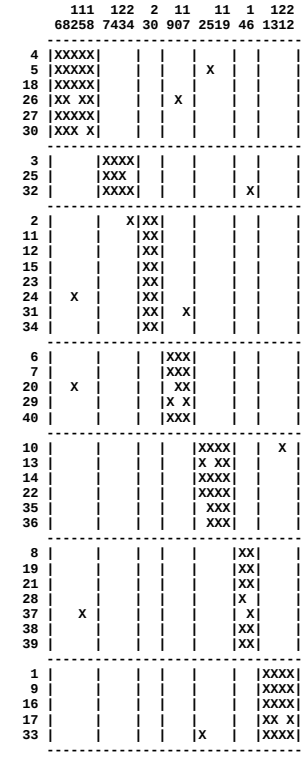
Carrie (1973) - 20×35



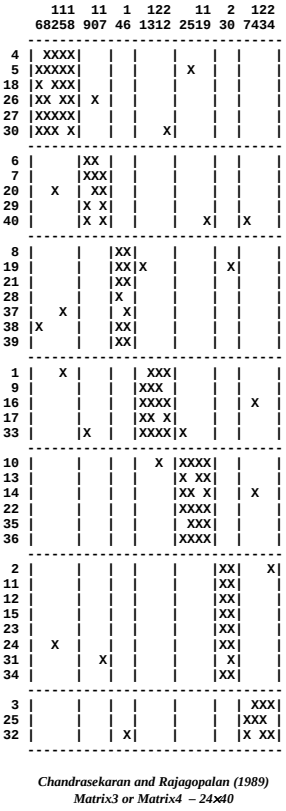
Boe and Cheng (1991) – 20×35



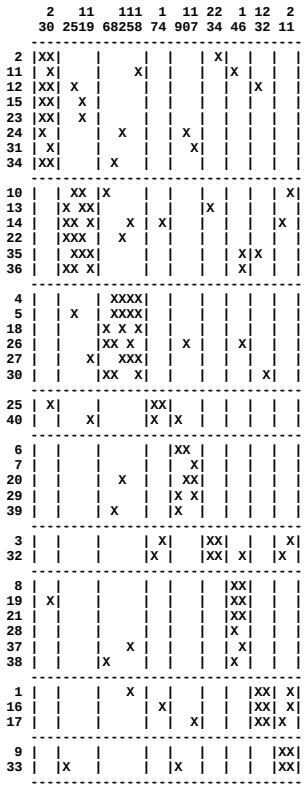
Chandrasekaran and Rajagopalan (1989)
Matrix1 – 24×40



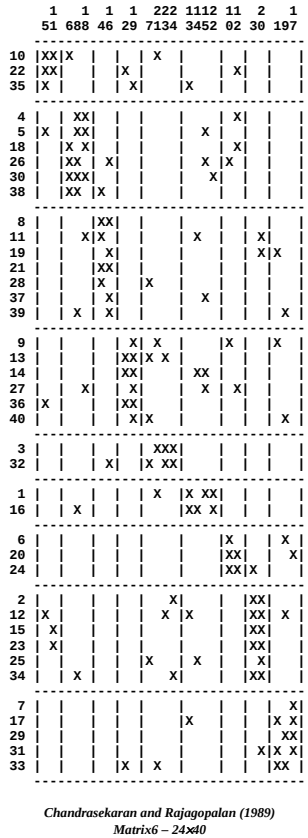
Chandrasekaran and Rajagopalan (1989)
Matrix2 – 24×40



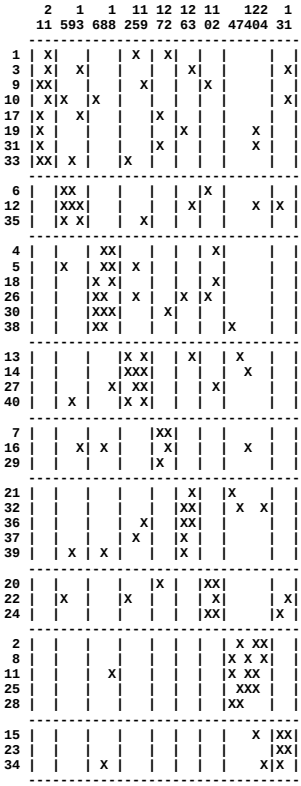
Chandrasekaran and Rajagopalan (1989)
Matrix3 or Matrix4 – 24×40



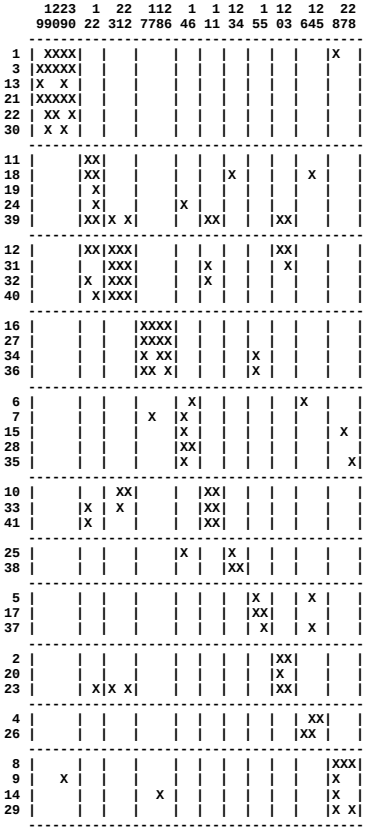
Chandrasekaran and Rajagopalan (1989)
Matrix5 – 24×40



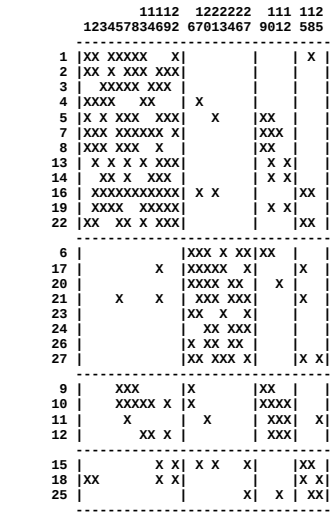
Chandrasekaran and Rajagopalan (1989)
Matrix6 – 24×40



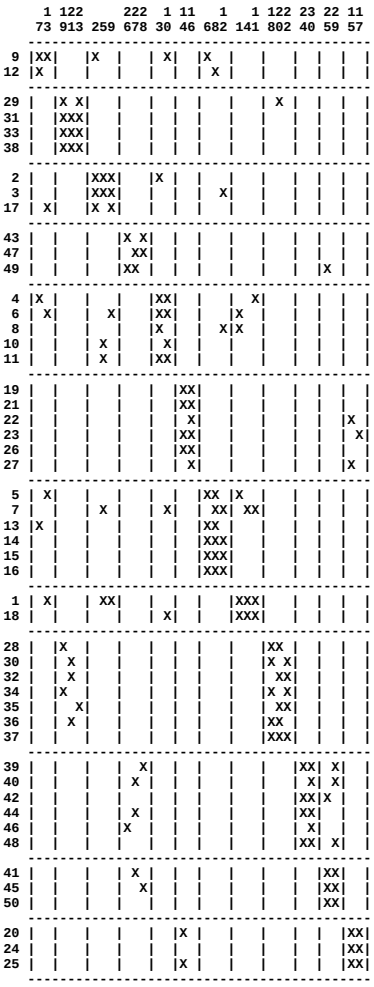
Chandrasekaran and Rajagopalan (1989)
Matrix7 – 24×40



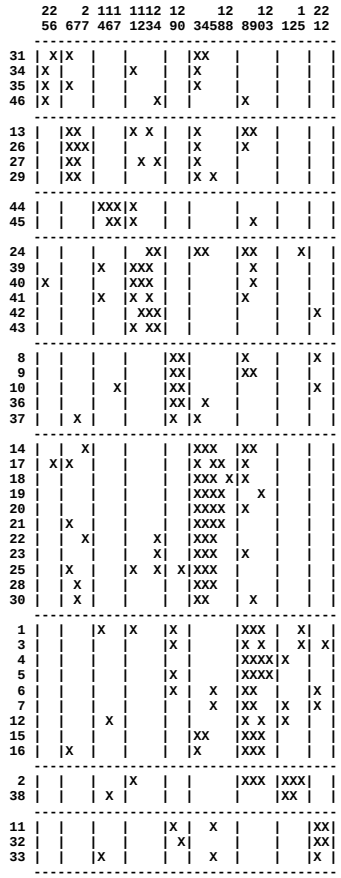
Kumar and Vannelli (1987) - 30×41



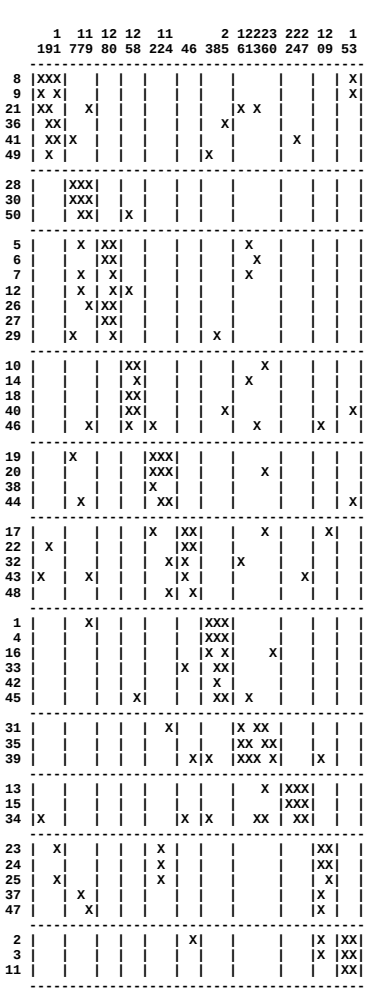
McCormick et al. (1972) - 27×27



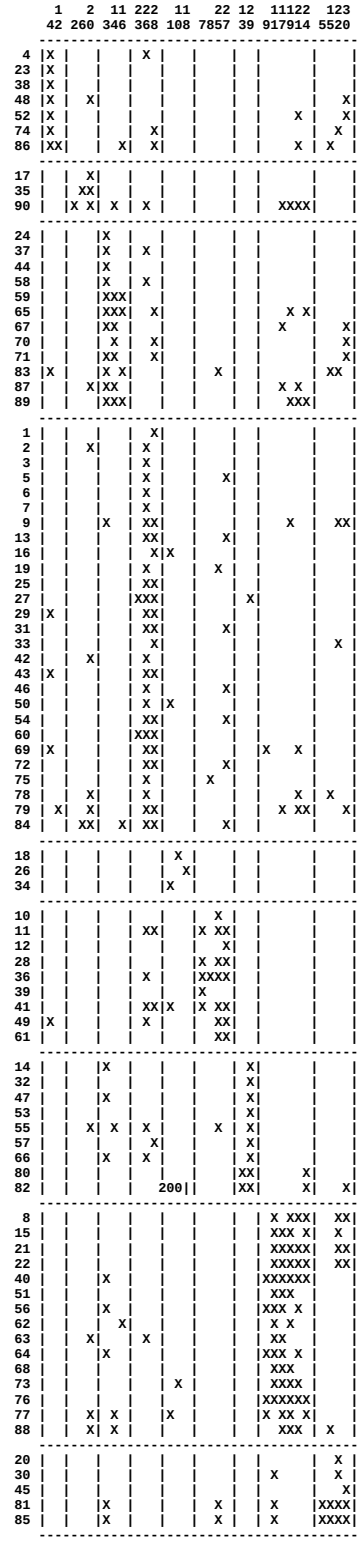
Stanfel (1985), (fig. 5) – 30×50



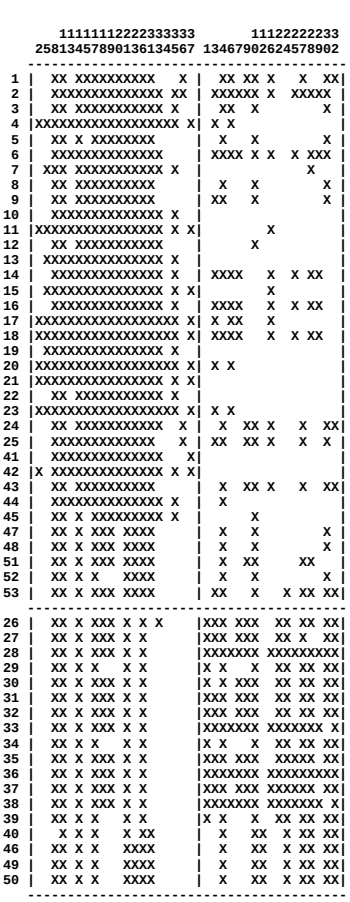
Carrie (1973) – 28×46



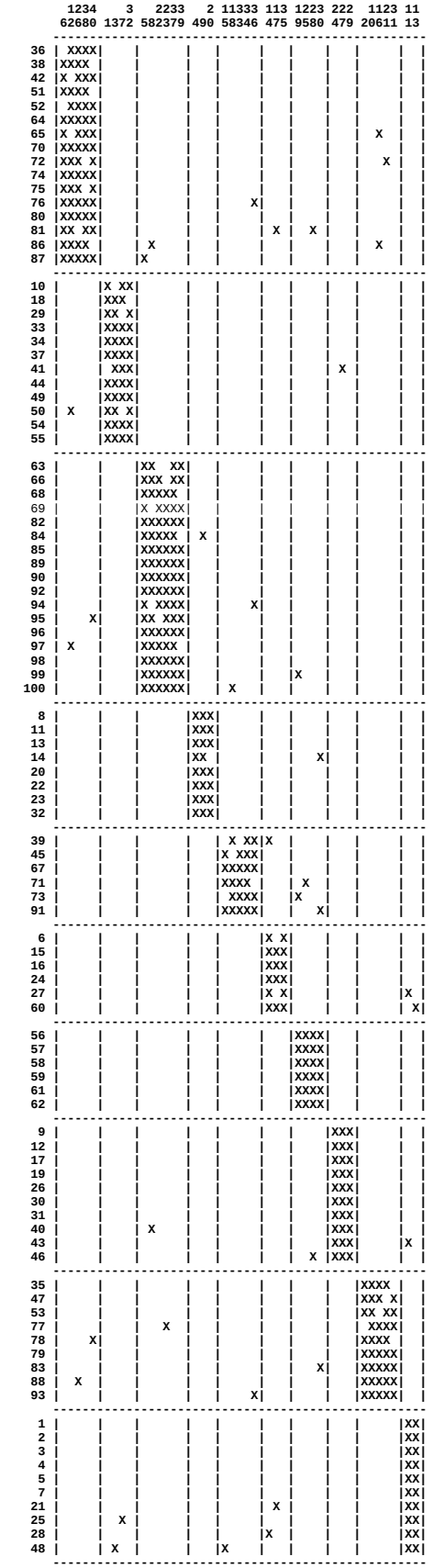
Stanfel (1985), (fig. 6) – 30×50



King and Nakornchai (1982) - 36×90



McCormick et al. (1982) – 37×53



Chandrasekaran and Rajagopalan (1987) – 40×100

References

Akturk, M.S., and Turkcan, A., 2000. Cellular manufacturing system design using a holonistic approach. *International Journal of Production Research,* 38(1) 2327-2347.

Askin, R. G. and Subramanian, S., 1987. A cost-based heuristic for group technology configuration. *International Journal of Production Research,* 25(l) 101-113.

Askin, R.G., and Chiu K. S., 1990. A graph partitioning procedure for machine assignment and cell formation in group technology. *International Journal of Production Research,* 28(8) 1555-1572.

Askin, R.G.. Cresswell, S. H.. Goldberg J. B. and Vakharia A. J., 1991. Hamiltonian path approach to reordering the part-machine matrix for cellular manufacturing. *International Journal of Production Research, 29,* 1081-1100.

Ballakur, A. and Steudel, H.J., 1987. A within-cell utilization based heuristic for designing cellular manufacturing systems. *International Journal of Production Research,* 25(5) 639-665.

Bean, J. C., 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing,* 6(2) 154-160.

Bertsekas, D. and Tseng, P., 1988. Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research,* 36(l) 93-114.

Boe, W. and Cheng, C.H., 1991. A close neighbor algorithm for designing cellular manufacturing systems. *International Journal of Production Research,* 29(10) 2097-2116.

Burbidge, J. L., 1979. Group Technology in Engineering Industry (London: mechanical engineering Publications).

Burbidge, J. L., 1969. An introduction of group technology in *Proceedings of the Seminar on GT, Turin.*

Carrie, S., 1973. Numerical Taxonomy applied to Group Technology and Plant Layout *International Journal of Production Research.* 11, 399-416.

Caux, C., Bruniaux, R., and Pierreval, H: 2000. Cell formation with alternative process plans and machine capacity constraints: A new combined approach. *International Journal of Production Economics,* 64 (1-3) 279-284.

Chan, H. M. and Milner, D. A., 1982. Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing System, 1,* 65-75.

Chandrasekharan, M.P. and Rajagopalan, R., 1989. GROUPABILITY: Analysis of the properties of binary data matrices for group technology. *International Journal of Production Research,* 27(6) 1035-1052.

Chandrasekharan, M.P. and Rajagopalan, R., 1987. ZODIAC - An algorithm for concurrent formation of part families and machine cells. *International Journal of Production Research.* 25(6) 835-850.

Chandrashekharan, M. P., and Rajagopalan, R., 1986a. An ideal seed non-hierarquical clustering algorithm for cellular manufacturing. *International Journal of Production Research,* 24(2) 451-464.

Chandrashekharan, M. P., and Rajagopalan, R., 1986b. MODROC**:** An extension of rank order clustering for group technology. *International Journal of Production Research,* 24(5) 1221-1233.

Chen, J. and Heragu, S.S., 1999. Stepwise decomposition approaches for large scale cell formation problems. *European Journal of Operational Research,* 113(1) 64-79.

Cheng, C.H., Gupta, Y.P., Lee, W.H. and Wong, K.F., 1998. A TSP-based heuristic for forming machine groups and part families. *International Journal of Production Research.* 36(5) 1325-1337.

Choobineh, F., 1988. A framework for the design of cellular manufacturing systems. *International Journal of Production Research,* 26(7) 1161-1172.

Chu, C.H. and Tsai, M., 1990. A comparison of three array-based clustering techniques for manufacturing cell formation. *International Journal of Production Research,* 28(8) 1417-1433.

Co, H.C. and Araar, A., 1988. Configuring cellular manufacturing systems. *International Journal of Production Research,* 26(9) 1511-1522.

Dimopoulos, C., and Mort, N., 2001. A hierarchical clustering methodology based on genetic programming for the solution of simple cell-formation problems. *International Journal of Production Research,* 39(1) 1-19.

Goldberg, D.E., 1989. Genetic Algorithms in Search Optimization, and Machine Learning, Addison-Wesley, Reading, MA.

Gunasingh, K. R., and Lashkari, R. S., 1989a. Machine grouping problem in cellular manufacturing systems-an integer programming approach. *International Journal of Production Research,* 27(9), 1465-1473.

Gupta, T. and Seifoddini, H., 1990. Production data based similarity coefficient for machine-component grouping decisions in the design of a cellular manufacturing system. *International Journal of Production Research,* 28(7) 1247-1269.

Holland, J. H., 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan.

Joines, J.A., Culbreth, C.T. and King, R.E., 1996. Manufacturing cell design: An integer programming model employing genetic algorithms. *IIE Transations*, 28, 69-85.

Khan, M., Islam, S. and Sarker, B., 2000. A similarity coefficient measure and machine-parts grouping in cellular manufacturing systems. *International Journal of Production Research,* 38(3) 699-720.

Khator, S.K. and Irani, S.K., 1987. Cell formation in group technology: A new approach. *Computers in Industrial Engineering,* 12(2) 131-142.

King, J.R. (1980) Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. *International Journal of Production Research,* 18(2) 213-232.

King, J.R. and Nakornchai, V., 1982. Machine-component group formation in group technology: Review and extension. *International Journal of Production Research,* 20(2) 117-133.

Kumar, K. R., and Vannelli, A., 1987. Strategic subcontracting for efficient disaggregated manufacturing. *International Journal of Production Research,* 25(12)1715-1728.

Kumar, K. R., Kusiak, A-, and Vannelli, A., 1986. Grouping of parts and components in flexible manufacturing systems. *European Journal of Operations Research,* 24, 387-397.

Kumar, K.R. and Chandrasekharan, M.P., 1990. Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research,* 28(2) 233-243.

Kusiak, A. 1987. The generalized group technology concept. *International Journal of Production Research,* 25(4) 561-569.

Kusiak, A. and Chow, W., 1987. Efficient solving of the group technology problem. *Journal of Manufacturing Systems,* 6(2) 117-124.

Lee, H. and Garcia-Diaz, A., 1993. A network flow approach to solve clustering problems in group technology. *International Journal of Production Research,* 31(3) 603-612.

McAuley, J., 1972. Machine grouping for efficient production. *Production Engineer,* 51(2) 53-57.

Miltenburg, J. and Zhang, W., 1991. A comparative evaluation of nine well-known algorithms for solving the cell formation in group technology. *Journal of Operations Management,* 10(1) 44-72.

Mosier, C.T. and Taube, L., 1985a. The facets of group technology and their impact on implementation, *OMEGA,* 13(6) 381-391.

Mosier, C.T. and Taube, L., 1985b. Weighted similarity measure heuristics for the group technology machine clustering problem. *OMEGA,* 13(6) 577-583.

Ng, S., 1993. Worst-case analysis of an algorithm for cellular manufacturing. *European Journal of Operational Research,* 69(3) 384-398.

Ng, S., 1996. On the characterization and measure of machine cells in group technology. *Operations Research,* 44(5) 735-744.

Onwubolu, G.C., Mutingi, M., 2001. A genetic algorithm approach to cellular manufacturing systems. *Computers and Industrial Engineering*, 39(1-2) 125-144.

Plaquin, M., Pierreval, H., 2000. Cell formation using evolutionary algorithms with certain constraints. *International Journal Of Production Economics*, 64 (1-3) 267-278.

Rajagopalan, R. and Batra, J.L., 1975. Design of cellular production systems: a graph-theoretic approach. *International Journal of Production Research,* 13(6) 567-579.

Sarker, B. R.., Mondal, S., 1999. Grouping efficiency measures in cellular manufacturing: A survey and critical review. *International Journal of Production Research,* 37(2) 285-314.

Sarker, B. R, 2001. Measures of grouping efficiency in cellular manufacturing systems. *European Journal of Operational Research,* 130 (3) 588-611.

Seifoddini, H., 1989. Single linkage versus average linkage clustering in machine cells formation applications. *Computers and Industrial Engineering,* 16(3) 419-426.

Selim, H.M., Askin, R.G. and Vakharia, A.J., 1998. Cell formation in group technology: Evaluation and directions for future research. *Computers and Industrial Engineeraing ,* 34(1) 3-20.

Shtub, A., 1989. Modelling group technology cell formation as a generalized assignment problem. *International Journal of Production Research,* 27(5) 775-782.

Spears, W.M. and DeJong, K.A., 1991. On the virtues of parameterized uniform crossover, in *Proceedings of the Fourth International Conference on Genetic Algorithms,* 230-236.

Srinivasan, G. and Narendran, T.T., 1991. GRAFICS - A nonhierarchical clustering-algorithm for group technology. *International Journal of Production Research,* 29(3) 463-478.

Srinivasan, G., Narendran, T. and Mahadevan, B., 1990. An assignment model for the part-families problem in group technology. *International Journal of Production Research,* 28(l) 145-152.

Uddin, M. K., Shanker K., 2002. Grouping of parts and machines in presence of alternative process routes by genetic algorithm, *International Journal of Production Economics*, 76(3) 219-228

Vannelli, A., and Kumar, K. R., 1986. A method for finding minimal bottleneck cells for grouping part-machine families. *International Journal of Production Research,* 24(2) 387-400.

Vohra, T., Chen, D., Chang, J. and Chen, H., 1990. A network approach to cell formation in cellular manufacturing. *International Journal of Production Research,* 28(11) 2075-2084.

Won,Y., 2000. Two-phase approach to GT cell formation using efficient p-median formulations. *International Journal of Production Research,* 38(7) 1601-1613.

Yasuda, K., Yin, Y., 2001. A dissimilarity measure for solving the cell formation problem in cellular manufacturing. *Computers and Industrial Engineering*, 39 (1-2) 1-17.

Zhao, C., and Wu, Z., 2000. A genetic algorithm for manufacturing cell formation with multiple routes and multiple objectives. *International Journal of Production Research,* 38(2) 385-395.