Int J Adv Manuf Technol (2008) 36:589–597
DOI 10.1007/s00170-006-0862-2

ORIGINAL ARTICLE

An efficient ant colony optimization system for the manufacturing cells formation problem

K. Spiliopoulos · S. Sofianopoulou

Received: 27 July 2006 / Accepted: 24 October 2006 / Published online: 6 December 2006

Abstract An ant colony optimization (ACO) scheme for the manufacturing cells design problem is proposed, which uses a tight eigenvalue-based bound to guide and accelerate the search. This feature is combined with a good initialization procedure and with ideas from successful ACO implementations in other areas, to achieve efficiency and reliability with the minimum structure and set of parameters. The resulting algorithm produces most promising results for medium to large size problems, with negligible computational effort.

Keywords Cellular manufacturing · Cell formation · Ant colony optimization

1 Introduction

Cellular manufacturing is a key production strategy, in the framework of group technology. The general idea is to decentralize processing and create manufacturing cells, which are effectively flow shop "islets" in a job shop environment. This is achieved by grouping the machines into clusters and the various parts into part families, and then allocating the processing of each part family to a single machine cluster. There are many benefits from such a production scheme, see for example [1]. These include the reduction of transport, queuing and processing times, the elimination of the need for frequent set-ups and tool changes in the machines, the reduction of inventories and the simplification of the production plan. These benefits lead to better delivery times, quality improvements, more efficient management and customer satisfaction. The application of cellular manufacturing is also an appropriate first step towards unmanned production [2, 3] and JIT production schemes [4].

In practice, the creation of completely independent manufacturing cells is very seldom feasible [5] and one attempts to decrease as much as possible the intercellular traffic, that is the traffic generated by parts visiting machines in different cells. This optimization problem is usually called the cell formation problem (CFP) and bears almost all the computational complexity of the design. The result is an initial solution which can then be improved with machine duplication, parts subcontracting and consideration of alternative process plans. A brief description and evaluation of methods proposed for the CFP is given in [6–9] among others. Extended bibliographies are also given in [10, 11].

CFP is a NP-hard clustering problem and as such difficult to solve, if the problem is not over-simplified (for example with statistical clustering, p-median models or network flow methods). Exact algorithms based on mathematical programming models do exist (see for example [12–14]), but because these are computationally expensive, the usual approach for medium to large-scale problems is to use a heuristic technique to obtain near-optimal solutions. Typical examples are simulated annealing [9, 15], tabu search [16, 17], genetic algorithms [18, 19] and neural networks [20, 21].

In this work an application of the ant colony optimization (ACO) meta-heuristic framework is presented. Due to their successful application in the quadratic assignment problem (QAP), ACO systems have been employed for QAP-related problems in group technology, see for example [22] for a layout design problem. An ACO algorithm

K. Spiliopoulos · S. Sofianopoulou (✉)
Department of Industrial Management & Technology,
University of Piraeus,
80 Karaoli & Demetriou Str.,
185 34 Piraeus, Greece
e-mail: sofianop@unipi.gr

for the CFP has also been proposed in [23]. This last work is based on virtual "tours" on a graph with potentially too many nodes, corresponding to the machines, the parts and the cells to be formed. The present, however, approach is a graph partitioning one, with nodes corresponding to the machines only, as it will be described in Sect. 2. The proposed algorithm also addresses an important consideration in ACO systems, namely the use of a tight bound which is employed to evaluate solutions properly and accelerate the search.

The problem statement is presented in Sect. 2, including the integer programming formulation for the CFP problem and the method for the calculation of the "distances" between machines. Section 3 presents the description of the algorithm and computational results with test cases are given in Sect. 4. Finally, conclusions are given in Sect. 5.

2 Problem statement

To eliminate dimensionality problems associated with the existence of many parts and the use of the traditional 0-1 part-machine incidence matrix, we prefer to form groups of machines according to "distances" between them. The manufacturing cells are then trivially completed by allocating each part to the group where it induces the most inside traffic. The data are given in the form of a part-machine sequence matrix $A=(\alpha_{pi})$, $p=1,...,P$, $i=1,...,n$, where P is the number of parts and n the number of machines:

$$\alpha_{pi} = \begin{cases} t, \text{ if part } p \text{ visits machine } i \text{ at step } t\ (1 \le t \le n) \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

For each pair of machines i, j and for each part p, let:

$$\delta_{ijp} = \begin{cases} 1, \text{ if} \alpha_{pi}\ \alpha_{pj} \neq 0 \text{ and } \left|\alpha_{\mathrm{pi}} - \alpha_{pj}\right| = 1 \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

Then, in (2), δ_{ijp} shows whether part p visits machines i, j in two successive steps or not. Collecting the contributions from all parts, to represent the traffic between machines, let:

$$c_{ij} = \sum_{p=1}^{P} \delta_{ijp} \quad i = 1,\ldots,n-1 \quad j = i+1,\ldots,n \tag{3}$$

With the cost matrix $C=(c_{ij})$ as defined above, we can use the following 0-1 integer programming formulation for the CFP with bounded cell size from above:

$$z = \min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}\left(1 - x_{ij}\right) \tag{4}$$

s.t.

$$\sum_{i=1}^{k-1} x_{ik} + \sum_{j=k+1}^{n} x_{kj} \le M - 1 \quad k = 1,\ldots,n \tag{5}$$

$$\left.\begin{array}{l} x_{ij} + x_{ik} - x_{jk} \le 1 \\ x_{ij} - x_{ik} + x_{jk} \le 1 \\ -x_{ij} + x_{ik} + x_{jk} \le 1 \end{array}\right\} \begin{array}{l} i = 1,\ldots,n-2 \\ j = i+1,\ldots,n-1 \\ k = j+1,\ldots,n \end{array} \tag{6}$$

$$x_{ij} = \begin{cases} 1, \text{ if machines } i,j \text{ are in the same cell} & i = 1,\ldots,n-1 \\ 0, \text{ otherwise} & j = i+1,\ldots,n \end{cases} \tag{7}$$

The objective function (4) minimizes the inter-cellular moves. Constraint (5) ensures that any machine is in the same cell with at most M-1 other machines, and constraints (6), the "triangular" constraints, enforce the integrity of cells.

This formulation for the CFP was first given in [9, 24] and takes into account the most important parameters, namely the processing sequence of the parts in the routings and the maximum allowed size for the cells. The importance of considering the processing sequence of the parts in a routing, a somewhat neglected factor, is discussed in [25]. The formulation combines simplicity and adequate fit to realistic situations. The modifications for the case of parts re-visiting some machines are trivial: in definition (1), α_{pi} can take more than one non-zero values, all of which must be taken into account in (2). Other considerations, such as having volume weighting in the routings, although not tested in this work, can be easily incorporated in the proposed algorithm. In this case, the elements of the cost matrix are multiplied by corresponding factors.

CFP is known to be NP-hard, see [26]. In fact, when the cell sizes are fixed, it can be equivalently formulated as a quadratic assignment (QAP), a quadratic transportation (QTP) or a matrix norm minimization problem, see for example [14]. The QTP formulation is of relevance; therefore, it is also described here.

If there are K cells of pre-specified sizes $m_1, m_2,..., m_k$ and y_{ik}, $i=1,...,n$ $k=1,...,K$ are binary variables indicating whether machine i is allocated to cell k or not, then the variables x_{ij} in (7) are connected with the new variables y_{ik} with the relation:

$$x_{ij} = \sum_{k=1}^{K} y_{ik} y_{jk} \quad i = 1,\ldots,n-1 \quad j = i+1,\ldots,n \tag{8}$$

and the optimization problem (4)–(7) can equivalently be stated as follows:

$$z = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} - \max \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{K} c_{ij} y_{ik} y_{jk} \tag{9}$$

$$\sum_{k=1}^{K} y_{ik} = 1 \qquad i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} y_{ik} = m_k \qquad k = 1, \ldots, K \tag{11}$$

$$y_{ik} \in \{0, 1\} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, K \tag{12}$$

In this formulation, the maximization term in (9) expresses the within-cells cost, that is the intracellular traffic. Constraints (10) and (11) ensure that each machine is placed in exactly one cell and that the cells are of the pre-specified sizes.

3 Description of the algorithm

In this section, the implementation of the ant colony optimization (ACO) procedure is described. The ACO meta-heuristic is by now an established optimization framework, therefore much of the usual introductory description is omitted. The interested reader can find a complete textbook description in [27]. An extended description, together with a review of applications in important combinatorial problems is also given in [28]. As noted in the introduction, some strategic elements of the ACO implementation are based on ideas from QAP-related work, notably the algorithm described in [22] for the solution of the inter-cell layout problem.

In the present work the problem is first transformed to an equipartition one, by adding dummy machines without any parts processing, therefore creating "buckets" of capacity equal to the maximum cell size. This effectively transforms the problem to the quadratic transportation problem (QTP) (9) – (12) with equal capacities for the "destinations". If the number of the machines is already a multiple of the maximum cell size, another "bucket" is added, to allow for instances where the optimal solution is not an equipartition. This maximum number of cells may also be increased during the calculation of the initial solution, as it will be explained later in this section.

At each iteration of the algorithm, there is a pre-specified number Q of virtual ants, each one building a feasible solution. Each solution being built is based on the following criteria:

- A priori available heuristic information.
- The history of the search and the quality of previous solutions found.

The end of an iteration triggers an update of the historical information. Iterations continue until a certain number has passed since the last improvement of the best solution found.

In the following sections, the above main elements of the ACO procedure are presented, that is, the construction of a feasible solution, in Sect. 3.1, and the update of the historical information, in Sect. 3.2. The latter includes the calculation of a lower bound for the problem which is discussed in Sect. 3.3.

3.1 Individual ants solutions

Machines are allocated to cells sequentially. For the heuristic information, we calculate the *attractiveness* η_{ik} of assigning machine i to every candidate cell k that is not already occupied in full:

$$\eta_{ik} = \frac{1}{1 + \sum_{\substack{j=1 \\ j \notin k}}^{n} c_{ij}} \qquad k = 1, \ldots K \qquad k \notin tabu \tag{13}$$

In (13), the *tabu* vector contains the cells that are saturated. Also, $j \notin k$ denotes that machine j is not allocated to cell k. Therefore, the sum in the denominator is the "external" total cost between machine i and machines already allocated to cells other than k. The smaller this value, the more attractive is the allocation of machine i to cell k.

The historical information for this "move", that is the assignment of machine i to the candidate cell k is kept in the *pheromone trail level* τ_{ik}. To take into account both heuristic and historical information, a weighted sum f_{ik} is calculated:

$$f_{ik} = \alpha \tau_{ik} + (1 - \alpha)\eta_{ik} \tag{14}$$

In (14), α is a pre-specified constant. When these values have been evaluated for all candidate cells, they are scaled to represent probabilities, and machine i is assigned to one of the candidate cells. That is, the probability P_{ik} of allocating machine i to a cell k is:

$$P_{ik} = \frac{f_{ik}}{\sum_{k \notin tabu} f_{ik}} \qquad k = 1, \ldots K \qquad k \notin tabu \tag{15}$$

When all machines have been allocated to cells, that is a complete solution has been constructed by an ant, this

solution is improved by swapping machines between cells. The most favorable swap is located, and if this results to a gain, the swap takes place. The swaps continue until there is no more a profitable one.

3.2 Update of historical information

The mechanism followed for the update of historical information is, strictly speaking, a modification of the more common ACO strategy,which includes an *evaporation* parameter, that is, a decay in the influence of pheromone deposited, to reduce the influence of solutions visited in the early stages of the search (see for example the Simple-ACO (S-ACO) algorithm in [27]). This modification was initiated in [29], in the framework of the ANTS method, where it was proposed to evaluate each solution against a moving average, thus eliminating old memory altogether. The mechanism applied in the present work is a further improvement followed and justified in [22]. Instead of a moving average with a fixed number of solutions, each solution is evaluated against the other solutions obtained *in the same iteration*. The main advantage of this strategy is robustness, since one utilizes a limited set of parameters.

More specifically, at the end of each iteration, there are Q individual solutions, $q=1,...,Q$, each one with solution value z^q. If $\bar{z}$ is the average solution value during the iteration, one first calculates the amount of pheromone to add to each "move" (that is, trail) involved in solution q:

$$\Delta\tau_{ik} = \tau_0\left(1 - \frac{z^q - z_{lower}}{\bar{z} - z_{lower}}\right) \tag{16}$$

In (16) z_{lower} is a lower bound on the objective function (4) or (9), discussed in Sect. 3.3, and τ_0 is the maximum trail level increment due to one ant.

Next, these amounts are added for all ants, to update the pheromone trail levels:

$$\tau'_{ik} = \tau_{ik} + \sum_{q=1}^{Q} \Delta\tau_{ik} \tag{17}$$

If any trail level happens to be negative, an appropriate pheromone amount is added to all trails, to make them nonnegative. For the reasons explained in the start of this section, in expression (17), there is no "persistence" parameter, that is, no coefficient reducing the influence of the trail level of the previous iteration. Pheromone is not accumulated without limit, of course, because of the form of (16).

3.3 Calculation of lower bounds

A crucial point in the present algorithm is the calculation of a tight lower bound for the calculation of the pheromone to add to each trail in (16). The importance of this is evident. The individual ants solutions do not typically differ much from the average value in an iteration, therefore, if this bound is too low, the term in parentheses becomes too small, and one has to tune very carefully both the τ_0 parameter and the weighting factor α in (14), to achieve proper differentiation. In other words, the algorithm loses much in terms of robustness.

In the proposed algorithm, this lower bound is obtained before the start of iterations, by solving the following nonsmooth convex optimization problem, with a bundle trust method proposed in [30].

$$z_{lower} = \frac{K-1}{K}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} - \frac{n}{2K}\min\left\{\sum_{j=1}^{k-1}\lambda_j(V^t(C + diag(d))V) : d \in R^n, \sum_{i=1}^{n} d_i = 0\right\} \tag{18}$$

In the above:

- K is the number of cells
- n is the number of machines
- $C=(c_{ij})$ is the symmetric $n\times n$ cost matrix defined in Sect. 2
- The operator λ_j (.) denotes the j-th algebraically largest eigenvalue of a matrix
- V is any n $(n-1)$ matrix with column sums equal to 0 satisfying $V^tV = I_{n-1}$, where V^t is the transpose of matrix $V$and I_{n-1} is the identity matrix of dimension $n-1$
- *diag(d)* is a diagonal matrix formed by the elements of the unknown real vector d of dimension n.

More precisely, the lower bound is the smallest integer larger than or equal to the expression in the right hand size of (18), since the solutions to the CFP have integer values. This bound is an application of the bound for the quadratic transportation problem proposed in [31] in the framework of the graph partitioning problem, for equipartition into K node groups of equal sizes. A possible form of the V matrix is given in [32].

The existence of a good starting point is essential for fast convergence of the bundle trust method. To this end, the present algorithm uses the "merge and swap" heuristic presented in [14]. This is a fast technique that has proved to

be much efficient in practice, especially for initialization purposes. The procedure is very simple:

- Start with each machine in a separate cell and then proceed with the most profitable merges of cells, given the maximum cell size constraint. Repeat until no more cells can be merged.
- Continue with the most favorable swaps of machines between cells. Repeat until there are no more favorable swaps.

If the number of cells created by this heuristic technique is greater than the maximum one set initially, the number of "buckets" is increased accordingly.

An outline of the ant colony optimization procedure is given in the following Fig. 1.

4 Computational results

The proposed algorithm was tested with several problems. The machine used for the tests was a Pentium III 256MB with processor speed 1 GHz. The programming language was Fortran 90 and the compiler was NAGWare f95 for Windows (which produces C code for the Cygwin gcc compiler).

The parameters used by default were:

- weighting factor in (14), α=0.7
- maximum trail level increment due to one ant in (16), τ_0=0.25
- number of ants $Q=n$, number of machines
- maximum iterations=n, number of machines

In all the tests that are described in this section, there was no significant sensitivity to relatively small changes in these settings.

The main series of runs involved all 265 problems that were tested in [17] with a tabu search (TS) algorithm. Except from CFP data with 16–30 machines, these also included some large graph partitioning problems based on graphs with 35–50 nodes, which, to be precise, are in a sense "easier" than true CFP data (the cost matrix elements have a very large spread, therefore there is a natural priority in the variables) but can anyway provide indications for the performance of the proposed algorithm in large dimensions.

Although the results in this past work with the TS algorithm were already good, the ACO algorithm per-

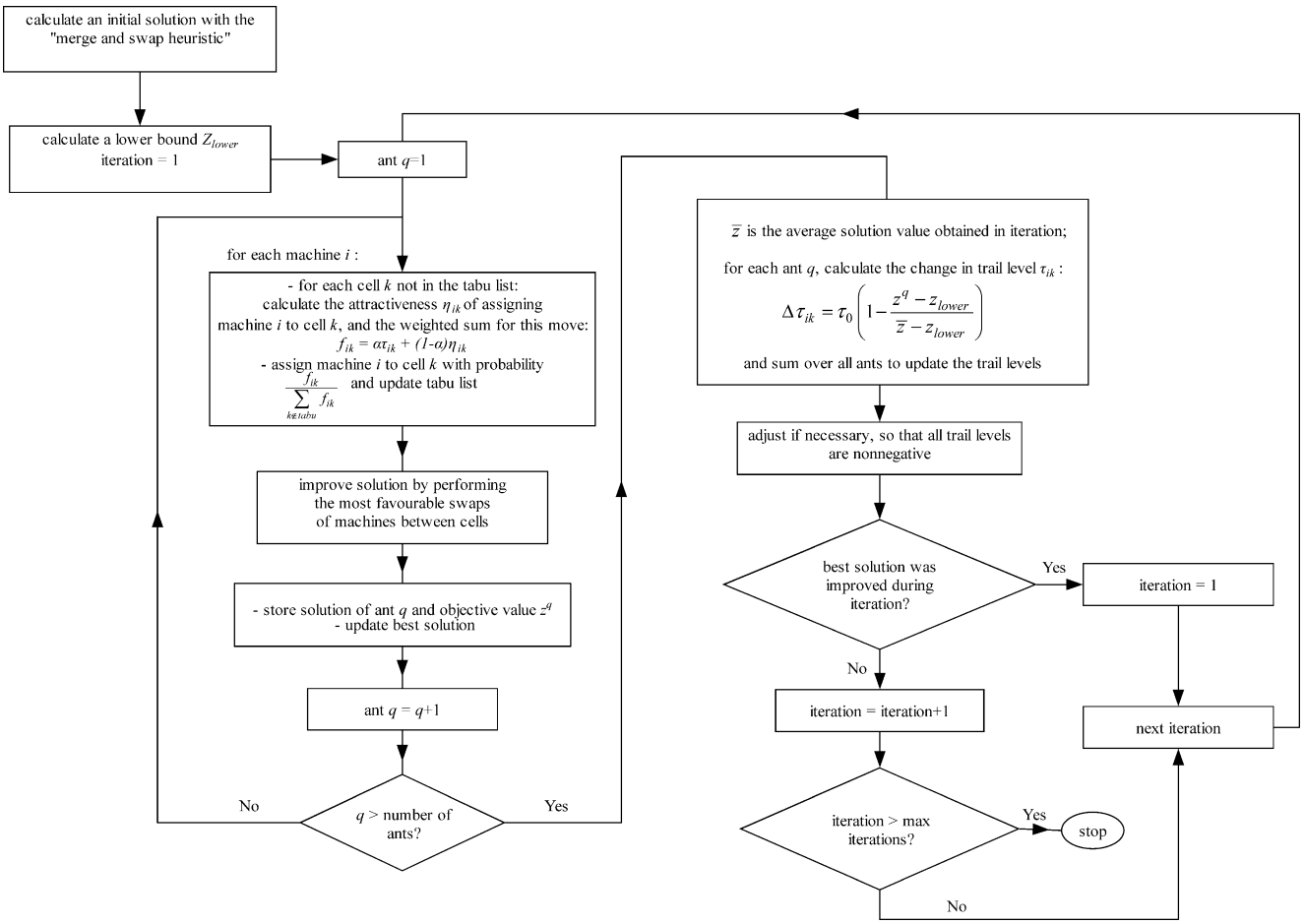


Fig. 1 Outline of the ant colony optimization procedure

Table 1 Comparative performance of tabu search (TS) and ACO algorithms

Dimension	Number of problems	Known optimal solutions	TS Results	ACO Results	TS iterations limit	ACO parameters
16	80	yes	Optimal solutions	Optimal solutions	1000	(n,n)
20	50	yes	Optimal solutions	Optimal solutions	1000	(n,n)
25	35	yes	Optimal solutions	Optimal solutions	1000	(n,n)
30	45	yes	Optimal solutions	Optimal solutions	1000	(n,n)
35–40	10	yes	Optimal solutions	Optimal solutions	1000[a]	(n,n)
35–40	30	no	The best known solutions, improvement in 4 instances	Further improvement in 1 of the 4 better values	5000	(n,n)
50	5	yes	1 sub-optimal solution	Optimal solutions	5000	(2n,2n)
50	10	no	The best known solutions	Inferior solution in 1 instance	5000[b]	(2n,2n)

[a] Except from one problem where the optimal solution was found in iteration 2404.
[b] Except from two problems where the best known solutions were found with iterations limit set to 10 000.

formed even better and found the same solutions faster. In the solutions that differed, it improved the best known values in two cases and lost only one. A comparison is shown in Table 1 below, copied from that work and completed with two columns containing the ACO results and settings.

In this table, existence of known optimal solutions indicates optimal solution of the integer programming model (4)–(7) obtained with exact methods (in large computation times, of course), see [14, 17] for details. Also, the ACO parameters refer to the pair (number of ants, maximum number of iterations) expressed as functions of n, the number of machines. These parameters were doubled for problems with 50 nodes, a size which was experimentally found to be too large for the default values. As shown in Fig. 1, the number of ants determines the number of solutions tried per iteration, therefore doubling these two parameters approximately quadruples the number of solutions visited, and the algorithm reaches again its full potential.

Computation times are not noted because they were small. The largest average computation times with the TS algorithm in [17] were 53–59 seconds in the two groups of problems with 50 nodes. The corresponding time with the ACO algorithm was 18 seconds. Note that the computation times can be further reduced if the size of the problem does not exceed about 20 machines. In that case, one can restrict the maximum number of bundle trust iterations for the minimization of (18), without losing much in accuracy. In the current work, which includes the solution of large problems, it was preferred to keep this iterations limit at a relatively high and conservative level in all runs, for reasons of uniformity.

For completeness, the tests included some indicative compatible problems in the literature, with at least 20 machines. These are listed in Table 2 below.

All problems were stated in terms of the classic 0–1 part-machine incidence matrix, and they were modified to have processing sequence information for all parts. To do this, it was simply assumed that the processing sequence of each part increases with machine number. An illustration with a small example is given in Fig. 2 below.

Problems P1–P4 are solved in [39] using the grouping efficacy criterion (which also takes into account "gaps" in the cells), proposed in [40], and are those from a much larger set in [39] that have at least 20 machines and have comparable solutions, i.e., no two cells have combined size less than or equal to the maximum cell size observed. The present approach is, of course different, in that cells are always merged when possible, to achieve the same degree

Table 2 Other data sets used and corresponding parameters

Problem	Source	Machines	Parts	Maximum cell size
P1	[33]	20	35	5
P2	[34]	20	35	5
P3	[35]	30	50	3
P4	[36]	37	53	20
P5	[37]	24	40	3
P6–P8	[38]	30	41	9,11,15

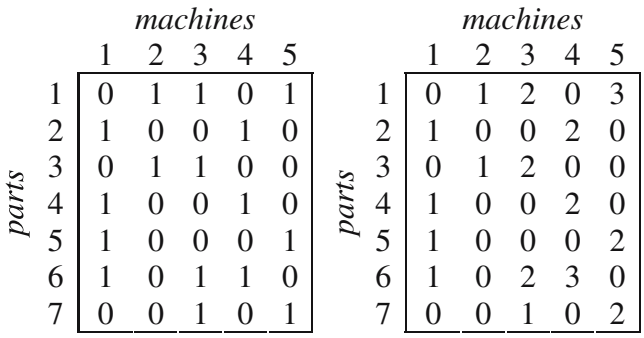


Fig. 2 Example of adding part-machine sequence data

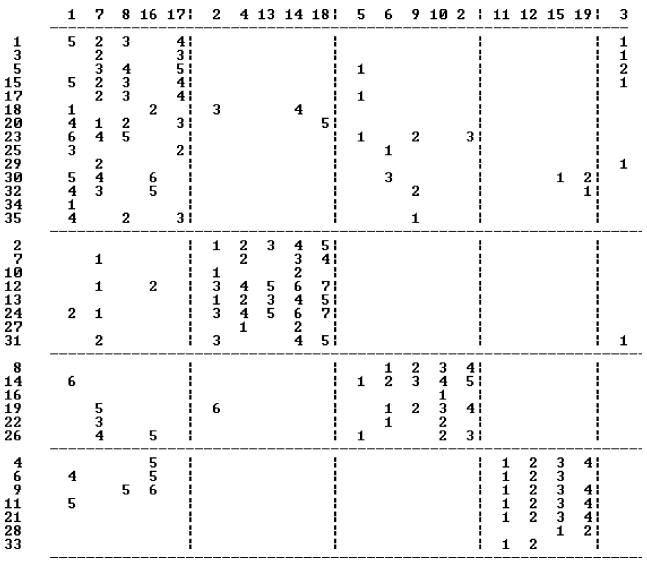


Fig. 3 Solution of problem P2

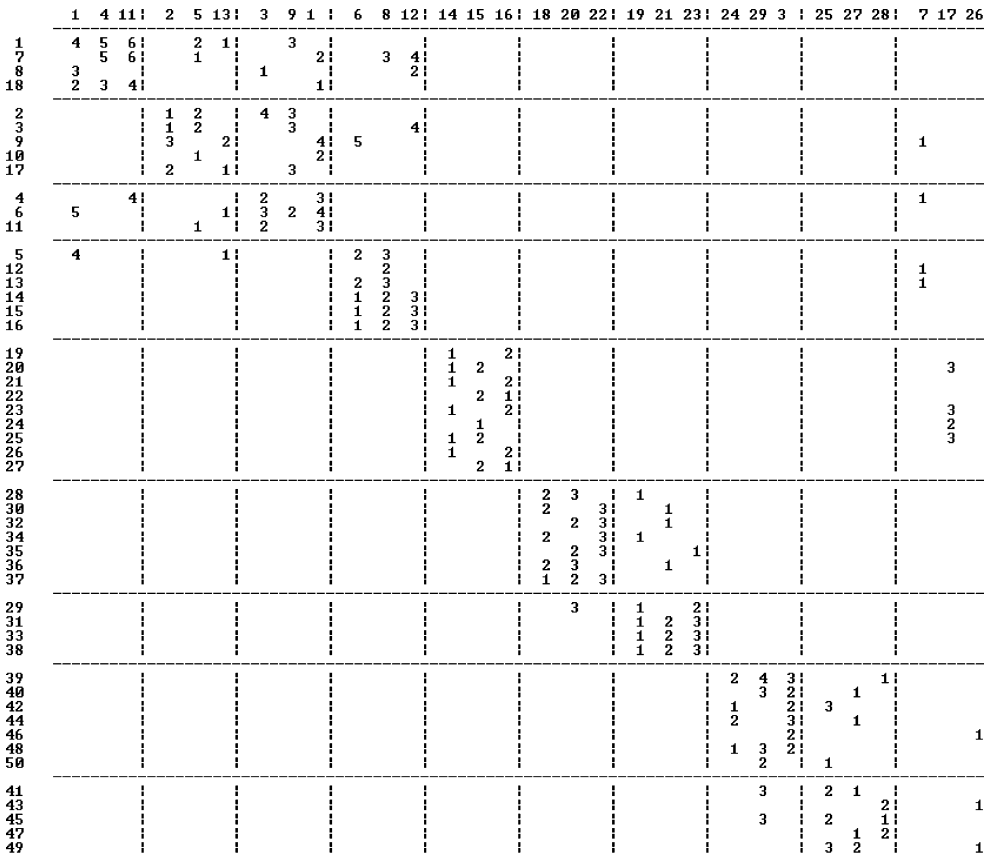
of centralization with fewer cells. For these 4 problems, a decrease in the number of exceptional elements, that is the number of out-of-cells processes was expected, at the expense of the grouping efficacy value. The corresponding problem numbers in [39] are 20, 21, 31 and 34.

For problem P1, the solution reported in [39] has a 76% grouping efficacy, and is obtained in 16 seconds. The proposed algorithm required only 1 second and produced the same solution.

For problem P2, the cited work reports a solution with 58% grouping efficacy, obtained in 16 seconds. The present solution has fewer exceptional elements (39 instead of 41) and, necessarily, a smaller grouping efficacy (53%). It was obtained in only 1 second. For indicative purposes, the solution of problem P2 is given in Fig. 3. The rows in Fig. 3 refer to the parts, while the column blocks represent the machines assigned in each one of the cells formed. The individual numbers along the lines represent the processing sequence for each part.

In this figure and all following, the processing sequence numbers were added in the original data, as already mentioned. It is interesting to note that the proposed algorithm produced 4 cells and left out machine 3 as an isolated one, because the addition of this machine to the only available cell, the last one, is indifferent. In fact, this presentation of the result makes it easier to see that if the maximum cell size is increased, it is worthwhile adding this machine to the first cell, thus eliminating 5 more exceptional elements. The reason such a configuration happens with the proposed algorithm is because there are actually more machines in the solution (the dummy ones) which are not shown.

Fig. 4 Solution of problem P3



For problems P3 and P4, the reported solutions in the cited work had grouping efficacies 60% and 56%, and required computation times 52 and 88 seconds, respectively. The proposed algorithm required 2–3 seconds for each problem. In the first case, the present solution has 10 cells instead of 12, and is better in terms of the number of exceptional elements (49 instead of 50). The grouping efficacy is, of course, smaller (53%). In the second case, the proposed algorithm produced the same solution. The solution of problem P3 is given in Fig. 4. Again, because of the way parts are assigned to groups of machines, the last "cell" is a set of machines only, without corresponding parts. Modifications in the fifth cell to accommodate the processes of machine 17 would eliminate four more exceptional elements.

Problem P5 is the one used in [37] for the illustration of the MST approach. The reported solution had nine cells and a 43% grouping efficacy. The proposed algorithm produced a solution with fewer cells (8) and only slightly smaller grouping efficacy (40%). This solution is shown in Fig. 5.

Finally, problems P6–P8 are from [38], all based on a 30-machines 41-parts problem (dataset "manf1"). This is solved with maximum cell sizes 15, 11 and 9, as observed in the results given for different partitions and parts subcontracting (figures 15–17 in that work). In all cases, the number of exceptional elements coincided with the number of subcontracted parts reported in that work (5, 6 and 8, respectively). There is a slight difference in the criteria used (in the cited work, the authors minimize the number of exceptional parts, not elements), but it so happened that no two exceptional elements in our solutions were in the same part.

5 Conclusions

A new algorithm for the solution of the manufacturing cells design problem is proposed. The model used includes the main parameters of the design, namely the maximum cell size and the processing sequence of the parts, and avoids difficulties associated with the use of the traditionally used part-machine incidence matrix. Other parameters can either be incorporated in the algorithm, or left as part of the subsequent improvement stage.

The proposed algorithm is based on the modern ant colony optimization (ACO) meta-heuristic framework and on a special case formulation of the quadratic transportation problem. Several features are combined to achieve efficiency and robustness, in particular, fast creation of a starting solution, the use of a tight eigenvalue-based bound to

Fig. 5 Solution of problem P5

	19	21	23	1	13	17	4	14	22	12	15	18	2	5	9	3	10	11	7	16	2	6	8	24
1		2							3		1													
3		3	4		2													1						
9	3	4		1													2							
10		4												1				3				2		
13	3		4										1						2					
14	3							1			2													
21			2				1																	
35	3				2									1										
36	2		3																	1				
7						1			2															
17				1	2	3																		
20						3				2							1							
29						1																		
31				1		2															3			
8							1	2																3
11							2	3			4					1					5			
16					2			3	4														1	
28							1												2					
4										2		3											1	
5											3	4		1									2	
18										1		2												
27	4									1	2	3												
37											1									2				
6														1	2		3							
12		6			4								1	2	3						5			
22										4			1	2				3						
33		4		1									2		3									
39															2					3			1	
40	3												1		2									
15																1		2			3			
23																1		2						
24										3						1	2							
2																			1		2			3
19				1																3	4	2		
25								2											1		3			
32			3																1	2				4
26											4						3			5		1	2	
30									4			3										1	2	
34																1							2	3
38							1															2	3	

differentiate solutions and accelerate the search, and the exploitation of ideas taken from successful ACO implementations in other areas.

As a result, the proposed algorithm is fast and reliable. It produces most promising results for medium to large-size problems, with the minimum set of parameters and a simple structure. The addition of recent enhancements in the ACO framework should produce even better results. This is an area of further investigation, together with the most proper way to add other parameters for the manufacturing cells design problem.

References

1. Hyer NL (1984) The potential of group technology for U.S. manufacturing. J Oper Mgmt 4(3): 183–202
2. Wemmerlov U, Hyer NL (1987) Research issues in cellular manufacturing. Int J Prod Res 25(3): 413–431
3. Singh N, Rajamani D (1996) Cellular Manufacturing Systems. Design, Planning and Control. Chapman & Hall, London
4. Da Silveira G (1999) A methodology of implementation of cellular manufacturing. Int J Prod Res 37(2): 467–479
5. Wemmerlov U, Hyer NL (1989) Cellular manufacturing in the U.S. industry: A survey of users. Int J Prod Res 27(9): 1511–1530
6. Cheng CH (1992) Algorithms for grouping machine groups in group technology. OMEGA Int J Mgmt Sci 20(4): 493–501
7. Cheng CH, Goh CH, Lee A (1996) Solving the generalized machine assignment problem in group technology. J Oper Res Soc 47: 794–802
8. Kandiller L (1994) A comparative study of cell formation in cellular manufacturing systems. Int J Prod Res 32(10): 2395–2429
9. Sofianopoulou S (1997) Application of simulated annealing to a linear model for the formation of machine cells in group technology. Int J Prod Res 35(2): 501–511
10. Boctor FF (1996) The minimum-cost, machine-part cell formation problem. Int J Prod Res 34(4): 1045–1063
11. Heragu SS (1994) Group technology and cellular manufacturing. Trans Syst Man and Cybernetics 24: 203–215
12. Chen JS, Heragu SS (1999) Stepwise decomposition approaches for large scale formation problems. Eur J Oper Res 113: 64 –79
13. Heragu SS, Chen JS (1998) Optimal solution of cellular manufacturing system design: Bender 's decomposition approach. Eur J Oper Res 107: 175 –192
14. Spiliopoulos K, Sofianopoulou S (1998) An optimal tree search method for the manufacturing systems cell formation problem. Eur J Oper Res 105: 537–551
15. Sofianopoulou S (1999) Manufacturing cells design with alternative process plans and/or replicate machines. Int J Prod Res 37(3): 707–720
16. Lozano S, Adenso-Díaz B, Eguia I, Onieva L (1999) A one-step tabu search algorithm for manufacturing cell design. J Oper Res Soc 50: 509–516
17. Spiliopoulos K, Sofianopoulou S (2003) Designing manufacturing cells: a staged approach and a tabu search algorithm. Int J Prod Res 41(11): 2531 –2546
18. Cheng CH, Gupta YP, Lee WH, Wong KF (1998) A TSP-based heuristic for forming machine groups and part families. Int J Prod Res 36(5): 1325–1337
19. Dimopoulos C, Mort N (2001) A hierarchical clustering methodology based on genetic programming for the solution of simple cell-formation problems. Int J Prod Res 39(1): 1–19
20. Chen SJ, Cheng CS (1995) A neural network-based cell formation algorithm in cellular manufacturing. Int J Prod Res 33: 293–318
21. Venugopal V, Narendran TT (1994) Machine cell formation through neural network model. Int J Prod Res 32: 2105–2116
22. Solimanpur M, Vrat P, Shankar R (2004) Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing. Eur J Oper Res 157: 592–606
23. Islier AA (2005) Group technology by an ant system algorithm. Int J Prod Res 43(5): 913–932
24. Sofianopoulou S (1996) Part and machine partitioning problem in cellular manufacturing: multiple machine environment. Yugoslav J Oper Res 6: 183–191
25. Harhalakis G, Nagi R, Proth JM (1990) An efficient heuristic in manufacturing cell formation for group technology applications. Int J Prod Res 28: 185–198
26. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York
27. Dorigo M, Stützle T (2004) Ant Colony Optimization. Bradford Books, MIT Press, Cambridge, Massachusetts
28. Maniezzo V, Gambardella LM, De Luigi F (2004) Ant Colony Optimization. In: Onwubolu GC, Babu BV (eds). New Optimization Techniques in Engineering, Springer-Verlag, Berlin, Heidelberg, pp 101–117
29. Maniezzo V (1999) Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS J Comp 11: 358–369
30. Schramm H, Zowe J (1992) A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. SIAM J Opt 2(1): 121–152
31. Rendl F, Wolkowicz H (1995) A projection technique for partitioning the nodes of a graph. Ann Op Res 58: 155–179
32. Falkner J, Rendl F, Wolkowicz H (1994) A computational study of graph partitioning. Math Prog 66: 211–239
33. Carrie AS (1973) Numerical taxonomy applied to group technology and plant layout. Int J Prod Res 4: 399– 416
34. Boe WJ, Cheng CH (1991) A close neighbour algorithm for designing cellular manufacturing systems. Int J Prod Res 29(10): 2097–2116
35. Stanfel LE (1985) Machine clustering for economic production. Eng Costs Prod Econ 9: 73–81
36. McCormick WT, Schweitzer RJ, White TW (1972) Problem decomposition and data reorganisation by clustering techniques. Op Res 20: 993–1009
37. Srinivasan G (1994) A clustering algorithm for machine cell formation in group technology using minimum spanning trees. Int J Prod Res 32(9): 2149–2158
38. Vannelli A, Hall RG (1993) An eigenvector solution methodology for finding part-machine families. Int J Prod Res 31(2): 325–349
39. Goncalves JF, Resende MGC (2004) An evolutionary algorithm for manufacturing cell formation. Comp Ind Eng 47: 247–273
40. Kumar KR, Chandrasekharan MP (1990) Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. Int J Prod Res 28(2): 233–243