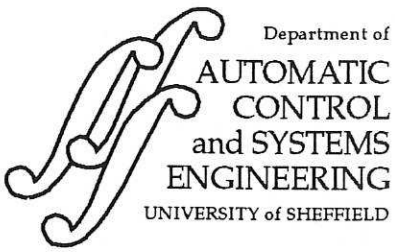

An Evolutionary Computation Approach to a Numerical Cell Formation Problem

Christos Dimopoulos, Ali Zalzala and Jonathan Love

Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield S1 3JD, United Kingdom

Research Report #686
September 1997

An Evolutionary Computation Approach to a Numerical Cell Formation Problem

Christos Dimopoulos, Ali Zalzala and Jonathan Love

Department of Automatic Control and Systems Engineering

University of Sheffield, Mappin Street, Sheffield S1 3JD, UK.

(*Email:rrg@sheffield.ac.uk*)

Abstract

This paper examines a cellular manufacturing optimisation problem in a new facility of a pharmaceutical company. The new facility, together with the old one, should be adequate to handle current and future production requirements. The aim of this paper is to investigate the potential use of evolutionary computation in order to find the optimum configuration of the cells in the facility. The objective is to maximise the total number of batches processed per year in the facility. In addition, a two - objective optimisation search was implemented, using several evolutionary computation methods. The additional objective is to minimise the overall cost. The cost is proportional to the number of cells in the facility. The multi-objective optimisation programs were based on three approaches: The weighted - sum approach, the Pareto-optimality approach, and the Multiobjective Genetic Algorithm (MOGA) approach. Finally, a local search program was implemented, based on the MOGA approach.

Keywords: cellular manufacturing, multi-objective optimisation, evolutionary computation.

1. Introduction

Cellular manufacturing is an application of group technology to manufacturing optimisation problems (Wemmerlov and Hyer, 1989). The aim in cellular manufacturing is to divide the plant in a certain number of cells. Each cell contains machines that process similar type of products. The application of cellular manufacturing in a plant minimises makespan, reduces the set - up time of the machines, and improves the quality of the products (Singh, 1989). There are two major optimisation problems associated with cellular manufacturing:

- the cell - formation problem
- the cell - layout problem

In the case study that we are discussing in this paper, the aim is to find the best configuration of cells in order to maximise the total number of batches processed in the plant per year. The distinct characteristic of this case is that there is only one type of machinery in the plant, the reactor. However, reactors are grouped in cells due to the cross - contamination of products. Only one batch can be processed at a time in a group of reactors that stand close together. We define this optimisation problem as a 'numerical' cell - formation problem, due to its distinctive nature.

Evolutionary programming was used as a guide in our search for the optimum solution. The algorithm tests a population of potential solutions in parallel, in order to find the best configuration of cells. Domain knowledge was incorporated both to the genetic representation of the solutions and the design of the genetic operators.

The trend in manufacturing optimisation is to consider the reduction of cost as one of the most significant objectives. Using the traditional optimisation methods, it is very difficult to incorporate more than one objective in the optimisation process. Therefore, cost is either considered separately, or not considered at all.

Evolutionary computation provides the means of implementing multi-objective optimisation in an easy and efficient way. We

have performed multi-objective optimisation for this case study, using three different approaches: The weighted - sum approach, the Pareto - optimality approach, and the Multiobjective Genetic Algorithm (MOGA) approach. The minimisation of cost was used as a second objective in the multi-objective optimisation search. By combining partial preference information in the form of a goal vector, with the Pareto - optimality approach, a local search was performed in certain areas of the solutions' space.

2. Cell - Formation Problem

Numerous researchers have extensively discussed the cell - formation problem. Several methods have been proposed for the determination of the type and the number of machines that should form each cell.

The objective in this problem is to identify machine families that process similar parts and to group these families into cells. The traditional approach to this problem is the selection of cells by direct observation, which is generally possible for the simplest cases. Although this is a very old method and has obvious limitations, it is still used by companies throughout the world.

Another way to determine the configuration of cells, is to code components according to their features. Families of components are then formed, according to the similarity of their code. Each family determines a group of machines that will form a cell. There are various coding and classification methods that have been proposed (Bennett, 1986). Each of them uses different features or combination of these features. The main drawback of these methods is that they do not divide the plant directly into cells. The components are grouped in easily identified families, and this data is used as a guide for the cell - formation. However, the actual machines that will form each cell are derived directly from the data.

The most popular method for solving the cell - formation problem is the Production Flow Analysis (PFA) method (Burbidge,1975). PFA is a technique that assumes that the physical shape of the components is less important than the route that they have to follow in order to be manufactured. PFA is mainly concerned with manufacturing methods, and the aim is to identify family parts that follow similar routes during their processing. Any group of machines that process a part-family will form an independent cell. The data required for PFA is contained in the process route cards and in the list of machines.

The classic PFA method is inefficient for large problems, due to the fact that the grouping of parts into families is implemented manually. A number of alternative methods based on PFA have been proposed. Array Based Clustering uses the part - incidence matrix to identify the potential cells of the plant. This matrix contains all the information about the route that each part has to follow in order to be processed. The machines and the parts are grouped into families, by performing a series of row and column manipulations. Rank - Order Clustering (King, 1980) uses a slightly different way for the implementation of array clustering, by assigning binary weights in each row and column. Single Linkage Cluster Analysis (Mcauley,1972) is a method which seeks to find a measure of similarity between machines, tools, and every other feature of production. The part - families are then formed, based on this similarity. Mathematical programming methods, like the one proposed by Boctor (1991), address the formation of cells as an optimisation problem, where the objective is to maximise the total sum of similarities between each pair of components.

Due to the nature of the cell - formation problem, many artificial intelligence methods have been proposed for the search of the optimum configuration. Kao and Moon (1991) used the Artificial Neural Network (ANN) to form part families based on design features. Fuzzy logic has also been applied by Xu and Wang (1989).

In recent years there is a growing interest in the use of Genetic Algorithms for this optimisation problem. Modified Genetic Algorithms, like the one proposed by Falkenauer (1993), use problem - specific chromosome representation and purpose - based genetic operators to determine the

formation of cells. Unlike classic GAs, these algorithms proved to be very effective in finding the best configuration of cells, because they incorporate domain knowledge in their search for the optimal solution.

3. Evolutionary Computation Approach to Manufacturing Optimisation

In this paper we introduce a method for solving numerical cell - formation problems, based on the principles on Genetic Algorithms (Holland,1975). We have approached the problem following the procedure proposed by Michalewicz (1991). The algorithm searches for the best configuration of cells in the plant, so that the total number of batches produced per year will be maximised. Cost is also introduced in the problem as a second objective. The algorithm is modified according to different multi-objective optimisation methods, and several alternative solutions are proposed.

3.1 Problem Description

Work load projections on the utilisation of an existing pilot plant facility, at the factory of a pharmaceutical company, indicate that it will not be adequate to handle current and future production requirements. Therefore, the company decided to build a new facility, in order to accommodate their future needs. The company produces a number of different products. The products are classified as in the following example:

PROD 5-4 : product batch that requires 5 days of processing, and occupies 4 reactors while being processed.

Based on prior and projected work knowledge, and statistical data generated in-house by the company, a basis for modelling including 15 products was developed. This basis is presented in a relevant table in the Appendix (Table 1). The constraints of the problem are the following:

$$2 \le a_n \le 6$$

$$2 \le n \le 6$$

$$\sum_{k=1}^{n} a_k \in [12,13]$$

a_n : integer number representing the total number of reactors in the nth cell

n : integer number representing the total number of cells

The company proposes the following scheduling methods:

- A list of products is to be produced over a period of a year. Batches are processed in strict sequence, but there's no restriction in the type of batch that a cell can process. Only one batch can be processed at a time in a cell, due to cross - contamination of the products. This method will be identified from now on as CASE1 scheduling method
- The same list of products is to be produced in a year's time. Batches are again processed in strict sequence, but in this case, each cell processes a certain type of batch. There are small, medium and large-sized batches. A cell can process either small and medium-sized batches or large and medium-sized batches. Only one batch can be processed at a time in a cell, due to cross - contamination of the products. This method will be identified from now on as CASE2 scheduling method

3.2 Chromosome Representation

Many researchers have argued that the binary representation of the solutions is inefficient for a series of optimisation problems (Davis,1987). De Jong (1985) believes that when a search space is best represented by complex structures like arrays, trees, integers, etc., the programmer should not try and linearise them in binary strings. Instead, it would be better to work directly on them.

The modified Genetic Algorithm that is described in this paper, was designed to fit the case study problem. It incorporates domain knowledge not only to the representation of the solutions, but to the design of the genetic

operators as well.

A potential solution of the problem has the following representation:

$$\{a_1, a_2, a_3, \ldots . a_n\}$$

a_n : integer number representing the total number of reactors in the nth cell

n : integer number representing the total number of cells

This representation has no fixed length, a fact that makes the design and the function of the genetic operators a difficult task. On the other hand, this representation has the advantage of carrying both the variables of the problem in one chromosome, one variable being explicit and the other implicit.

3.3 Genetic operators

The regular crossover and mutation operators, produce most of the times illegal offspring when applied to the previous type of chromosomes. Instead of using penalty functions, which are time consuming and inefficient, two purpose-based operators were produced from 'scratch', to fit this particular problem.

3.3.1 Same - Total Operator

This is a crossover - type operator. We will illustrate its operation by using an example. Let's suppose that the two following chromosomes have been selected for genetic alteration:

{ 4 6 2 }

and

{ 2 2 4 4 }

The algorithm performs a serial search in the first chromosome and finds a number or a sum that belongs to the space [4,9]. The algorithm then searches in the second chromosome to find a number or a sum of adjacent genes which is equal to the previous number. The genes that represent the same total in the two chromosomes exchange positions. For our example the operator will function as follows will be:

SEARCH ⇒ { 4 6 2 } GENE: 1

TOTAL: 4

SEARCH ⇒ { 2 2 4 4 } GENES: 1,2

TOTAL: 4

EXCHANGE POSITIONS:

{ 2 2 6 2 }

and

{ 4 4 4 }

None of the chromosomes violates the constraints, while the offspring are totally different from the parents. In some special cases the application of this operator does not alter the shape of the parent chromosomes. The algorithm compensates for these cases by having an increased probability for this particular operator.

3.3.2 Decomposition Operator

This is a mutation - type operator. It was designed taking in account problem - specific information. Preliminary research showed that small-sized cells perform better than the large-sized ones because they are less affected by the cross - contamination factor. Large-sized cells suffer from low utilisation of their reactors, because they cannot process more than one batch at a time.

The idea of the decomposition operator is to guide the search for the optimal solution towards the small-sized cells' area. Therefore, when a large-sized cell is selected for genetic alteration, the following offspring is produced:

SELECTED GENE	RESULT AFTER APPLYING DECOMPOSITION OPERATOR
4	2 2
5	3 2
6	2 2 2

3.4 Fitness Evaluation

The measure of fitness for the solutions of the algorithm, is the total number of batches processed per year in the plant. In order to find this number, we simulate the annual

manufacturing production for each of these solutions.

For the case of multi-objective optimisation, a cost model was constructed, related to the total number of cells in the plant. The cost increases linearly with the number of cells, as the following table shows:

NUMBER OF CELLS	COST
2	2
3	3
4	4
5	5
6	6

3.5 Multiobjective Optimisation methods

The following three methods were used in the case of multi-objective optimisation:

3.5.1 Weighted - Sum Approach

In this approach a weight is given for every objective to be optimised. Each objective's weight determines the importance of the objective. If the objective function for the objective i, is $f_i(x)$, then the overall objective function will be:

$$F(x) = \sum_{i=1}^{k} w_i \cdot f_i(x)$$

where the weights $w_i \in [0,1]$ and $\sum_{i=1}^{k} w_i = 1$

By assigning different weight vectors for the objectives, the algorithm will converge to different solutions.

3.5.2 Pareto - Optimality Approach

The concept of Pareto - Optimality is defined as follows (Morad,1997):

Let's consider the vector optimisation problem:

$$\min_{x \in X} f(x) = \min_{x \in X} \{f_1(x), f_2(x), \ldots, f_m(x)\}$$

where:

$x = n$ dimensional vector of decision variables

X = the set of all feasible solutions subject to constraints

A decision vector $x_\mu \in X$ is said to be Pareto-optimal, if and only if there is no other $x_\nu \in X$ such that $f(x_\nu) = (\nu_1, \nu_2, \ldots, \nu_m)$ dominates $f(x_\mu) = (u_1, u_2, \ldots, u_m)$. In other words, there is no x_ν such that:

$$\forall i \in \{1, \ldots, m\}, \nu_i \leq u_i$$

This solution is called a *non-inferior* or *non-dominated* solution. If we use a simple GA for a multi-objective optimisation problem, the first generation will normally evolve a population of solutions. Some of them will be non-dominated, and they will form the so-called **dominant front** of the solutions. All these solutions are considered as rank '1' individuals, and they will be assigned with the same fitness value for the next generation. We then remove from the population the dominant front, and the new dominant front will be the solutions of rank '2'. They will all be assigned with the same fitness value, but of course lower than the rank '1' solutions. This procedure will go on until all the solutions are assigned a fitness value, and then the evaluation of the new population will start.

3.4.3 Multi - Objective Genetic Algorithm (MOGA) Approach

This method was proposed by Fonseca and Fleming (1993). In this scheme the ranking of each individual corresponds to the number of individuals in the current population by which is dominated. In that way, the dominant front of the solutions is assigned the same rank, while the rest of the solutions are assigned a lower ranking according to the population density of the region of solutions that dominate them. The basic advantage of this method is that it can perform local search, by combining Pareto dominance with partial

preference information in the form of a goal vector. In this way, the ranking mechanism can exclude objectives that already satisfy their goals. If fully unattainable goals are specified, then we have the basic Pareto ranking, because no objective will ever be excluded from comparison.

4. Results

The evolutionary approach was tested in single objective and multi-objective optimisation. The results are presented in the following paragraphs.

4.1 Results of the single objective optimisation

The algorithm was tested fifty times, for both CASE1 and CASE2 scheduling methods, using only the same-total operator. Then, a second series of tests were implemented, with the decomposition operator present as well. The values for various parameters of the program can be found in Table 2 (Appendix).

The performance of the algorithm was tested by recording the mean fitness value of each generation. The algorithm converged to a single solution for both CASE1 and CASE2 methods, as Figures 1 & 2 show respectively. The presence of the decomposition operator improves the performance of the algorithm, as the same figures show, because this is a purpose-based operator, constructed to fit this particular problem.

The best solution of each test was recorded, and the results showed that configurations having a large number of small-sized cells, outperform any other solution. These solutions suffer less from low-utilisation of their machines due to the cross-contamination factor. The best solution ever recorded for both cases, is the following:

BEST FACTORY LAYOUT

{ 3 2 2 2 2 2 }

This layout processes 396 batches per year for CASE1 scheduling method, and 436 batches per year for CASE2 scheduling method. A number of alternative solutions were recorded during the evolutionary process. These solutions together with their associated fitness can be found in Tables 3 & 4 in the Appendix.

4.2 Results of the multi-objective optimisation

In the weighted - sum approach, several different weights were given to the objectives of the optimisation for both scheduling methods. Some examples of the results are given in Figures 3, 4, 5 & 6.

The technique of fitness sharing was used in the Pareto - Optimality approach, to prevent the

FITNESS RESULTS
CASE1:

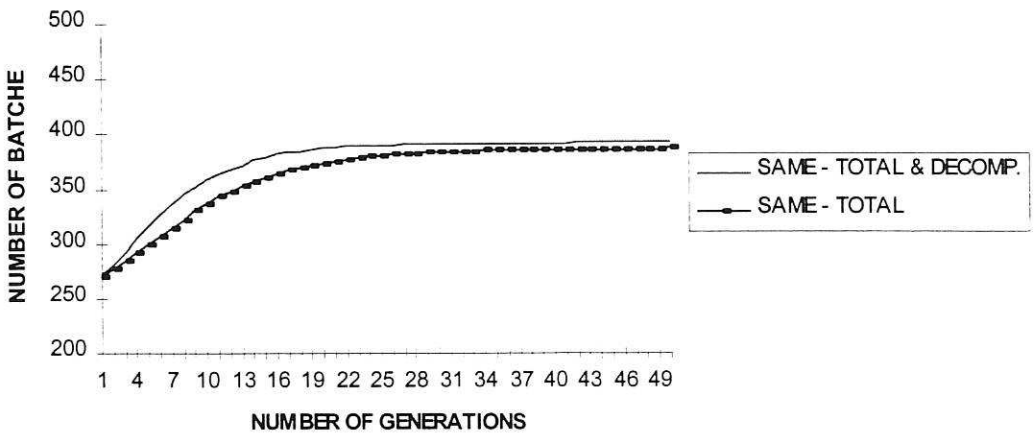

Figure 1 : CASE1 scheduling method

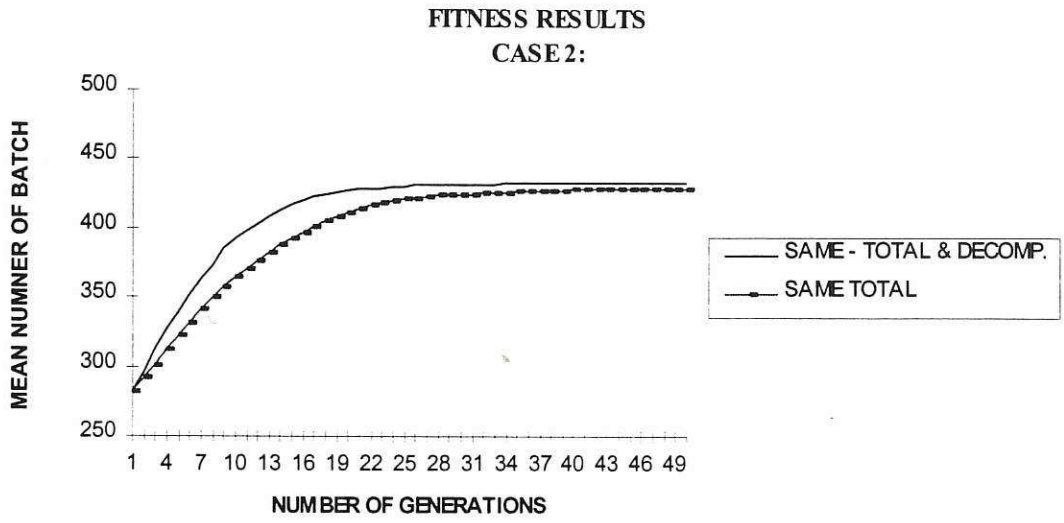

Figure 2 : CASE2 scheduling method

premature convergence of the algorithm to a single solution (Goldberg,1987). This technique leads to the formation of stable sub-populations (species) of solutions. The results are presented in Figures 7 & 8.

The same technique was used in the case of Multi - Objective Genetic Algorithm (MOGA) approach. The algorithm converged to a number of alternative solutions that are presented in Figures 9 & 10. The same algorithm was modified in order to be able to perform local search in the solutions' search space. Results were obtained using different goal vectors. A graphic presentation of some results is given in Figures 11 & 12, while the numerical analysis of the same results is given in the Appendix (Tables 3 & 4).

4.3 Comparisons

Considering the previous results, we can say that CASE2 scheduling method outperform CASE1, because of the grouping of products and cells in certain families. A comparison of the two scheduling methods in terms of their fitness is given in Figure 13.

There are no significant differences in the results of the three multi-objective optimisation approaches. The entire algorithm seems to converge more or less to the same population of alternative solutions.

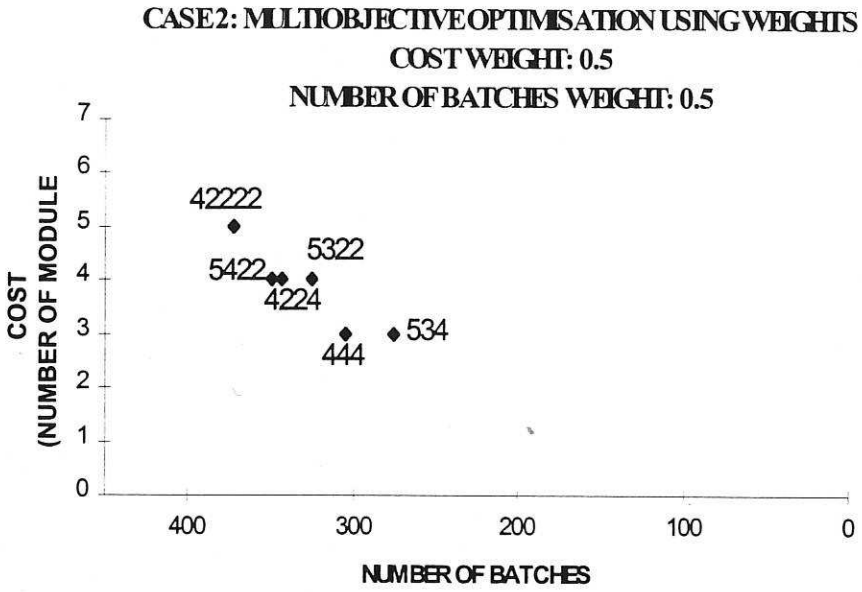

Figure 3 : Weighted - sum approach

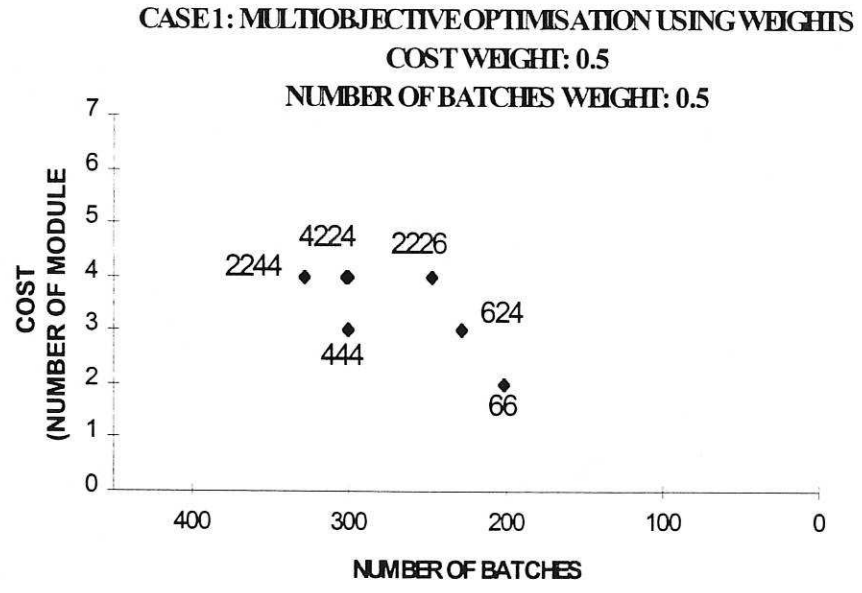

Figure 4 : Weighted - sum approach

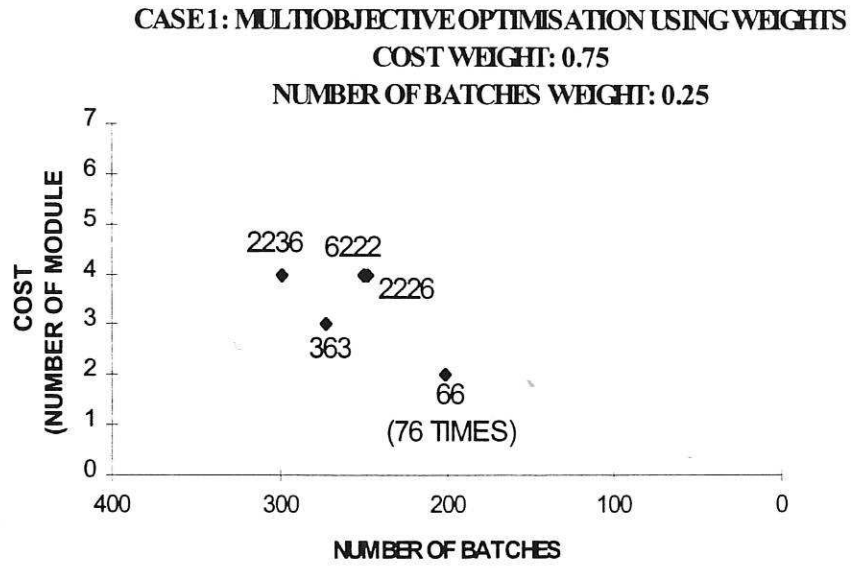

Figure 5 : Weighted - sum approach

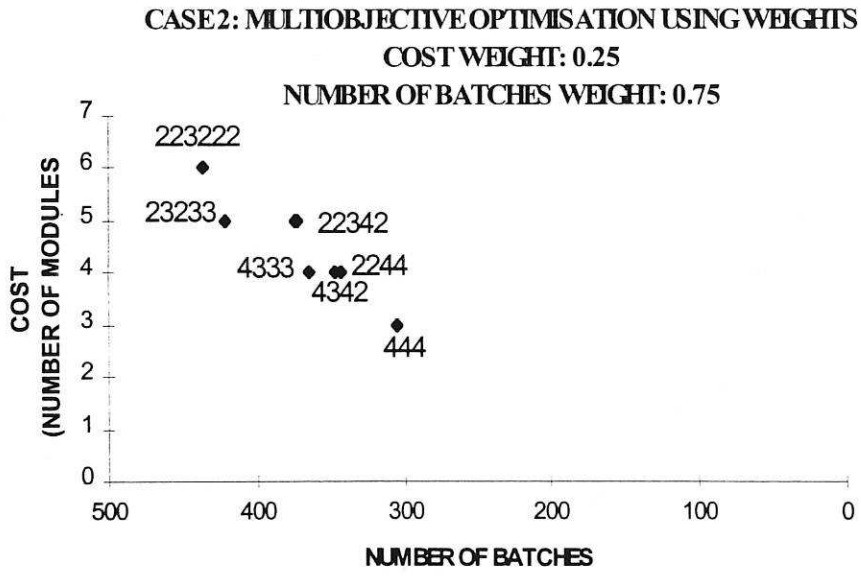

Figure 6 : Weighted - sum approach

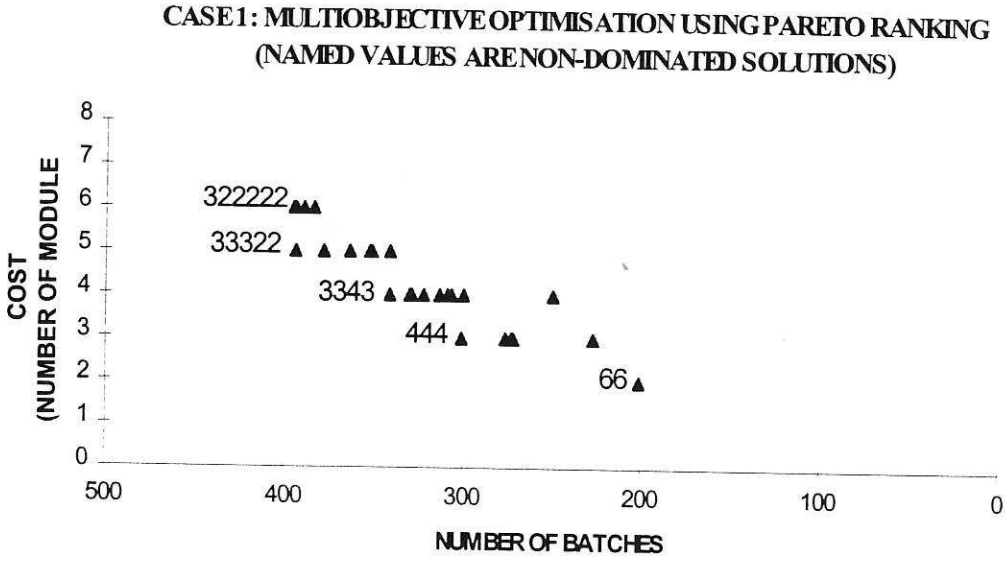

Figure 7 : Pareto -Optimality approach

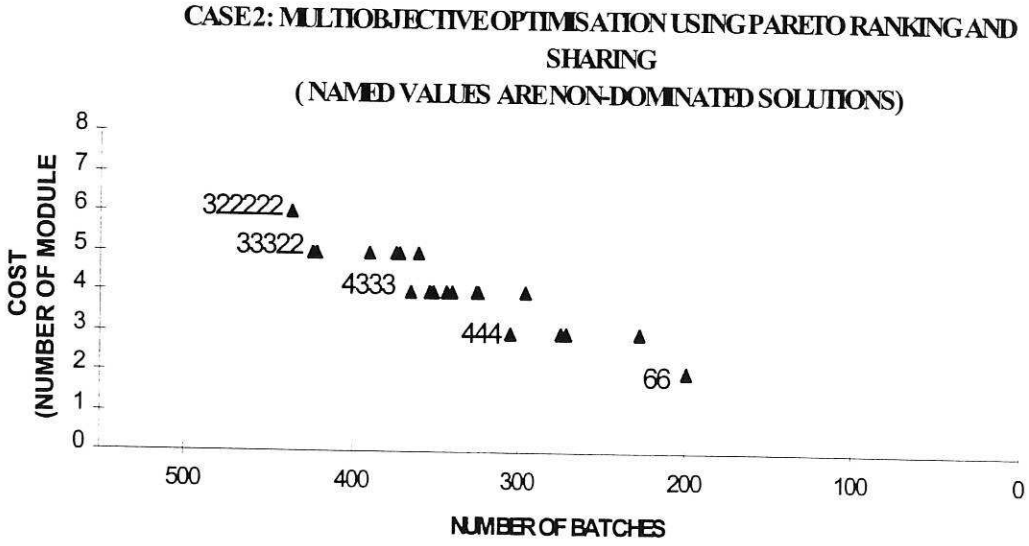

Figure 8 : Pareto -Optimality approach

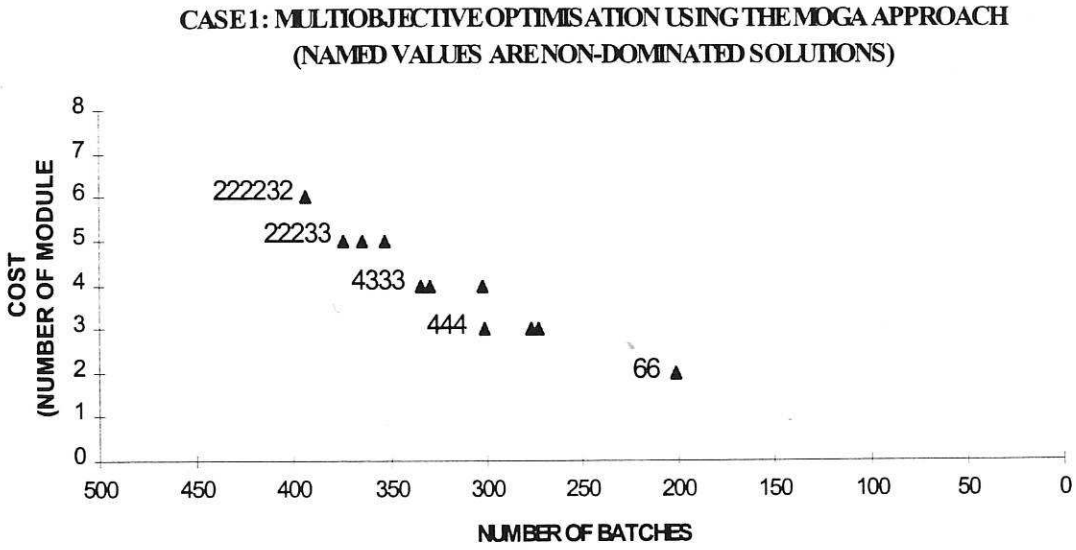

Figure 9 : MOGA approach

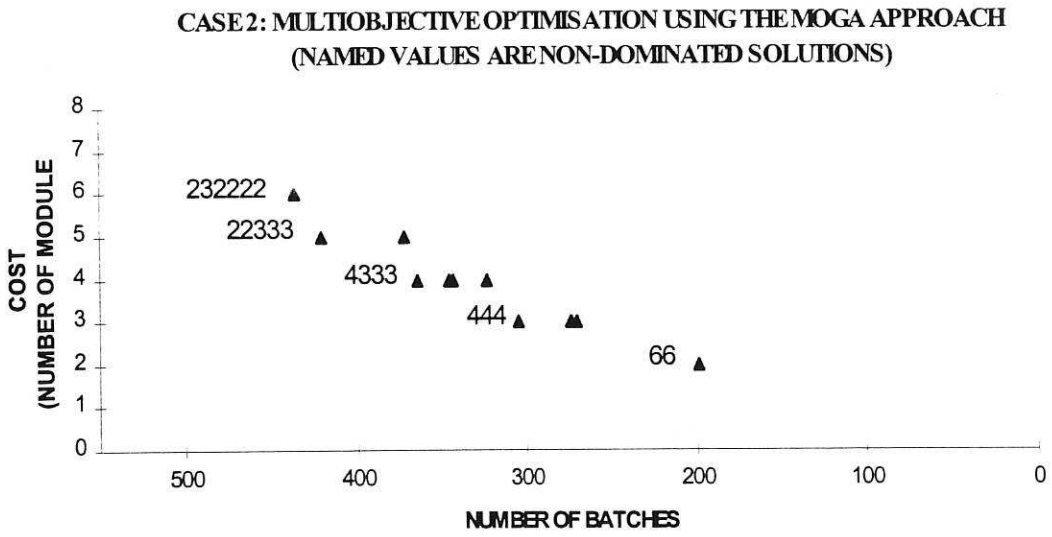

Figure 10 : MOGA approach

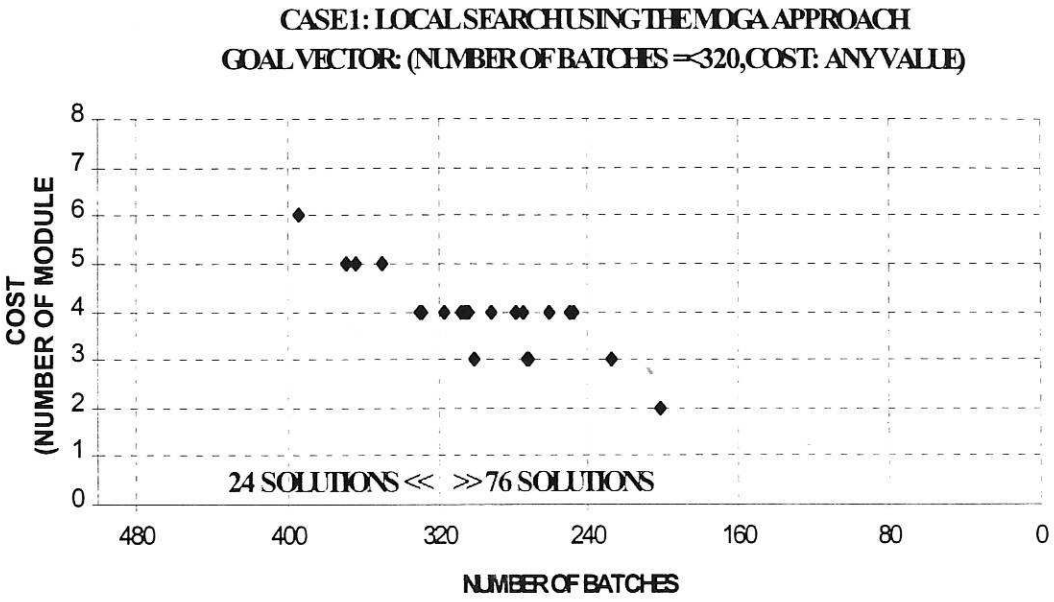

Figure 11 : Local Search using the MOGA approach

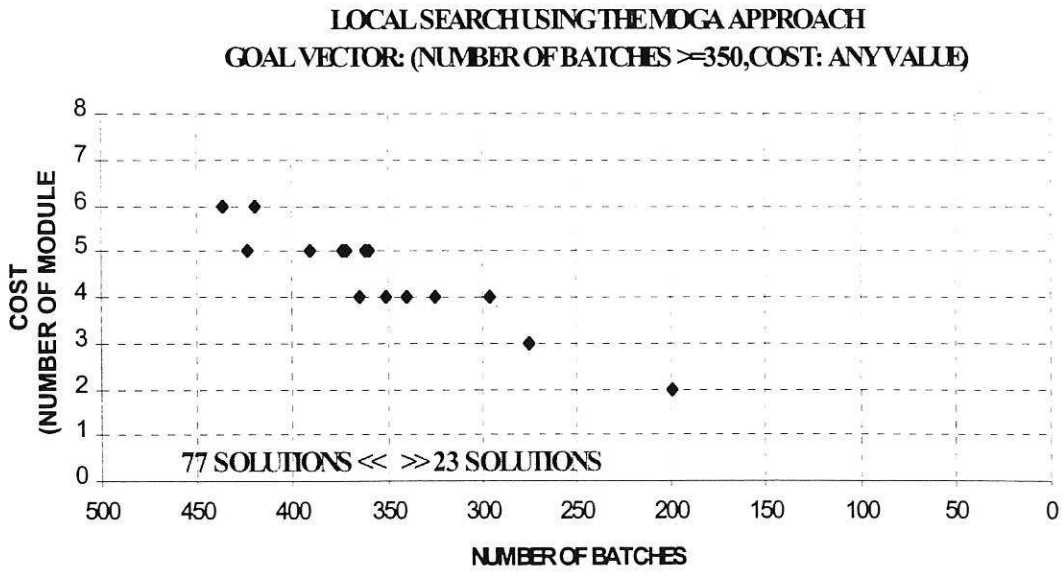

Figure 12 : Local Search using the MOGA approach

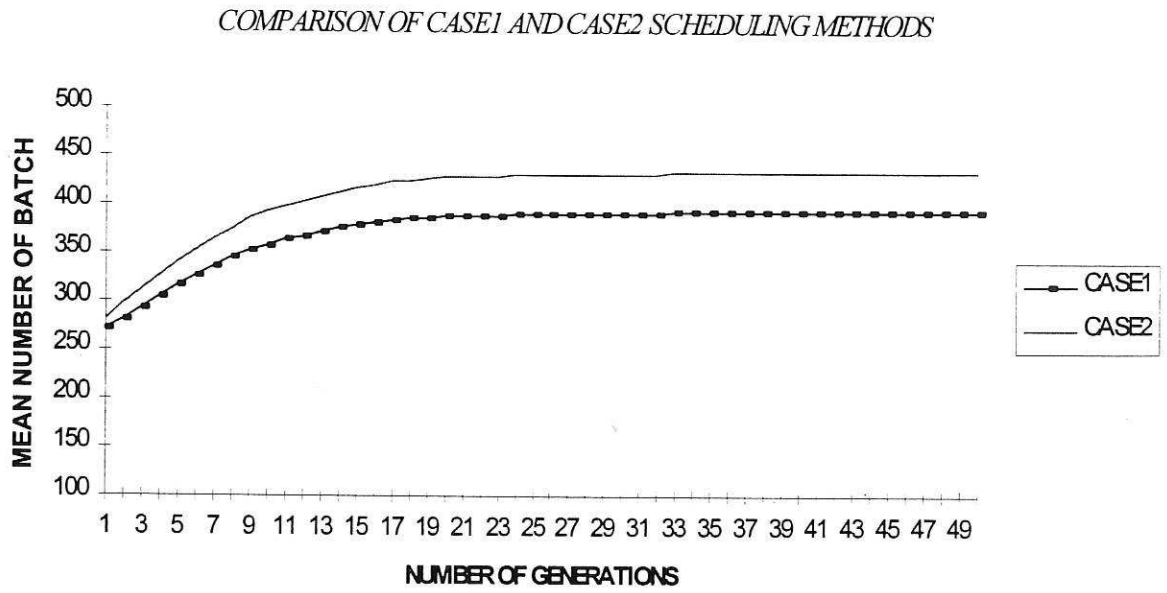

Figure 13 : Comparison of scheduling methods

5. Discussions

Results showed that the evolutionary approach to this numerical cell formation problem outperformed the traditional approach in terms of efficiency and number of alternative solutions produced, especially if purpose-based operators and chromosome representation is used. In addition, the evolutionary approach to multi-objective optimisation provides the means of incorporating more than one objective to the optimisation process. Different multi-objective optimisation methods were used, and all performed equally well.

6. Conclusions

Most of the manufacturing optimisation problems cannot be solved easily and efficiently using the traditional optimisation methods. This paper describes an evolutionary algorithm designed to solve a numerical cell formation problem. Results showed that when the plant is divided to a large number of small-sized cells, the total number of batches processed in the plant per year is increased. In addition, if these cells are grouped according to the type of the products that they process, the performance of the plant is improved furthermore. Multiobjective optimisation is a major consideration in a manufacturing plant, since the cost factor is a critical issue in every aspect of the production. Evolutionary algorithms provide the means of implementing multi-objective optimisation, using a number of different approaches. Local search can also be implemented in a subset of the solutions' search space.

References

1. Bennett, D., *Production Systems Design*, Butterworths, 1986.
2. Boctor, F. 'A linear formulation of the machine-part cell formation problem', *International Journal of Production Research*, Vol 29, part 2, pp 343-356, 1991.
3. Burbidge, J.L., *The Introduction of Group Technology*, Heineman, London, 1975.
4. Davis, L., and Steenstrup, M. *Genetic Algorithms and Simulated Annealing: An Overview,* Genetic Algorithms and

Simulated Annealing, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

5. De Jong, K.A., Genetic Algorithms, *A 10 Year Perspective*, Proceedings of the first International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.
6. Falkenauer, E., 'New representation and operators for GAs applied to Grouping problems', *Research Report No CP 106-P4*, Research Centre for Belgian Metalworking Industries, 1993.
7. Fonseca, C.M., and Fleming P.J., *Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalisation*, Proceedings of the fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
8. Goldberg, D.A. and Richardson, J., *Genetic Algorithms with Sharing for Multimodal Function Optimisation*, Proceedings of the second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
9. Goldberg, D.E., *Genetic Algorithms in Search, Optimisation, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
10. Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.
11. Kao, Y. and Moon, Y.B., 'A unified group technology implementation using the back propagation learning rule of neural networks', *Computers and Industrial Engineering*, Vol 20, No 4, pp 425-437, 1991.
12. King, J.R., 'Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm', *International Journal of Production Research*, Vol 18, No 2, pp 213-232, 1980.
13. McAuley, J.,'Machine Grouping for Efficient Production Production', *Production Engineer*, pp 53-57, 1972.
14. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, (second edition), Springer-Verlag, 1994.
15. Morad, N., *Integrated production planning and scheduling in cellular manufacturing using Genetic Algorithms*, (Doctoral Dissertation), University of Sheffield, Dept. of Automatic Control and Systems Engineering, Sheffield, 1997.
16. Singh, N., *Computer Integrated Design and Manufacturing*, Wiley, 1996.
17. Wemmerlov, U. and Hyer, N.L., *Cellular Manufacturing in the U.S. Industry: A Survey of Users*, International Journal of Production Research, 27(9), 1989.
18. Xu, H., and Wang, H.P., 'Part family formation for GT applications based on fuzzy mathematics', *International Journal of Production Research*, Vol 27, No. 9, pp 1637-1651, 1989.

Appendix :

PRODUCT TYPES	BATCH TIME (DAYS)	NUMBERS OF REACTORS USED	OVERALL % OF PRODUCT MADE	PREPARATION TIME TO BE ALLOWED
PROD 5-4	5	4	4%	0.33 DAYS
PROD 4-4	4	4	4%	»
PROD 3-4	3	4	4%	»
PROD 2-4	2	4	4%	»
PROD 5-3	5	3	11%	»
PROD 4-3	4	3	4%	»
PROD 3-3	3	3	4%	»
PROD 2-3	2	3	21%	»
PROD 5-2	5	2	11%	»
PROD 4-2	4	2	6%	»
PROD 3-2	3	2	11%	»
PROD 2-2	2	2	6%	»
PROD 3-1	3	1	6%	»
PROD 2-1	2	1	2%	»
PROD 1-1	1	1	2%	»

Table 1 : Basis for modelling

VARIABLE	SIGNIFICANCE	VALUE
N_OF_GENERATIONS	Number of generations	50
N_OF_GENES	Maximum number of cells in the plant	6
PROB_OF_CROSSOVER	Prob. of a chromosome to be selected for crossover	0.4
PROB_OF_MUTATION	Prob. of a gene to be selected for mutation	0.05
POP_SIZE	Number of chromosomes in each generation	100
UP_BOUND	Maximum number of reactors in a cell	6
L_BOUND	Minimum number of reactors in a cell	2

Table 2 : Values for various parameters

Solution	Cost	Batches	Solution	Cost	Batches	Solution	Cost	Batches	Solution	Cost	Batches
222232	6	394	2442	4	317	3262	4	279	2226	4	247
222232	6	394	2442	4	317	2325	4	275	2226	4	247
222232	6	394	2442	4	317	2325	4	275	2226	4	247
222232	6	394	2442	4	317	2325	4	275	2226	4	247
222232	6	394	2442	4	317	2325	4	275	642	3	227
222232	6	394	3342	4	308	2325	4	275	642	3	227
222232	6	394	3342	4	308	633	3	273	642	3	227
222232	6	394	3342	4	308	633	3	273	642	3	227
32332	5	369	3342	4	308	633	3	273	642	3	227
32332	5	369	4233	4	306	633	3	273	426	3	227
32332	5	369	4233	4	306	633	3	273	426	3	227
32332	5	369	4233	4	306	336	3	272	426	3	227
33232	5	364	4233	4	306	336	3	272	426	3	227
33232	5	364	4233	4	306	336	3	272	426	3	227
22242	5	350	2433	4	305	336	3	272	426	3	227
22242	5	350	3225	4	304	336	3	272	426	3	227
3333	4	330	444	3	301	4242	4	261	66	2	201
3333	4	330	444	3	301	4242	4	261	66	2	201
3333	4	330	444	3	301	4242	4	261	66	2	201
3333	4	330	444	3	301	4242	4	261	66	2	201
3333	4	330	444	3	301	4242	4	261	66	2	201
3333	4	330	6232	4	292	4242	4	261	66	2	201
3333	4	330	6232	4	292	6222	4	250	66	2	201
2244	4	329	6232	4	292	6222	4	250	66	2	201
2442	4	317	3262	4	279	2226	4	247	66	2	201

Table 3 : MOGA approach, local search (low number of batches, CASE1)

Solution	Cost	Batches	Solution	Cost	Batches	Solution	Cost	Batches	Solution	Cost	Batches
222322	6	436	222222	6	419	3334	4	365	4432	4	351
222322	6	436	32332	5	390	3334	4	365	4432	4	351
222322	6	436	23332	5	390	33222	5	362	4234	4	340
222322	6	436	32332	5	390	33222	5	362	6322	4	325
222322	6	436	23332	5	390	33222	5	362	6322	4	325
222322	6	436	23332	5	390	33222	5	362	6222	4	296
222322	6	436	22432	5	374	23322	5	360	6222	4	296
222322	6	436	22432	5	374	23322	5	360	2226	4	296
33322	5	423	42222	5	372	23322	5	360	6222	4	296
33322	5	423	22423	5	372	23322	5	360	6222	4	296
33322	5	423	42222	5	372	23322	5	360	634	3	275
33322	5	423	42322	5	372	23322	5	360	634	3	275
33322	5	423	42222	5	372	4432	4	351	634	3	275
33322	5	423	42222	5	372	4432	4	351	634	3	275
33322	5	423	42322	5	372	4423	4	351	634	3	275
222222	6	419	22234	5	372	4432	4	351	634	3	275
222222	6	419	22234	5	372	4432	4	351	634	3	275
222222	6	419	42322	5	372	4432	4	351	634	3	275
222222	6	419	42222	5	372	4432	4	351	634	3	275
222222	6	419	22234	5	372	4432	4	351	66	2	199
222222	6	419	3334	4	365	4432	4	351	66	2	199
222222	6	419	3334	4	365	4432	4	351	66	2	199
222222	6	419	3334	4	365	4432	4	351	66	2	199
222222	6	419	3334	4	365	4432	4	351	66	2	199
222222	6	419	3334	4	365	4432	4	351	66	2	199

Table 4 : MOGA approach, local search (high number of batches, CASE2)