Assessing the Performance of Multiobjective Genetic Algorithms for Optimization of a Batch Process Scheduling Problem

K. J. Shaw, A. L. Nortcliffe, M. Thompson, J. Love, P. J. Fleming.

Department of Automatic Control & Systems Engineering,
University of Sheffield, Mappin Street, S1 3JD, UK.
Email: shaw@acse.shef.ac.uk;

{a.l.nortcliffe | m.thompson | j.love | p.fleming}@shef.ac.uk

C. M. Fonseca

Instituto de Sistemas e Robótica, Portugal,
and ADEEC/UCEH, Universidade do Algarve,
Campus de Gambelas, 8000 Faro, Portugal
Email: cmfonsec@ualg.pt

Abstract - Scheduling optimization problems provide much potential for innovative solutions by genetic algorithms. The complexities, constraints and practicalities of the scheduling process motivate the development of genetic algorithm (GA) techniques to allow innovative and flexible scheduling solutions.

Multiobjective genetic algorithms (MOGAs) extend the standard evolutionary-based genetic algorithm optimization technique to allow individual treatment of several objectives simultaneously. This allows the user to attempt to optimize several conflicting objectives, and to explore the trade-offs, conflicts and constraints inherent in this process. The area of MOGA performance assessment and comparison is a relatively new field, as much research concentrates on applications rather than the theory. However, the theoretical exploration of MOGA performance can have tangible effects on the development of highly practical applications, such as the process plant scheduling system under development in this work. By assessing and comparing the strengths, variations and limitations of the developing MOGA using a quantitative method, a highly efficient MOGA can develop to suit the application. The user can also gain insight into behaviour the application itself.

In this work, four MOGAs are implemented to solve a process scheduling optimization problem; using two and five objectives, and two schedule building rules. A quantitative comparison of their performances is conducted, and the results and implications of this comparison are then examined within the context of the problem.

1. Introduction

Scheduling problems have provided much potential for innovative solutions by genetic algorithms over the past decade, (e.g., Davis, 1985; Bagchi et, al, 1991; Cleveland and Smith, 1993; Lee et al., 1993; Shaw and Fleming, 1996; Löhl et al., 1998). This work explores the use of a schedule optimization tool which employs the properties of the multiobjective genetic algorithm for a complex batch scheduling problem. This tool aims to allow flexible, interactive and informed schedule optimization, for typical problems found within the process scheduling industry. Shaw & Fleming, (1997), suggested that manufacturers might use MOGA techniques for discrete event scheduling problems, to define or prioritise their optimization goals interactively, according to the changing demands of the system. In batch process plant scheduling, the additional demands include a need for batch sizing, continuous flow of certain elements, and a choice of trains or routes through the plant.

Besides optimizing the multiple objectives presented by the plant, the MOGA may be a good way to deal with the various conflicting constraints found within such a problem. The MOGA must be designed to represent the problem concisely but comprehensively, particularly for the two key decision variables of order sequence and batch size. To avoid redundancy of information or inaccuracy of representation, a suitable method of handling four plant objectives must be decided. Their consistent treatment with the fifth, non-plant-defined, objective, that of feasibility, must also be considered. This is discussed in more detail in section 3.

In this paper, four MOGAs are applied to a prototype batch scheduling problem which aims to model the key processes typically found in chemical processing plants. A generic process plant model provides a prototyping system for the scheduling tool, framed in S88 constructs (ISA, 1995), the industry standard for batch scheduling specifications and definitions. The model allows different production scenarios and rules to be applied, giving the user a framework in which to explore the effects of the MOGA application.

0-7803-5536-9/99/$10.00 ©1999 IEEE

2. Problem description

2.1 Literature

Research into the schedule optimization for multi-purpose or multi-product batch plants generally focuses on optimization of one aspect of a series of campaigns. Ku and Karimi (1990) explored the issue of meeting the customer due dates. Grau et al. (1996) used a recursive algorithm to optimize completion times of batches, by reducing various unit operation times. Kim et al. (1996) used a modified GA to determine a multi-product production sequence and path for each batch. Löhl et. al., (1998), compare a GA with the frequently used MINLP method, for a mixed batch / continuous problem. However, in this work, the batch sizes remain constant, as the authors explain; "variable batch sizes would introduce a complexity which cannot currently be handled appropriately". Shaw, Nott and Lee, (1998), compare a GA with MILP on a problem based on a mixed batch/continuous processing plant which includes a choice of discrete batch sizes, and a continuous flow decision variable in addition to the schedule sequencing problem. Both the latter papers present mixed conclusions as to the relative performance of the two methods at this stage.

Research into multi-objective scheduling of multi-product or multi-purpose is minimal. Ku and Karimi (1991) concluded that simulated annealing gave the best results in meeting the tardiness penalties, using the dual objectives of meeting the customer due dates, and reducing storage times. These two objectives are later explored in this work. Stochastic optimization techniques were employed by Grau et al. (1996) to optimize a campaign batch schedule which aimed to meet deliveries and due dates.

GAs are commonly applied to scheduling problems as they offer a series of highly relevant advantages in this area. They are capable of solving NP-complete problems, as is the case with many scheduling problems, and can handle both continuous and discrete functions. The parallel nature of their search method allows them to handle complex or difficult search spaces, to offer a choice of potential solutions from the population at any stage in the optimization process, and allows them to work with .incomplete or inexact data. In addition, their powerful search abilities can be combined with a high level of user-interaction and direction to interact with the search, or focus upon a particular area of interest. For all these reasons, they are a source of interest to schedulers.

Work on multiobjective optimization genetic algorithms for scheduling problems is more commonly found beyond the multistage batch process domain, in a variety of other applications, (e.g., Viennet et al., 1995, Tamaki et al., 1992, 1996; Niemeyer and Shiroma, 1996; Shaw and Fleming, 1996, and Ishibuchi and Murata, 1998). Multiobjective optimization methods continue to offer potential advantages to schedulers requiring genuine multiobjective treatment of their various objectives, and commonly, constraint handling, within their solutions.

2.2 Problem Description

The chemical industry is moving away from high volume continuous production of low cost products to low volume, specialist chemicals of high value produced by batch operations. The latter involves complex chemistry, process operations and results in small yields. To make the manufacture of such chemicals more cost effective and efficient, multi-purpose batch plants are used. These plants are designed to be generic, to process a broad range of products and variety of operations. The plants are designed to accommodate many process operations and simultaneously operations, but there are bottlenecks where streams merge to single resource or different products call for the same process unit. Careful scheduling is required to ensure that all the resources are used effectively and efficiently.

S88.01 (ISA, 1995) and IEC 61512-1 (IEC, 1997) standards provides guidelines to designing batch processes, control strategies and schedules for such plants. This standardises the framework of terminology, models and structures to be used to describe and articulate the scheduling requirements of a multi-purpose batch plant (Love and Bunch, 1998). The proposed GA-based scheduling tool is to schedule a multi-purpose batch plant expressed in S88 constructs, i.e. the overall plant is a process cell consisting of units and equipment modules, see Figure 1.

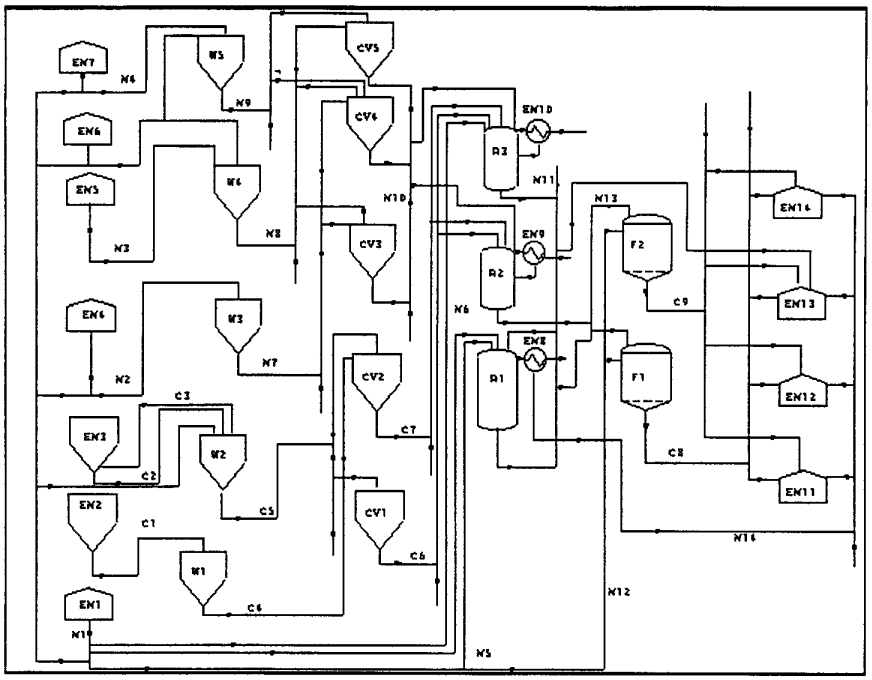

Figure 1- Process cell of the generic multi-purpose batch plant

For purpose of this paper, a simpler multi-purpose batch plant is used to provide an initial test problem for the GA implementation scheduling tool, as shown in Figure 2.

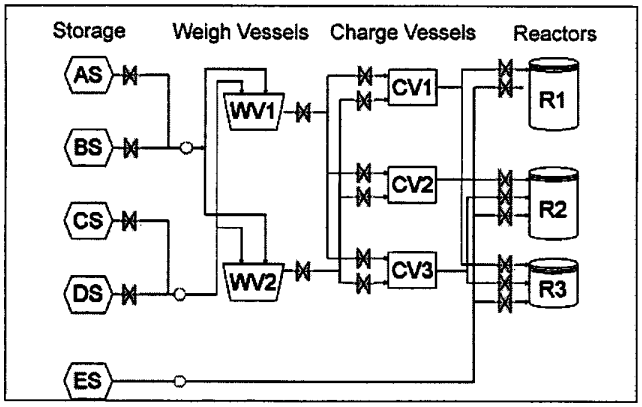

Figure 2 - Simple model of multi-purpose batch plant

More details of the schedule optimization problem designed for this plant are given in 3.2.

3. Using Genetic Algorithms for the Batch Scheduling Problem

The search for an effective representation for all the complexities of scheduling problems is a common area of research. Due to the difficulties of finding a concise and accurate representation of including all relevant data within the GA representation, the notion of infeasible individuals commonly appears in scheduling problem representation, as it does in many other constrained applications. 'Infeasible' in this sense means that the individuals represent some part of the solution but that one element may be invalid in some way and therefore invalid for the final solution. In this example, individuals are allowed in the population which translate into schedules that do not satisfy the batching constraint; that the batch size decisions made must allow the exact completion of the required

customer orders. Such individuals are therefore called 'infeasible'.

Infeasible individuals are not automatically disqualified from the population, nor are they repaired to create feasible individuals. Instead, the amount of infeasbility is defined as an additional objective that must be minimised. The MOGA is used to deal with this infeasibility as an objective. Additional justification for treating the infeasibility in this way is provided in Shaw, et. al., 1999 , which indicates that the use of MOGA in this way is a promising approach. This approach has been suggested (Chu & Beasley, 1992; Surry, Radcliffe and Boyd, 1995), as a method of avoiding the process of defining a more complex, but constantly feasible representation, specifically designed operators, or a repair method (Michalewicz, 1997).

3.1 MOGA

Multiobjective genetic algorithms (MOGAs) allow the solution of problems in which more than one objective defines the optimization goal. The result of the optimization process is a set of possible solutions to a problem, all optimal in a multiobjective sense, but which cover the range of varying emphases on the objectives. Genetic algorithms have been shown to be effective methods for allowing multi-objective optimization, particularly for optimization of real-life applications that have proved demanding previously. To avoid the 'aggregation' method of simply adding the various objectives in a form of weighted sum - a method which includes all the objectives but may not treat them in an adequately separate manner, several other forms of MOGA have been implemented. These include sub-population methods (Schaffer, 1985; Kursawe, 1991) and a selection of Pareto-based approaches (Goldberg, 1989, Fonseca and Fleming, 1993, Horn, Nafpliotis, and Goldberg, 1994, Srivinas and Deb, 1995).

Hybrids of these techniques have also been explored (e.g., Tamaki et al., 1996; Osyczka and Tamura, 1996) In this work, the Pareto-ranking MOGA technique of Fonseca and Fleming (1993; 1998) is used. This method ranks the solutions according to their Pareto-dominance of other solutions, thus providing a whole set of Pareto-optimal solutions at the end of the GA run, from which the user can choose the final schedule to suit the immediate requirements of the plant. This flexibility has, in initial discussions with manufacturers, provided an attractive extension to the more common provision of one single optimal solution based on a pre-defined priority placed upon the objectives.

3.2 Genetic Algorithm Implementation

For an initial problem based on the plant illustrated in Figure 2, the optimization problems focuses on two main decision variables, batch-sizing decision and batch-reactor assignment. Additional details of this problem may be found in Shaw et al, 1999, but briefly the relevant points are as follows.

The optimal schedule of the required *campaigns* (set amounts required of a particular product) must be found, by dividing the amounts into suitable batches as necessary, and assigning the batches to suitable reactors. Batches must be of a size to meet the plant constraints, and must be made by a set time deadline, whilst minimising the following objectives;

Objective 1. Minimise cleaning time between batches

Objective 2. Minimise storage cost of batches completed which are completed early

Objective 3. Maximise percentage wasted plant capacity

Objective 4. Minimise late orders, i.e., failure to meet customer deadlines

Objective 5. Minimise variation in binary sections of schedule from exact amounts required ("batch variation" / "infeasibility")

The three batch sizes are 0.5, 1 or 2 tonnes, defined by the maximum holding capacity and minimum running capacity of the three reactors. All other processes are assumed to meet the scheduling requirements; for example, raw materials are available and ready when required. For each campaign, the required tonnage, speed of each reaction, cost of storage for the completed product, and the due date are supplied. This data is sufficient for the initial scheduling problem.

The GA aims to optimize the five objectives by manipulating two major decision variables of the batch scheduling plant. A concise representation of these two variables within the GA uses two parts of the population to reflect this. The individuals are formed in two parts. The first part, a binary string, represents the batch sizing decision, in a similar GA-based technique to that used for solving 0/1 knapsack problems (e.g., Michalewicz and Arabas, 1994; Zitzler and Thiele, 1998). The second, a permutation part, represents the sequencing. The two parts are concatenated to form one individual.

Each bit in the binary part of the individual indicates whether a batch of a particular size and product should be made to complete the overall order, according to a pre-defined set of possible batches which represent the available range within the constraints of the problem. The permutation part of the individual represents the sequence in which the batches should be made. Standard binary and permutation crossover and mutation operators can be used on relevant parts of the population.

Translating this basic information about batching and sequencing into a working schedule requires an encoding, or schedule builder stage. From the list of sequenced batches, an algorithm is used to assign each batch to the appropriate unit for the relevant procedure for that recipe to be performed. Within this assignment process, there is often more than one suitable unit is available for allocation to a procedure, even after several constraints upon such

assignments have been satisfied. Thus the schedule builder includes a set of possible rules from which the assignment decision must be chosen. For this work, a choice of two schedule builders, and two objective functions are implemented. SB1 uses a rule of choosing the fastest unit to complete the task, even if this means waiting for the unit to become available and SB2 chooses the first suitable unit to become free for the job, even if it is the slowest once the job is underway. The completed allocation allows a final schedule to be created, to include start, finish, and cleaning times for each product, from which the plant objectives can be evaluated, as listed above and returned to the GA optimization process.

The MOGA generally includes some form of separate treatment of each objective at the selection stage. In this example, given the four plant objectives and feasibility objective, it is necessary to decide whether the optimization process benefits from a separation of the four plant objectives for individual optimization, in the form of a five-objective MOGA, or whether it is sufficient to optimize two objectives, one based on an aggregation of the plant-based objectives and the other being the feasibility cost.

This provides four MOGAs for comparison; the two-objective (weighted sum of all plant costs versus feasibility cost) and five-objective (all costs separated) MOGAs, each using the two schedule builders. The GA code is implemented in GA MATLAB Toolbox (Chipperfield et. al., 1994); running on a SPARC station (Sun Ultra 5, 333 MHz, 128 Mb RAM). The average run times were: approximately one hour for the two-objectives, and approximately ninety minutes for the five-objectives.

4. Experimental results of MOGA comparisons

The MOGAs were run eleven times each, with a population of 100 individuals, over 100 generations. Standard binary operators were used, together with the Order Crossover (Davis, 1985; Oliver, Smith and Holland, 1987), and Swap Mutation for the binary and permutation parts of the population respectively, using following operator rates. Various sets of test problem orders were generated and scheduled.

4.1 MOGA Comparison Technique - AFCM

Several recent works (Van Veldhuizen and Lamont, 1998; Zitzler and Thiele, 1998; Coello, 1999; Shaw, Fonseca and Fleming, 1999) begin to explore the field of MOGA comparison. The various methods of MOGA comparison generally aim to assess performance in two ways. The first examines performance according to their spread across the available trade-off surface. The second measures their ability in terms of attainment of the optimum goal in some multiobjective sense. In some cases, a combination of both these measures is used. In this work, to suggest the best approach for implementing MOGA optimization upon the five-objective batch scheduling problem, results such as the absolute minimum values achieved are explored together with an implementation of the attainment function comparison method (AFCM) (Shaw, Fonseca and Fleming, 1999).

The AFCM method is developed as a statistical, quantitative method for evaluating the performance and comparison of MOGAs, using the notion of attainment surfaces. Given a problem, a point in objective space will lie on the 50%-attainment surface of a given optimizer if, and only if, a single run of the optimizer on that problem will find a better solution (in the Pareto sense) with probability 0.5. Thus, attainment surfaces provide a quantitative method of characterizing the performance of a genetic algorithm, both in terms of absolute values achieved and spread of solutions.

In Fonseca and Fleming, (1996), a statistical interpretation of performance of a multiobjective optimizer was given, based on the probability that the optimization technique would attain an arbitrary goal, (i.e., produce at least one solution better than or equal to a given point in objective space) in a single run. Thus, the probability of attaining a given goal can be seen as a function of the goal, and is called an *attainment function*. Such probabilities can be estimated directly from data, thus, attainment functions can be used as measures of multiobjective optimizer performance. The empirical estimates of attainment functions will be called empirical attainment functions (EAF), by analogy with the empirical cumulative distribution function. Computing the EAFs for each algorithm considered, and plotting their contours, particularly at 0, 0.5, and 1, gives a clearer idea of the variation observed across runs of each of the algorithms, in combination with the spread of solutions observed in each run.

In addition, this method introduces performance comparisons in terms of statistical hypothesis testing. For the comparison of the performance of the two methods, MOGA1 and MOGA2, the following null and alternative hypotheses are formulated:

> ***H_0: There is no performance difference between MOGA1 and MOGA2***
>
> ***H_1: There is a performance difference between the methods***

Under H_0, MOGA1 and MOGA2 should exhibit equal attainment functions. By implementing a form of the permutation test (Good, 1994), a test-statistic may be estimated. The null hypothesis may be rejected if the estimated probability p of the test statistic being greater than, or equal to, the value observed is not higher than a given (small) significance level, usually from 0.05 to 0.01. This method is discussed in more detail in Shaw, Fonseca and Fleming, 1999, but the results of applying such a test to the MOGA results in this work are demonstrated below in XXX.

4.2 First comparison - Comparison of the schedule builders for two objective MOGAs

The first test compared the performances of two-objective MOGAs, incorporating schedule builders SB1 and SB2 respectively. Initial comparisons were made as to which schedule builder might allow the algorithm to attain optimum values of both costs. Summary statistics of the costs found by all non-dominated solutions from each MOGA are given in Table 1. This indicates the comparative sets of values found in the entire search, but little additional information or insight into the behaviour of the MOGA itself.

		Cost 1	Cost 2
SB1	Min	0.0003	0.0000
	Mean	0.0085	0.4450
	Max	0.0404	0.8390
SB2	Min	0.0003	0.0000
	Mean	0.0075	0.4530
	Max	0.0385	0.9330

Table 1 - Comparison of summary statistics for all solutions found in experimental runs of two-objective MOGA with both schedule builders

Both methods are able to achieve the minima for both costs, in particular, finding completely feasible solutions, as shown by the zero values for Cost 2. SB2 appears to show better performance regarding the mean cost for Cost 1 (aggregate plant costs) and SB1 for Cost 2 (feasibility). However, applying the AFCM hypothesis test, as outlined in section 4.1, with null hypothesis that there is no difference between the MOGAs' performances, gave a p-value, p=0.5240. This allows a conclusion that there is no significant difference in the performance of these two MOGAs for this problem, and therefore, the two different schedule builder rules have no particular result on the overall set of solutions found. The two schedule builders are now implemented in five-objective MOGAs, in which each cost is treated as a separate objective.

4.3 Second comparison - comparison of the schedule builders for five-objective MOGAs

The same tests were performed to examine the effects of the two scheduling rules within the schedule builders given a five objective problem. The aims of the comparison were not only to explore the consequences of the rules, if any, upon the optimization behaviour, but also to attempt to gain insight into their effects. Table 2 shows a summary of the costs found by all non-dominated solutions over all runs.

		Cost 1	Cost 2	Cost 3	Cost 4	Cost 5
SB1	Min	0.0016	0.0000	0.0000	0.0000	0.0000
	Mean	0.0869	0.0847	0.0507	0.0635	0.2720
	Max	0.2594	0.6941	0.2787	0.4830	0.8060
SB2	Min	0.0016	0.0000	0.0000	0.0000	0.0000
	Mean	0.0907	0.0697	0.0552	0.0715	0.2660
	Max	0.2910	0.5804	0.3320	0.5060	0.8060

Table 2 - Comparison of summary statistics for all solutions found in experimental runs of five-objective MOGAs with SB1 and SB2

In this instance, a clear pattern can be seen that SB1 seems to find lower costs 1, 3 and 4, and SB2 appears better for Cost 2 and Cost 5. Both are capable of finding feasible solutions, shown by values of zero for Cost 5. The effects of the two schedule builders on the plant costs can be more clearly seen as by applying the AFCM.

Figure 3 - Figure 6 show the plots of attainment surfaces 0.5 ("median surface") and 1 ("worst case surface") at this stage, as parallel co-ordinate plots. It is recommended practice to examine the plot of attainment surface 0 ("best case surface"), but in this instance, there was little of significant interest in these plots and they are omitted for reasons of space. They are also less relevant given that the application of this work is intended to be a real-life process plant. The results of surface 1 indicate the typical performance from a 'one-off' single run, which would be more commonly found in a practical situation from one run of the scheduling program. Conversely, the attainment surface 0 plot is representative of performance after multiple experimental runs.

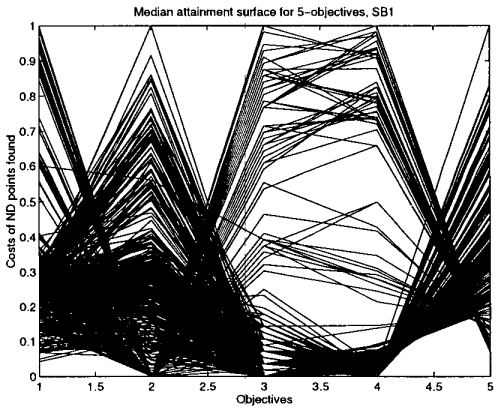

Figure 3: Attainment surface at 0.5 (median), SB1

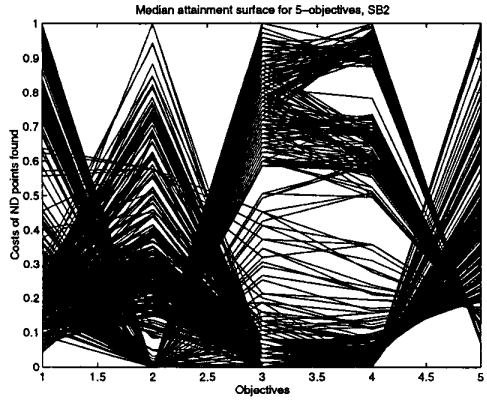

Figure 4: Attainment surface at 0.5 (median), SB2

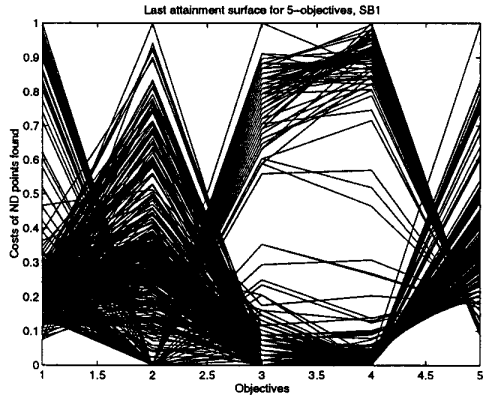

Figure 5: Attainment surface at 1 (worst), SB1

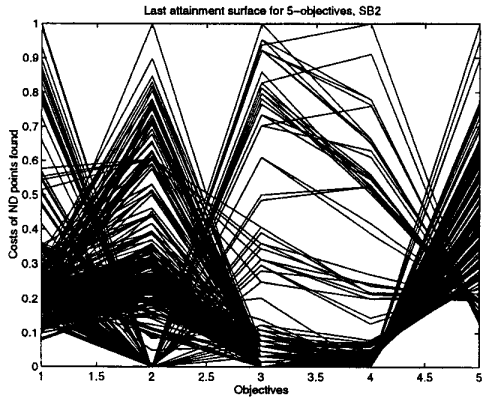

Figure 6: Attainment surface at 1 (worst), SB2

Regarding the comparison of the 0.5 (median) plots, the MOGA using SB2 provides a denser set of solutions throughout, particularly around the cost3-cost4 area and in its coverage of the range of values for cost 1. However, this advantage is lost in the attainment surface 1 plot. In these, the MOGA using SB1 seems to show a slightly wider range of possible solutions than that using SB2, particularly in the cost 1 and cost 3-cost 4 range. The use of SB2 loses its advantage from the median plot, which represents "typical" performance from half the full number of repeated runs, compared to the 'worst' plot, which is more indicative of the results that might be expected in practice. Whilst these plots are useful to a limited extent, it can hard to extract additional information visually, given the high number of objectives and solutions shown. Applying the confidence test to the results allows additional identification of the areas in which the MOGAs differ.

The p-value for the test outlined in 4.1 - that there was no performance difference between the two MOGAs, was p=0.0000, allowing the conclusion that there are significant differences in the methods. By identifying the points at which this difference was significant, discussed in more detail in Shaw, Fonseca and Fleming, (1999), various points were found at which one method or another was significantly better. These are plotted in Figure 7.

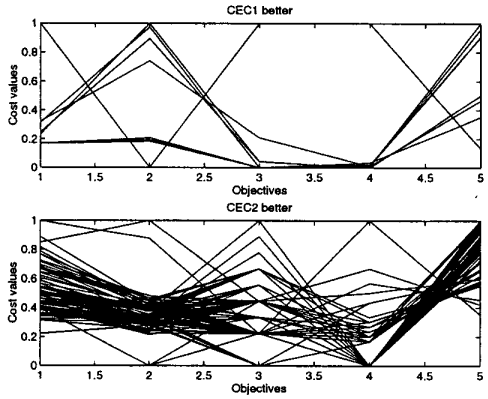

Figure 7 - Points at which the methods appeared to show significant differences in performance

As indicated in Table 2, these plots show that there seem to be two clear types of solution regarding the set of trade-offs occurring. These are subsequently referred to as solution type A - low cleaning (1), high storage (2), low % wastage (3), low lateness (4), medium-high feasibility (5)- , and solution type B - high cleaning (1) - low storage (2) - mid-high % wastage (3) - high lateness (4) - low feasibility (5). The combination of costs within these solutions suggests that type B solutions may be focusing on the smaller batches, which would require more frequent cleaning and less storage, and examination of the batches found in 100 randomly sampled 'type B' solutions gave a ratio of batch sizes (0.5, 1, 2) as (0.53, 0.3, 0.17). Similarly, type A solutions seem to focus on the selection of larger batch sizes, (the equivalent ratio being (0.04, 0.16, 0.8)). The larger batch sizes will allow more frequent satisfaction of the capacity constraint; 80% of the batches are a maximum sized batch and therefore can run in two out of three reactors with no wastage. Generally, SB1 seems slightly better for finding A-type solutions, and SB2 for B-type. Neither scheduler builder seems better on the attainment of feasibility (that is, reaching zero for cost 5), but SB1 offers a wider range of values for this cost, whilst SB2 offers a greater number of solutions within a limited range.

It is interesting to compare this result with that in 4.2, which concludes that the schedule builders make no significant difference to performance in the two-objective

MOGAs. Therefore the final comparison test compares the two and five objective MOGAs directly. Finally, factory practices commonly focus on 'type A' solutions at present, and therefore, as SB1 seems slightly better for finding these types of solution, as well as having a more robust performance, SB1 is now applied in a comparison of the two-objective MOGA against the five-objective.

4.4 Third comparison - Two objective versus five objective, using SB1

In the third comparison, the two-objective MOGA including SB1 was compared to the five-objective MOGA including SB1. To allow the costs to be compared effectively, all non-dominated costs in five objective space were stored during the run of the two-objective MOGA. This comparison was performed mainly by a AFCM confidence test, which provided a p-value of zero, p=0.0000, and allowed the following points to be identified at which the methods differed significantly (Figure 8).

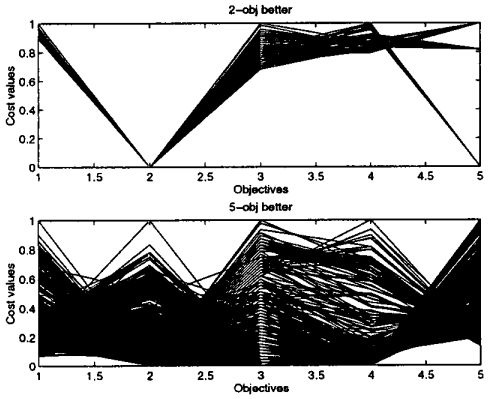

Figure 8 - Points at which the methods appeared to show significant differences in performance

There are immediate clear differences. The two-objective MOGA does have its advantages: it has some significantly better solutions that achieve zero infeasibility (cost5=0); but these are all very limited within the range of "type B" solutions. This may not be surprising given that the aggregation of the plant objectives allows their individual influences to contribute less significantly to the overall optimization process, compared to the influence exerted by cost 2 (feasibility). It is clear that the five-objective MOGA can search the whole space with more much effectively, and finds many solutions that are significantly different, allowing a whole range of possible trade-offs between the various costs - and solutions of type A or B - to be offered to the user for the final decision as to which schedule to use.

It is therefore suggested that the implementation of the five-objective MOGA is the recommended technique with which to continue with the work on this problem. Section 5 provides the conclusions to this work.

5. Conclusions and Suggestions for Further Work

This paper demonstrates the application of two-objective and five-objective MOGAs to a simplified batch process scheduling problem. Both MOGAs attempt to optimize the sequencing sizes of batches used to complete the overall orders. The MOGA is used as a method of handling the infeasibility constraint, and allows some flexibility in this; by allowing some infeasibility, better results may be achieved for the other objectives. It may be possible to relax the feasibility requirement considering these visible trade-offs. With more detailed modelling, this particular infeasibility may be incorporated into existing or additional 'non-feasible' related objectives (e.g., a financial penalty for not making the correct amounts, or increased storage costs for handling surplus batches.) This would, however, increase the number of objectives further. It remains to be seen whether this is an effective way of avoiding this particular infeasibility.

The MOGA is also effective in handling the variety of plant objectives available, and in addition, allowing insight into the actual problem. Figure 8 suggests that the two-objective MOGA may be more successful with an alteration in the weighting used to explore these other costs more fully, if this method is implemented further. However, the five objective implementation seems to offer many benefits in this work, given continuing attention to the effects of the schedule builders upon the results as the model becomes more complicated. Results may be more pronounced with a more detailed model, in which the effects of both the batch sizing and routing decisions may have more effect.

The work has provided insight into the trade-offs between the objectives that are influential in the optimization process. Manipulation of these trade-offs also provide solutions that are recognisable as current plant practice (e.g., 'type A') solutions, but also many other solutions have been suggested which are optimal in the multiobjective sense and could provide potentially useful scheduling solutions to the problem. It is also interesting that the effects of the batch sizes on the objectives, and vice versa, perhaps indicating some further guidelines for batch sizing decision-making within the plant.

The AFCM method has been implemented to demonstrate the benefits of the various MOGA implementations, and the confidence test implemented allows identification of the areas in which the MOGAs differ. This is, to the authors' best knowledge, the first example of this MOGA comparison method being applied to a five-objective problem. The area of the performance comparison of MOGAs with a high number - in this sense, greater than six - of objectives is also a novel area for exploration. Van Veldhuizen and Lamont, 1998, comment that many MOGA comparisons focus on two or three objective problems. However, the comparisons performed are extremely computationally intensive. There is also scope for additional work in this area.

Acknowledgements

The authors gratefully acknowledge the support of EPSRC grants GR/K31343, GR/J70857 and GR/L7351 in this work.

References

Bäck, T., Fogel, D.B, and Michalewicz, Z., 1997 (eds.). *Handbook of Evolutionary Computation*, IPP / OUP.

Bagchi, S., Uckan, S., Miyabe, Y., Kawamura, K., 1991. Exploring Problem-Specific Recombination Operators for Job-Shop Scheduling, *in Belew and Booker, 1991.*

Chipperfield, A. J. Fleming, P. J., and Pohlheim, H., 1994. A genetic algorithm toolbox for MATLAB, *Proc. Int. Conf. on Systems Engineering*, Coventry, UK, 200 - 207.

Coello, C. A., 1999. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, *Knowledge and Information Systems. An International Journal.* (Accepted for publication).

Chu, P. & Beasley , J. E., 1992. Constraint Handling in Genetic Algorithms in the set partitioning problem, *Working Paper*, available from http://graph.ms.ic.ac.uk/jeb/jeb.html

Cleveland, G. A., and Smith, S. F., 1989. Using genetic algorithms to schedule flow-shop releases, *Proc. Third Int. Conference on Genetic Algorithms.*

Davis, L., 1985. Job Shop Scheduling with Genetic Algorithms, in *Grefenstette, 1985.*

Fonseca, C. M., Fleming, P. J., 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, in *Forrest, 1993.*

Fonseca, C. M., and Fleming, P. J., 1998. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms - Part II: *Application Example, IEEE Trans. Systems, Man and Cybernetics - Part A: Systems and Humans, 28, 1, January 1998.*

Fonseca, C.M. and Fleming, P. J., 1996. *On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers*, Parallel Problem Solving From Nature IV, pp. 584 - 594, Berlin.

Forrest, S., 1993, (Ed). *Proc. Fifth Int. Conf. on Genetic Algorithms.* Morgan Kaufmann Publishers.

Goldberg, D. A., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley

Good, P., 1994. *Permutation Tests, A Practical Guide to Resampling Methods for Testing Hypotheses*, Springer Series in Statistics, Springer-Verlag.

Grau, R., Espuna, A., and Puigjaner, L., Completion times in multi-purpose batch plants with set-up , transfer and clean-up times, *Comp. Chem. Eng.*, 20, S, ppS1143-S1148, 1996

Grefenstette, J. J., 1985. *Proc. First Int. Conf. on Genetic Algorithms and their Applications*, Lawrence Erblaum Publishers

Horn, J., Nafpliotis, N, and Goldberg, D. E., 1994. A Niched Pareto GA for Multi-Objective Optimization. *Proc. First IEEE Conf. on Evolutionary Computation*, 1994.

IEC, 1997. Batch Control Part 1- Models and Terminology, IEC 61512-1, IEC.

ISA, 1995. ANSI / ISA-S88.01.1995 *Standard Batch Control; Part 1: Models and Terminology.* ISBN 1-55617-562-0, Instrument Society of America, 1995.

Ishibuchi, H, and Murata, T, 1998. A Multiobjective Genetic Local Search Algorithm And Its Application To Flowshop Scheduling, *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 28,3.Kim, M., Jung, J.H., and Lee, I., 1996. Intelligent scheduling and monitoring for multi-product networked batch processes, *Comp. Chem. Eng.*, 20, S, pp S1149-S1154.

Ku, H. M. and Karimi, I. A., 1990. Scheduling in serial multi-product batch processes with due dates penalties, *Ind.Eng.Chem.Res.*29, pp580-586.

Ku, H. M. and Karimi, I. A., 1991. Scheduling algorithms for serial multi-product batch processes with tardiness penalties, *Comp.Chem. Eng.*, 15, 5, pp283-286.

Kursawe, F., 1991. A Variant of Evolution Strategy for Vector Optimization, *Parallel Problem Solving from Nature 1*, pp. 193 - 196.

Lee, I., Sikora, R., Shaw, M. J., 1993. Joint Lot Sizing and Sequencing with Genetic Algorithms for Scheduling - Evolving the Chromosome Structure, in Forrest, 1993.

Löhl, T., Schulz, C. and Engel, S., 1998. Sequencing of Batch Operations for a Highly Coupled Production Process: Genetic Algorithms versus Mathematical Programming, Comp. Chem. Eng., 22, S.pp. 579 - 585, 1998.

Love, J and Bunch, M., 1998. Decomposition of Requirement Specifications for Batch Process Control, *Trans. IChemE*, 76, 8, pp. 973 - 979.

Michalewicz and Arabas, 1994. GAs for the 0/1 knapsack problem, Methodologies for Intelligent Systems, *ISMIS '94, . LCNS 869*, pp 134 - 143, Berlin, Springer.

Michalewicz, Z., 1997. C5.1, Introduction, Constraint-Handling Techniques, in Bäck et al, 1997.

Niemeyer, G. and Shiroma, P., 1996. Production Scheduling with GA and Simulation, *Parallel Problem Solving from Nature 4*, Berlin, 930 - 936.

Oliver, I. M., Smith, D. J., and Holland, J. R. C., 1987. A Study of Permutation Crossover Operators on the Travelling Salesman Problem, in *Grefenstette, 1987.*

Osyczka, A. and Tamura, H., 1996. Pareto Set Distribution Method for Multi-Criteria Optimization Using GA. *Proc. of Mendel 96, the 2nd Int. Mendel Conference on Genetic Algorithms*, TU Brno, Czech Republic, June 1996.

Schaffer, J. D., 1985. Multiple Objective Optimization with Vector Evaluated Genetic Algorithm, in *Grefenstette, 1985.* pp. 93 - 100.

Shaw, K. J, Fleming, P. J., (1996). An Initial Study of Practical Multiobjective Production Scheduling Using Genetic algorithms, Proc. UKACC Int. Conf. On Control '96, University of Exeter, IEE.

Shaw, K. J. , A. L. Nortcliffe, M. Thompson, J. Love, P. J. Fleming, (1999). Exploring Use of Multiobjective Genetic Algorithms for Constrained Process Scheduling, to appear at IFAC '99, Beijing, July 5th - 9th, 1999.

Shaw, K. J. , Nott, H. P., Lee, P. L., 1998 A Study of the Development of a Genetic Algorithm for Scheduling Combined Batch / Continuous Process Plants, submitted to *Euro. Control Conf. 99.*

Shaw, K. J., and Fleming, P. J., 1997. Including Real-Life Problem Preferences in Genetic Algorithms to Improve Optimization of Production Schedules, *Proc. GALESIA '97*, Glasgow, 2nd - 4th September, 1997.

Srinivas, N. and Deb, K., 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, Vol. 2, No. 3, pp. 221 - 249.

Stefanis, S.K., Livingston, A.G., Pistikopoulos, E.N., 1997. Environmental impact considerations in the optimal design and scheduling of batch processes, *Comp. Chem. Eng.*, 21, 10, pp 1073-1094.

Surry, N. J., Radcliffe P. D., and Boyd, 1995. A Multi-Objective Approach to Constrained Optimization of Gas Supply Networks: the COMOGA Method , *Evolutionary Computation: AISB95*, 1995.

Tamaki, H., Nishikawa, Y., 1992. A paralleled GA based on a neighbourhood model and its application to Job-shop scheduling, *Parallel Problem Solving from Nature*, 2 1992, Ch. 60, pp. 573-582.

Tamaki, H., Kita, H., and Kobayashi, S., 1996. Multi-Objective Optimization by Genetic Algorithms; A Review. *Int. Conf. on Evolutionary Computation '96.* pp. 517 - 522.

Van Veldhuizen, D. A., and Lamont, G. B., 1998. *Multi-objective evolutionary algorithm research: a history and analysis*, (final draft) TR-98-03, research report, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. 45433-7765.

Viennet, R., Fonteix, C, and Marc, I, 1995. New Multicriteria Optimization Method Based On The Use Of A Diploid Genetic Algorithm: Example Of An Industrial Problem, *Proc. Artificial Evolution*, Brest, France, 120 - 127.

Zitzler, E. and Thiele L., 1998. *Multi-Objective Optimization Using Evolutionary Algorithms-a Comparison Case Study*, Parallel Problem Solving from Nature-PPSN5, Lecture Notes in Computer Science 1498, Amsterdam, September 1998, Springer Verlag.