Interactive Batch Process Schedule Optimization and Decision-Making using Multiobjective Genetic Algorithms

K. J. Shaw, A. L. Nortcliffe, M. Thompson, J. Love, and P. J. Fleming
Department of Automatic Control and Systems Engineering,
University of Sheffield, Mappin Street, S1 3JD, UK.
Emails: {jane.shaw | a.l.nortcliffe | m.thompson | j.love | p.fleming} @shef.ac.uk

ABSTRACT

A multiobjective genetic algorithm (MOGA) is applied to a test batch scheduling problem to attempt to optimize five objectives simultaneously. The design of the MOGA allows an emphasis on human interaction with the optimization process, including the ability to change priorities of preferences and plant data interactively, and to allow the MOGA to make decisions regarding batch size and the rules task allocation. Experimental results demonstrate the development of this technique, allowing the combination of human expertise and MOGA optimization power to provide scheduling solutions to a highly complex problem.

1. INTRODUCTION

Chemical batch processing scheduling problems commonly require the optimization of several conflicting objectives. The demanding nature and complexity of these problems means that a balance must be found in the optimization process, between the interaction of expert human knowledge and decision making, and the automated schedule optimization power and computing ability, within an accurate scheduling system.

Genetic algorithms (GAs) are evolutionary-based optimization tools [1], suited to scheduling, [2]. Multiobjective GAs (MOGAs), [3], [4], [5], [6], [7] are becoming increasingly popular for demanding applications including real-life scheduling systems. MOGAs also offer much potential for providing a general interactive decision-making tool throughout the optimization process. These decision-making interactions can be implemented by several techniques. Manufacturing rules may be generated or chosen and executed during the evolutionary scheduling process itself, to benefit the overall optimization goals, [8], [9], [10]. Additionally, the provision of a set of several equally optimal scheduling solutions to a problem, as an outcome of the multiobjective optimization process, allows user interaction with the choice of the final solution. Human expertise can be combined with computational intensive search techniques to direct the search and focus on solutions that can meet the users' immediate requirements, interests and particular needs. The evolutionary process also offers dynamic adaptation to changes in the plant variables during the optimization process itself.

A batch process problem is explored, and the need for the inclusion of a series of decision-making opportunities for the combined evolutionary computing / human-interactive system is discussed. A MOGA is implemented in a flexible, object oriented environment that facilitates user interaction, whilst adhering to the exact structures defined by a framework of the ISA standard for batch scheduling, S88 [11], [12].

The work is arranged as follows. Section 2 introduces the batch process plant problem being used for the testing of the optimization system. Section 3 provides details of the optimization system implementation itself, including the GUI, and MOGA, and discusses the interaction between the parts of the system, including the human user. Section 4 illustrates experimental development of the MOGA's decision-making capabilities, in the form of an evolving rule generation system. Section 5 discusses additional measures taken within the MOGA and by the human operator to allow reaction, interaction and flexible treatment of the batch process schedule optimization procedure. Finally, section 6 provides discussions, conclusions and suggestions for further work.

2. PROBLEM DESCRIPTION

In recent years, the chemical process industry has shifted its production methods from high volume continuous production of low cost products to producing low volume, specialist chemicals of high value, produced by batch operations. The latter involves complex chemistry, process operations and results in small yields. To ensure that such a manufacturing process is more cost effective and efficient, multi-purpose batch plants are used. The generic design of these plants allows the processing a broad range of products and variety of operations. Many process and simultaneous operations may be accommodated, but the additional complexity of the plant creates a need for careful scheduling to ensure that all the resources are used effectively and efficiently. The standards S88.01 [11] and IEC 61512-1 [13] provide guidelines which cover the design, control strategies and schedules for such plants. A standardised framework of terminology, models and structures is used to describe and articulate the scheduling requirements of a multi-purpose batch plant [12]. The definitions prevent ambiguity within the scheduling process and allow portability of the completed method.

In this work, a MOGA-based scheduling system is developed to be framed by the S88 standard, whilst attempting to expand and explore the degrees to which flexibility and interaction can be included within a batch process scheduling system. The system should allow both improved performance of schedule optimization, and also offer insights and feedback for the manufacturer into the plant behaviour and scheduling process itself. A generic multi-purpose batch plant and set of recipes for possible products have been expressed in S88 constructs to provide a test scheduling environment. The overall plant is a process cell consisting of units and equipment modules, shown in Figure 1.

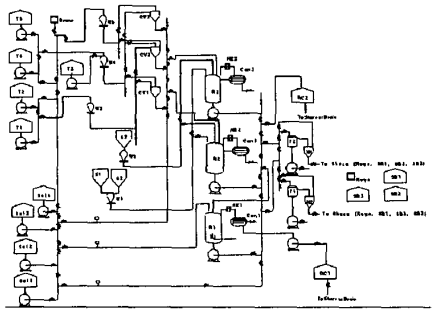

Figure 1- Process cell of the generic multi-purpose batch plant

Five products are made within the plant. The total amount of each product ordered must be divided into batches if the total is greater than the maximum reactor size; the maximum capacity of the reactors defines the maximum batch size. The batches are made according to recipes, which are broken down into S88 standardised *phases*. The set of phases required to complete a product must be scheduled to proceed through the various units of the above plant, in order to meet a set of four production objectives. The development of the S88 standardised recipe into a set of tasks suitable for scheduling by the MOGA is discussed in section 2.1, and plant objectives are provided in section 3.2.

0-7803-5731-0/99/$10.00 ©1999 IEEE

2.1 The implementation of S88-based scheduling system

Love and Bunch, [12], outline the batch process scheduling problem as follows: "whatever the structure of a batch plant... the objective is to make batches of product. There are many variations on this theme. [...] Whatever the scenario, consideration of any single batch reveals that it is always processed by a series of (one or more) operations in a (dedicated vessel or) train of vessels according to some recipe. [...] For each product, the route that the batch follows through the plant has to be established, and the operations carried out on that batch considered, i.e., which operations, where and in what order."

Under the definition of S88, phases are individually and separately defined tasks. A phase is defined to be the "smallest practicable group of control functions that can be sufficiently decoupled from the rest of the operation to exist in isolation." To make a product, various phases grouped under the larger procedures of 'weighing', 'charging', 'transferring', 'reacting', and 'filtering' are required in the predefined order. The generalised 'weighing' procedure consists of several more specific phases according to the recipe of each particular product, such as 'weigh solution 1, solvent 2, and solid 3.'

Phases may have idle time scheduled before or after them; they are defined to allow their production to be paused safely at the start or finish. They may run in series or in parallel. Normal practice is to schedule the phases themselves by assigning each to a separate unit (that is, piece of process equipment, such as weighing vessel, charge vessel, reactor or filter) for a particular time. However, in many cases, successive phases take place in the same unit by practical necessity. In the case of the reactors, several phases make up a reaction task (e.g., 'initialise, heat, hold, sample, cool'). It would be meaningless to assign the first step of this reaction to one reactor whilst the second is assigned to another. Thus in this example, the allocation decision need only be made for the first task, allowing some sets of phases may be effectively grouped into an equivalent single tasks.

The definition of such tasks – some consisting of single phases and some of compatible groups of phases - lends itself well to the scheduling process. To complete a product, the schedule must assign each task to a suitable unit whilst meeting the sequencing constraints, in order to optimize the required costs.

The complete batch recipes for the given problem consist of 80-100 phases in total, each of which must be scheduled for every batch to make up the complete orders. Even with as few as five products, this creates a large number of tasks to be scheduled. As an initial implementation, the work focused on smaller task of scheduling only of the reactor operations, each consisting of a number of reaction phases.

2.2 Motivation for development of a MOGA-based, interactive scheduling system

Multiobjective GAs (MOGAs) are becoming increasingly popular for demanding applications, including scheduling systems (e.g., [14], [15], [16], [17], [18], optimizing multiple conflicting and complex objectives. MOGAs also offer much potential for providing a general interactive decision-making tool throughout the optimization process.

There is additional motivation for providing an extensive batch process scheduling optimization system powered by a MOGA. The MOGA has demonstrated promising abilities on real-life scheduling problems related to the food industry [10], an application which holds many similarities to the problems also faced in the chemical industry. The MOGA's key advantage is its ability to treat the multiple, frequently conflicting objectives found within a real-life optimization problem separately. Coello [19], lists the rapidly increasingly range of problems for which such techniques can be highly beneficial for solving optimization problems. The search nature of the MOGA system allows vague, ill-defined or uncertain data to be included within the optimization system, which is advantageous for any problem in which the related human decision-making or reaction process cannot be modelled exactly. Indeed, doing so may even benefit the overall optimization performance [10]. Additional development work [20] has suggested a method to measure confidence in the results provided by MOGA scheduling systems, allowing the practical user to ascertain the expected optimization performance from a particular system. This also allows the developer to choose the best-performing MOGA for the problem, out of the many possible implementations. As is the case with any evolutionary computing optimization system, a MOGA may be specifically designed to suit the particular problem. Thus the MOGA in this work has been developed to include the ability to determine batch-sizing decisions [21], as well as batch sequencing and scheduling. In addition, section 4.1 develops the MOGA further to evolve batch allocation decisions within the optimization process. The particular details of the method and the inclusion of these properties are discussed further in 4.1.

3. SYSTEM IMPLEMENTATION

3.1 GUI implementation

The first part of the schedule system visible to the user is in the form of a Java-based GUI. As the system aims to provide a high degree of user interaction with the scheduling optimization process, the interface is an important feature. Through the interface, the MOGA takes the data required to define the plant, recipe and tasks, and acts as intermediary between the plant model, the MOGA and the human. Data can flow in either direction; inputs from the human to the GUI can alter the plant specification and MOGA implementation, whereas output from the MOGA scheduling system may advise the human to adjust their scheduling decisions as the optimization process continues; either as a reaction to the direction that the search is taking, or more generally, in response to a general trend suggested by results that may prove influential in human scheduling decisions.

The GUI relating to the plant specification is shown in Figure 2, containing information including: characteristics, functions, capacity, charge and discharge times of the units, their materials of construction, the type of resource, output connections, and additional equipment features, such as heating modules.

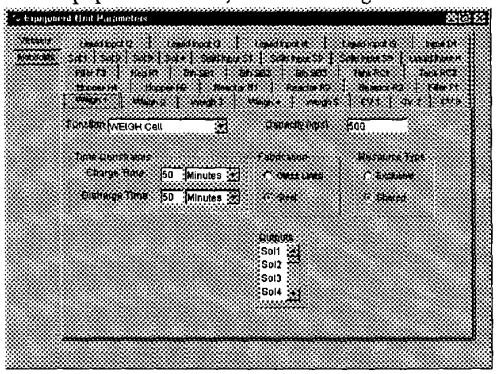

Figure 2 - Java GUI for plant specification

The user may change any data relating to these aspects via the GUI. This may be *a priori*, before the MOGA begins its optimization run, or interactively during the optimization. Further interaction may take place after the optimization is completed. The effects of this later interaction are demonstrated in section 5.

Information describing the recipes' breakdown into individual scheduling tasks is entered into the GUI shown in Figure 3. This includes such diverse elements as the tasks that need to be completed in order to manufacture the product. Each task window displays the type and duration of the task, additional ingredients added (if any) to the product at that stage, specific units required, and the cleaning requirements after completion.

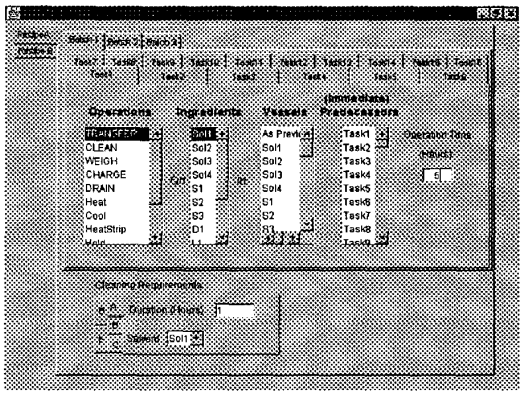

Figure 3 – GUI for recipe data

Again, the ability to alter any of this data either before or during a run is available to the user, providing extensive interaction with the data being relayed to the MOGA scheduling process at several levels, regarding all aspects of the plant model and recipe requirements.

3.2 MOGA implementation

The genetic algorithm (GA) is an optimization technique simulating various processes of natural evolution [1], namely repeated selection, recombination and mutation. Its search points are *individuals* within a randomly generated population, each representing a potential solution to a problem. The objective values of the individual's solution to the problem are evaluated, reflecting its *fitness* within the population. Fit solutions are proportionally selected to recombine creating a new generation of "offspring" solutions. A mutation process adds an element of novel material, randomly at a low probability. This repeated process creates a simulated 'survival of the fittest' effect, as good characteristics in solutions become predominant in succeeding generations, until some criteria for convergence are reached. Many variations of the basic implementation are possible, and it is common practice for a GA to be designed to meet specific requirements of a problem.

For an initial problem based on the plant illustrated in Figure 1, the optimization problem focuses on three main decision variables:

1. batch-sizing decision
2. the sequencing of the tasks relating to the batches required
3. the rule used to assign the batch to a unit, given a choice of more than one possibility for such an assignment

The optimal schedule of the required tasks must be found. This is performed by the division of the total orders into suitable batches, sized to contribute towards the optimum schedule, then assigning the batches to suitable reactors, according to the set of rules evolved for the assignment. 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;

1. Minimize cleaning time between batches
2. Minimize storage cost of batches which are completed early
3. Minimize percentage wasted reactor capacity
4. Minimize late orders, i.e., failure to meet customer deadlines
5. Minimize variation of the total amount made by the batches suggested from those required ("batch variation").

The fifth objective defines the notion of schedule feasibility. Whilst the MOGA may generate schedules which may under- or over-produce the amounts required by the customers, these are termed 'infeasible' solutions. Whilst not encouraged, they are still permitted within the population, to preserve variation within the individuals. The three batch sizes possible are 0.5, 1 or 2 tonnes; these values being defined by the maximum holding and minimum running capacities of the three reactors. All other processes are assumed to meet the scheduling requirements; for example, raw materials being available and ready when required. For each set of orders ('campaign'), the required tonnage, speed of each reaction, cost of storage for the completed product, and the due date are supplied. Plant data provides a list of possible reactors for each task, depending on the reactor material and other constraints. In addition, the possibility of plant data changing is simulated by the triggering of three separate events at randomly generated intervals during the run. This is discussed in more detail in 5.2.

3.3 Representation

The individuals within the population are formed by a concatenation of three substrings, each representing one of the decisions required.

1 0 0 1 0 1 1 0 0 1 | 2 1 9 6 7 8 3 5 4 10 | A B B C A E D B C D
0 0 1 0 1 1 0 1 1 1 | 7 3 5 10 8 2 6 4 1 9 | E C D A D A B B E C

represents: {Batch sizes | sequence of batches | allocation rule}

Figure 4 - Examples of two-part individuals

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, from the set of all possible batches that may be combined to complete the total campaign. This is discussed in more detail in [21]. The permutation part of the individual represents the sequence in which the batches will be made. The third section represents the allocation rules. These are further discussed in section 4.1

3.4 Schedule Building

Translating the individual into a working schedule requires an encoding, or "schedule builder" stage. From the list of sequenced batches, the relevant allocation rule is used to assign each batch to the appropriate unit for the relevant procedure to be performed. This process builds a completed schedule from which the objective costs may be evaluated.

The handling of the objective function by the MOGA has been determined by previous work [21]. It was demonstrated that by implementing a five-objective MOGA and treating each of the five objectives separately, rather than using a combination of the plant costs to decrease and simplify the number of objectives to just two, the scheduling performance was improved. Thus each objective is evaluated from the completed schedule separately. The MOGA itself uses Pareto ranking-based fitness assignment [5]. The idea of Pareto optimality is critical in providing the most useful effect of multiobjective optimization. Any individual that is found to have a set of objective costs that are non-dominated by those of the others in the population, may be a candidate optimal solution to the problem. It is rare that a single point is found which dominates all others, providing a single optimal solution. On the contrary, the user is presented with a set of possible solutions, which cover the range of possible trade-offs between objectives. Without additional user specification as to the priorities of each objective required in the final solution, all contenders are equally valid. In terms of scheduling, this allows the user to maintain a population of (near-)optimal solutions to meet a range of objective preferences that might match various real-life situations. In this example, there might be a requirement to meet deadlines (cost 4) with the highest priority, as a trade-off for raising the % reactor wastage within the schedule (cost 3), or minimizing the cleaning times (cost 1), yet raising the amounts in storage by doing so (cost 2). Such priorities may be set by the user, as discussed in more detail in Changing the objective preferences.

Thus by applying Pareto-based selection to the population, non-dominated individuals are more likely to be selected in order to recombine, and possibly mutate, according to the effects of all five objectives. Standard binary and permutation crossover and mutation operators can be used on relevant parts of the population. The resulting individuals replace the previous generation of solutions, and the process is continued until a stopping criteria, in terms of a set number of generations, is achieved. Mean and minimum fitnesses are tracked for each cost, and the set of resulting non-dominated solutions is stored, at each generation. It should be noted that at any point during the MOGA run, the current set of solutions may be provided as

output schedules to be implemented instantly, whether the optimization process has been completed or not; they will still provide valid schedules. This may be important in an environment in which time is highly constrained, and in which fast reactions are critical. The GA code is implemented in GA MATLAB Toolbox [23] running on a SPARC station (Sun Ultra 5, 333 MHz, 128 Mb RAM).

3.5 Decision making within the MOGA system

There are many decisions being made within the scheduling system, by a combination of the human and MOGA, and defined in some cases by the physical plant conditions. Specifically, the batch sizing, sequencing, and allocation choices are defined by a combination of the plant, MOGA and human. The human and MOGA each specify properties and decisions relating to the reaction to change within the plant and the allocation rules. The human alone dictates to the MOGA values for the relative priorities of the objectives, including the importance placed on feasibility of solutions within the population.

It should be stressed that many of the human's decisions may be made interactively during the scheduling process, either in reaction to the plant's behaviour, or as a result of the output scheduling results from the MOGA process at that generation. Simultaneously, the MOGA is itself evolving and updating its own decision-making process. Section 4 illustrates the development of the experimental design of the schedule builder that allows the MOGA to evolve its decision making rules regarding the specific assignment of products to units.

4. EXPERIMENTAL DEVELOPMENT OF EVOLVING ALLOCATION RULES

4.1 Allocation Rules

In previous work, [21], static allocation rules – effectively fixing the third part of the population to one element throughout - were used to make the choice of the assignment of job to unit. Yet, as Fang, Ross and Corne, [9], explain, 'there is no good reason to rely on a fixed heuristic for each choice of operation when building a schedule. Indeed, it is quite easy to see that varying the choice of heuristic according to the particular job being processed and according to the particular stage in the schedule building process, may make more sense". Their 'Evolving Heuristic Choice' technique provides superior solutions to any previously found for the open-shop scheduling benchmarks used.

In addition, marked effects of static rules within the schedule builder were seen in the resulting schedules, in that the solutions seemed to concentrate on specific areas of the solution space. Following the lead of previous work [9], [10], the static rule allocation is removed, and an evolving schedule builder (denoted "SBE") is developed. The population includes representation of a set of possible rules, each one dictating the method in which the related task should be assigned to a unit. The five rules available for this system offer the option to choose the unit

(A) with the shortest processing time,
(B) with the longest processing time,
(C) at random;
(D) that will be ready earliest;
(E) that will be ready last.

The rule that allows a random choice of unit is motivated by previous results [10] which illustrate that a schedule building rule which allows a certain amount of random choice in the allocations is actually beneficial to the optimization goals. In addition, the inclusion of a random element mirrors real-life: "a simple (rule) often used in practice, is the "service in random order" (SIRO) rule. Under this rule, no attempt is made to optimize anything." [22].

Previous implementations of the allocation process of tasks to units had used a single, static rule throughout the process; either 'use the fastest unit' (in a schedule builder hereafter termed 'SB1'), or 'use the earliest available unit' ('SB2'). These were implemented within full MOGA scheduling systems, and the optimization performance of the two systems compared, with neither outperforming the other [21]. However, it was suggested that SB1 might reflect more realistic practice within the factory. A new schedule builder, which allowed a single rule to be chosen for each allocation ('SBE'), was implemented within a MOGA designed to be comparable to those used in the previous experiments. The MOGA was run eleven times, with a population of 100 individuals, over 100 generations. Standard binary and common permutation operators were used [2], [23], [24] for the relevant parts of the population. The mean and minimum values of each cost were tracked at each generation, as were the sets of non-dominated points found.

4.2 Summary of Results

Table 1 summarieses the minimum and mean costs found in all the final solutions from all eleven experimental runs of the MOGA incorporating the dynamic SBE system, compared to previous static rule, SB1 ("choose the fastest unit").

		Cost 1	Cost 2	Cost 3	Cost 4	Cost 5
SB1	Min	0.2000	0.0000	0.0000	0.0000	0.0000
	mean	10.5990	2544.7600	3.0906	127.0600	**2.7235**
SBE	min	0.2000	0.0000	0.0000	0.0000	0.0000
	mean	**9.6916**	**1774.1966**	**2.7168**	**112.1765**	2.9741

Table 1 – Summary of minimum and mean costs for each objective by schedule builder; minimum values are highlighted.

Both schedule builders attain the same values for the lowest minimum costs. However it seems apparent that SBE excels in finding the overall smallest values for the various costs when comparing the means. It is only for cost 5 that the advantage does not hold. The comparison of the methods can be made in more detail by use of the ACFM technique [20]. This allows comparisons of MOGA performance to be made, identifying areas at which methods may differ from each other with some statistical significance. In a comparison of SBE with SB1, areas of the final solution space can be identified at which the methods' relative performances differ significantly. Figure 5 illustrates these areas, using the method of parallel co-ordinates to allow the solutions to be plotted in five-objectives. Each line represents a single schedule solution to the problem that is significantly better than might be expected if the methods' performance were the same, at a 95% level of confidence. In the plots, the x-axis represents the five objectives, whilst the y-axis represents the normalized values that each solution attains for the objectives. Thus in Figure 5, comparing the performance of a MOGA containing SB1 with a MOGA containing SBE, the former method actually finds no solutions that are significantly better than the latter method.

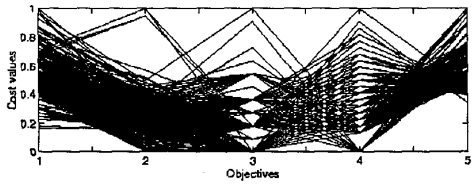

Figure 5 – Solution regions at which MOGA including SBE performs better MOGA including SB1

It is recommended, therefore, to use SBE, allowing the decisions as to how to allocate the tasks to the units to be made in conjunction to the decision as to where to allocate the tasks to the units. By adopting the GA search technique to allow it to evolve the allocation rules in conjunction with the other scheduling decisions, an improved performance in terms of objective value attainment and choice of possible solutions is achieved. In practical terms, the SBE can offer a wider range of better schedule solutions. The development of rules within the genetic algorithm naturally lends itself to a more extensive development of genetic programming within the schedule building process [8]. Additionally, analysis of the rules

developed within the best solutions found by the improved schedule builder may offer insights into the scheduling process itself, and provide suggestions of useful practice for the human scheduler. It is interesting to note that, perhaps contrary to immediate intuition, the provision of the 'choose slowest' and 'choose latest' options do not decrease the scheduling performance in any way. The solutions found do make use of these rules. The MOGA using SBE finds superior values for economically vital cost 4 (meeting the deadlines) by allowing the inclusion of these rules, compared to the previous static implementations, which did not. This may be explained by the fact that the occasional inclusion of idle times (in choosing the latest available unit), or delaying of task release times from one unit to the next (in choosing the slowest running unit), may be beneficial in terms of preventing bottlenecks and conflicts over equipment that may otherwise have taken place. Further examination and quantification of the actual patterns of rules may identify these bottlenecks or needs for idle times.

5. ADDITIONAL REACTION CAPABILITY

In addition to the MOGA's ability to adapt the set of rules used in the scheduling process to reach an optimal result, the system allows further flexibility by its handling of decisions regarding batch sizing rules, objective preferences and in its ability to begin to include a reaction to changing data within its evolving schedule solutions. These are summarised below.

5.1 Batch-sizing rules

The selection of the batch sizes into which the orders are divided is incorporated in the MOGA as discussed in 3.3. Previous results suggested that the MOGA exploits this provision of the a range of possible batch sizes in combination with the various types of schedule building systems in order to create a range of different types of solution. This deliberate usage of the variation in batch size rules moves the scheduling decisions beyond the most immediate objective, that of attempting to run the reactors at capacity by filling them only with maximum sized batches, as is the case in many real-life scheduling decisions.

5.2 Reaction to change within the plant

Real-life scheduling environments include frequently changing plant data; thus practical scheduling optimization techniques should reflect this. Genetic algorithms, with their use of a population of parallel solutions in solving a problem, seem ideally suited for coping with this problem. Individuals which may be are less fit under the previous conditions may hold advantageous evolutionary material as the definition of fitness changes with the new incoming data, and so it may be that good solutions to the new problem are still contained within the population. Cartwright, [25], comments, 'many schemata in the solution to the static problem are likely to be useful in deriving the solution to the dynamic problem, provided we have not allowed the constraints to change too dramatically before engaging in a recalculation'. For this reason, Husbands [26] cautions: 'in the dynamic situation, it is undesirable to allow the population to converge too strongly on a single solution; potentially useful partial solutions may be lost for good.'

At present, the system is designed to allow the user access to the data to be able to change or update the environment at any stage during the GA run, via the GUI shown in Figure 2, so that the MOGA can work with the most accurate and recent model. However, a change to the data will mean that the population, evolving to optimize the original model, will need a recovery period to include the reaction to the changed data. Experiments were undertaken to see how the optimization process would be damaged by the changes, whether the system could continue to find acceptable solutions by absorbing the new data, or whether it would be beneficial to simply reinitialise the MOGA with a new randomly-generated population. Bierwirth et al. [27] deal with 'temporal decomposition' of schedules. This means ascertaining which parts have already been made and rescheduling the remainder, showing adaptive scheduling on a rolling time basis. However, this current problem consists only of a finite set of orders, and to do this would mean the GA simply rescheduled a subset of the original orders after a set time. This addition suggests an immediate direction for future work.

5.3 Initial Experimental Work

A simulation demonstrates initial investigations as follows. Three events were chosen to reflect how the data in the model might be given cause to change. The first event included an event that had the effect of slowing the reaction times for one reactor. The second event caused changes in the times of the cleaning routines, some slowing down and others speeding up. The third event simulated a change in the ordered amounts for a particular product. The data representing each changed scenario could be supplied to the MOGA as it was running via the GUI, as the time the event took place. These events were triggered randomly throughout a set of MOGA runs. The results were compared to those of a set of equivalent MOGA runs with static data, to represent the final model after all the changes had taken place, providing control results of the MOGA's optimization performance of a run with unchanging data, after the events had taken place. Both sets of MOGA were run 11 times, using otherwise identical parameters. A particular interest of this simulation was to consider how the changes effected the individual optimization of each objective within the MOGA. Initial results are presented briefly below. Figure 6 shows a typical run, in terms of the evaluated population mean and minimum costs per objective, in which the changes were implemented.

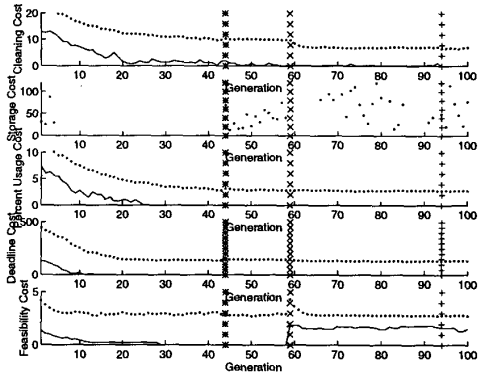

Figure 6 – Example of the effects of the events upon the MOGA performance. Key: : - mean performance – minimum performance, * - reactor change; + -cleaning routine; x- orders changing.

The effects of the events and resulting changed model data can be seen as the mean and minimum values for each cost are perturbed by the alteration; yet the population is able to continue to work and evaluate the fitnesses of the individuals according to the new objective function without losing the direction of its search. In the above and other runs, Cost 5 (the feasibility cost) appeared the most influenced in terms of ability to continue the search. Examining the effects of the changes in relation to the individual costs may suggest improvements that can be made to the MOGA in order that it might react to change with as little disruption to the optimization performance as possible. The benefits of treating each cost separately by MOGA may be seen again, regarding the method's ability to react. Had the five objectives been combined into a single-objective function, the fifth objective's inability to adapt would influence the total optimization process for all five objectives, rather than the single fifth objective relating to this cost alone as in the MOGA. Thus it is suggested that the changes in data cause less damage to the performance via use of the MOGA.

Additional measures may be needed in order to boost the performance regarding the fifth cost, in particular, such as *hypermutation* or the introduction of random immigrants [28], after a change has taken place, in order to improve the ability of the MOGA to react. Further investigation into methods for

handling change will be the subject of additional work. In addition, the MOGA's implementation offers the ability to change the priority placed on each objective, a property that may allow the search to be focussed upon the objective which has most difficulty recovering; in this case, objective 5.

5.4 Changing the objective preferences

The ability of the MOGA to allow interactive alterations to the users' specified preferences regarding the relative importance of the objectives is well documented [5, 10]. This ability may be used for a variety of reasons. The user may focus on an area of interest developing from the interaction and trade-off between the objectives, whether as a reaction to the immediate factory situation, or from a design point of view, to receive insights into the interaction of the objectives involved. For example, the user may decide to raise the priority of meeting the due-time of a particular customer's orders (cost 4), at the cost of paying extra for the other customers' orders to be stored (cost 2), lowering the % usage objective (cost 3), or requiring a change in the emphasis on shortening cleaning times (cost 1). As also suggested above (Initial Experimental Work), the MOGA may be directed to concentrate on the optimization of a particular objective above the others. The user can find the solution from the selection presented by the MOGA that reflects this new situation. This ability and its effects on the scheduling environment are discussed in more detail in [10].

6. DISCUSSION AND CONCLUSIONS

The MOGA offers a powerful system for optimization of the various conflicting and disparate objectives found in the complex batch process scheduling problem presented in this work. This power is particularly due to the flexible and interactive properties offered by the implementation. This becomes increasingly significant as the plant model becomes more complex. The inherent flexibility involved in increasing the degree of decision variables will continue to pose extremely difficult schedule problems.

The natural selection properties of the MOGA lend themselves to problem-specific variants, which may improve optimization performance and realism of the model further. In this example, the evolving schedule builder, SBE, allows the set of rules used to be evolved during the run, rather than set *a priori*, improving the optimization performance. By combining this ability for variation with the clearly defined standards set in S88, the MOGA may continue to develop whilst remaining relevant to the batch schedule process industry, in a portable and non-ambiguous manner.

The MOGA seems suited to coping with the changing data commonly found in scheduling environments by its parallel population nature; the fact that the MOGA treats objectives separately allows individual treatment of the effects of the changes on each objective. Immediate future work includes the extension of the model to include data relating to the complete plant, the fuller implementation of reactive rescheduling, and development of techniques for handling change specifically within the MOGA. The user may interact with the optimization process in several ways, and for several reasons, via the GUI, which provides communication between the MOGA, plant model and expert human decision-maker. The overall effects and advantages of this method will be seen in future work.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of EPSRC grant GR/L73517 for this work.

REFERENCES

[1] Holland, J., 1975. Adaptation in Natural and Artificial Systems, University of Michigan Press

[2] Davis, L., 1985. Job Shop Scheduling with Genetic Algorithms, Proc. First Int. Conf. GAs, ed. J. J. Grefenstette, Lawrence Erblaum.

[3] Schaffer, J. D., 1985. Multiple Objective Optimization with Vector Evaluated Genetic Algorithm, Proc. an Int. Conf. GAs, 93 - 100.

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

[5] Fonseca, C. M., Fleming, P. J., 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, Proc. Fifth Int. Conf. GAs, ed. S. Forrest. Morgan Kaufmann Publishers.

[6] Horn, J., Nafpliotis, N, & Goldberg, D. E., 1994. A Niched Pareto GA for Multi-Objective Optimization. Proc. First IEEE Conf. Evolutionary Computation, 1994.

[7] Srinivas, N. & Deb, K., 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation, Vol. 2, No. 3, 221 - 248.

[8] Langdon, W., 1996. Scheduling Maintenance of Electrical Power Transmission Networks Using Genetic Programming, Proc. GP-96 Conf., John Koza ed., Stanford Bookstore.

[9] Fang, H. L., Ross, P. & Corne, D., 1993. A Promising Genetic Algorithm Approach to Job-shop Scheduling, Rescheduling and Open-Shop Scheduling Problems. Proc. Fifth Int. Conf. GAs, ed. Stephanie Forrest. Morgan Kaufmann Publishers

[10] Shaw, K. J., & Fleming, P. J., 1997. Including Real-Life Problem Preferences in Genetic Algorithms to Improve Optimization of Production Schedules, Proc. GALESIA '97, Glasgow, 2-4 Sept.

[11] 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.

[12] Love, J & Bunch, M., 1998. Decomposition of Requirement Specifications for Batch Process Control, Trans. IChemE, 76, 8, 973 - 979.

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

[14] Viennet, R., Fonteix, C, & 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, 1995.

[15] Tamaki, H., Kita, H., & Kobayashi, S., 1996. Multi-Objective Optimization by Genetic Algorithms; A Review. Int. Conf. Evolutionary Computation '96. 517 - 522.

[16] Shaw, K. J., & Fleming, P. J., 1996. An Initial Study of Practical Multi-Objective Production Scheduling, Using Genetic Algorithms. Proc. Int. Conf. Control '96, University of Exeter, Sept. 2 - 5, 1996.

[17] Niemeyer, G., & Shiroma, P., 1996. Production Scheduling with GA & Simulation, Parallel Problem Solving from Nature 4, 930 - 936.

[18] Ishibuchi, H, & Murata, T, 1998. A Multi-Objective Genetic Local Search Algorithm And Its Application To Flowshop Scheduling, IEEE Trans. on Systems, Man & Cybernetics, Part C (Applications & Reviews), 28,3.

[19] Coello, C. A., 1999. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems. (Accepted for publication).

[20] Shaw, K. J., Fonseca, C. M. & Fleming, P. J., 1999. Developing Performance Comparison Techniques for Multiobjective Genetic Algorithms, Research Report 724, ACSE Department, University of Sheffield; submitted to Evolutionary Computation J., March 1999.

[21] Shaw, K. J., Nortcliffe, A. L., Thompson, M., Fonseca, C. M., Love, J. & Fleming, P. J., 1999. Assessing the Performance of Multiobjective Genetic Algorithms for Optimisation of a Batch Process Scheduling Problem, Congress of Evolutionary Computation, CEC99, Washington DC, July 1999.

[22] Pinedo, M, 1995. Scheduling: Theory, Algorithms and Systems. Prentice-Hall.

[23] Chipperfield, A. J. Fleming, P. J., & Pohlheim, H., 1994. A genetic algorithm toolbox for MATLAB, Proc. Int. Conf. Systems Engineering, Coventry, UK, 200 - 207, 1994.

[24] Oliver, I. M., Smith, D. J., & Holland, J. R. C., 1987. A Study of Permutation Crossover Operators on the Travelling Salesman Problem, Proc. Second Int. Conf. GAs and their Applications, ed. J. J. Grefenstette. Lawrence Erblaum Publishers

[25] Cartwright, H. M., 1994. Getting the Timing Right – The Use of Genetic Algorithms in Scheduling, Applications of Modern Heuristics, ed. V. J. Raymond-Smith, 1994.

[26] Husbands, P., 1993. An ecosystems model for integrated production planning, Int. J. C.I.M., 1993, 6, 1-2, 74-86.

[27] Bierwirth, C., Kopfer, H., Mattfeld, D. C., & Rixen, I., 1995. Genetic Algorithm based Scheduling in a dynamic manufacturing environment, Proc. IEEE Conf. Evolutionary Computation, 439 - 443

[28] Cobb, H. G., & Grefenstette, J. J., 1993. Genetic Algorithms for tracking changing Environments, Proc. Fifth Int. Conf. GAs, 523 - 529.