MULTIPLE CRITERIA SCHEDULING USING GENETIC ALGORITHMS IN A SHIPYARD ENVIRONMENT

David Todd and Pratyush Sen
EPSRC Engineering Design Centre and Department of Marine Technology, Armstrong Building, The University, Newcastle-upon-Tyne, NE1 7RU, United Kingdom.

ABSTRACT
The main aim of scheduling large manufacturing processes is to minimise the time taken to manufacture the product from start to finish, and maximise the utilisation of machinery/ personnel. However, as the number of tasks increases it becomes more difficult to find good schedules within a reasonable time. This paper explains the development of a Multiple Criteria Genetic Algorithm (MCGA) scheduler which evolves good schedules based on several criteria. The system is then demonstrated using a shipyard steel-shop problem.

INTRODUCTION
In most large scale engineering projects production scheduling plays an important role in making efficient use of resources and reducing manufacturing time. Research in scheduling has traditionally concentrated on using heuristics, ordered search or applying mathematical algorithms. However, as the size of the scheduling task increases so does the difficulty of finding good solutions within a reasonable time. It is common to split the task into a hierarchy (levels of scheduling), where the highest level considers the main events (milestones) and the lowest level is scheduling of workstations. This makes the project more manageable. The high level scheduling is performed first and is used to set targets for the next level down. If these targets cannot be met then re-scheduling occurs. If suitable schedules still cannot be found work may be sub-contracted or resources increased with associated cost penalties. For a major project three or four levels may be required.

In the case of Made-To-Order (MTO) products such as offshore structures and specialised marine vehicles there are additional constraints to handle. This increases the complexity even further. It is important to ensure that the processes of design, manufacture and assembly are integrated to their fullest extent to ensure the realisation of appropriate benefits. It is also often the case that orders are governed by strict deadlines with associated cost penalties. At this level of complexity new approaches to scheduling are required to generate good quality workable solutions. One such approach, increasingly being investigated and applied, is the use of Genetic Algorithms.

GENETIC ALGORITHMS (GA's)

The Genetic Algorithm (GA) is an evolutionary problem solving technique based on the Darwinian principle of survival of the fittest to obtain efficient solutions. It uses genetic operators like mating and mutation and was developed in the 1960's and 1970's by Holland [1]. However, it was not until the mid 1980's, with the increase in available computing power, that the technique started to be investigated more fully. Goldberg's [2] book outlines the simple GA as shown in Figure 1. GA's are extremely flexible optimisation tools in that they are able to avoid local maxima/minima as they start from multiple points. They can handle non-linearities and constraints and provide efficient search of large spaces using only a suitable representation of the problem and a 'black-box' evaluation mechanism. However it must be noted they are not guaranteed to always find the optimal solution.

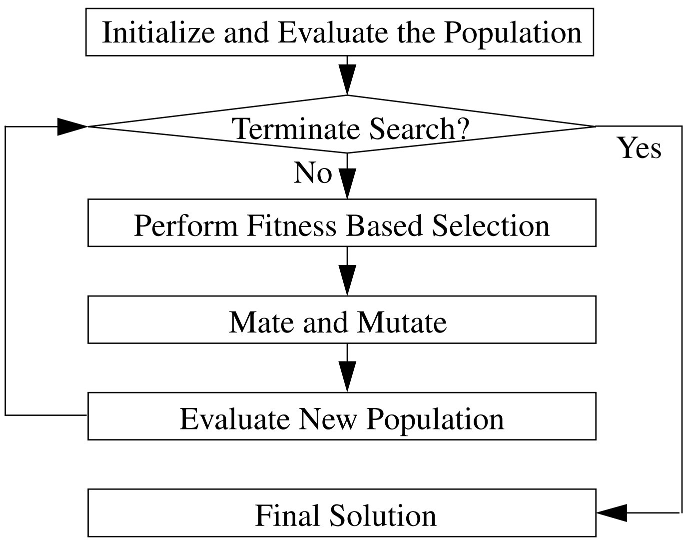

Figure 1. The Simple Genetic Algorithm.

The simple GA works as follows. A population of individuals of a fixed size is initially randomly generated. Each individual is characterised by a string of genes and represents one possible solution to the problem being tackled. Strings can then be evaluated in terms of a metric giving a measure of how good the solutions are. This is known as the fitness value. The solutions are checked to see if they have met the search terminating criteria. If this is not the case a loop is entered. Using the fitness value the population undergoes natural selection. The strings which are fitter (better) have more chance of surviving than other weaker ones. The surviving individuals are then paired up at random and mated (commonly known as crossover) with probability Pc to form new 'child' strings. The idea here is that the good properties from the two parents will combine to form an even better child. Mating is continued until a full population is generated. This population is then mutated with probability Pm where small random changes are made to the children in order to maintain diversity. The newly generated strings are then re-evaluated and given a fitness score and the process repeats until stopped. This is usually done after a fixed number of generations or terminated by a levelling out of the average population fitness. The best strings found can then be used as near-optimal solutions to the problem.

SCHEDULING WITH MULTPLE CRITERIA

Scheduling has commonly been tackled using long established Linear Programming, Monte-Carlo and Heuristic methods (Muth [3]). The aim of such procedures has commonly been to minimise makespan, the total time from the start of the first task to the end of the last. However, other important factors exist within a scheduling task. Regulated allocation of resources, limits on the allowed number of tasks in progress, the completion of tasks by specific deadlines, interrelationships between tasks and conflicts for shared resources are just a few examples. If we attempt to optimise combinations of these factors we will end up with a trade-off set of solutions each with their own balance of criteria. As a simple example, say we have a workshop with limited space available, thus limiting the amount of work in progress. We would like to minimise time taken to complete a set of tasks (the problem tackled later is minimising makespan and penalty costs). Three cases are shown in Figure 2 demonstrating samples from various cases in the trade-off set. If we specify the level of work in progress (x) which can be processed it can be seen that the a trade off solution (centre) gives the minimum makespan(t) for that level of work.

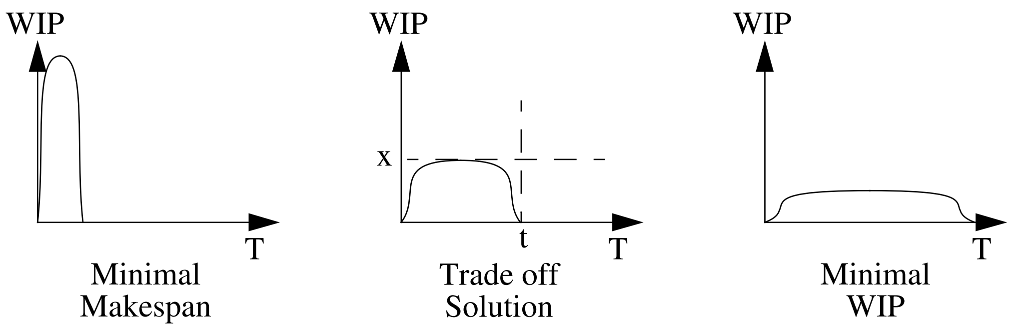

Figure 2. Time(T) versus Work in Progress (WIP) for Three Sample Trade-Off Solutions.

The trade-off can be more clearly shown on a trade-off curve (Figure 3). The solutions to the problem will be confined within a feasible region which is defined by system limitations and imposed constraints. Hence, the aim of scheduling in a multi-criteria context is to generate the trade-off boundary (also known as efficient boundary or Pareto front). This efficient boundary can then be used to choose the solution that has the most desirable balance of criteria for the given set of circumstances. The most desirable balance is obviously dictated by the priorities in question.

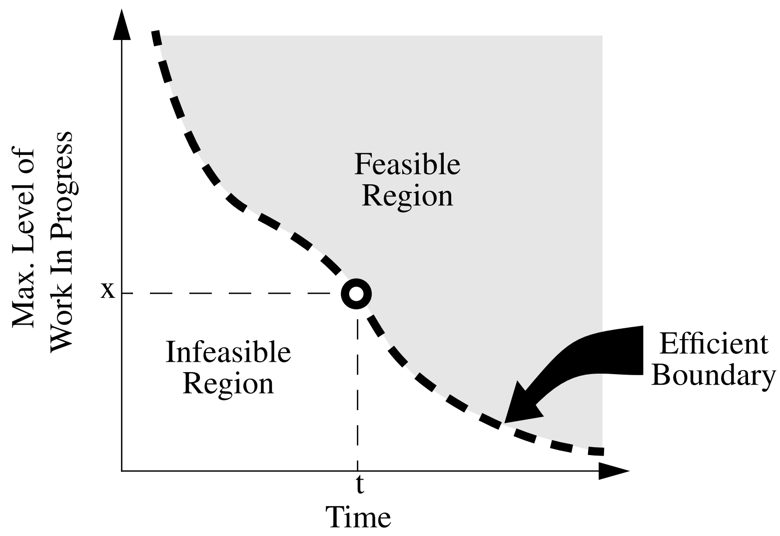

Figure 3. Jobs In Progress versus Time.

Scheduling plays a vital role in the integration of the design, manufacturing and assembly processes in order to meet deadlines (Figure 4). This has been recognised by the increasing use of concurrency in engineering practices. By increasing the level of overlap between stages of the product creation process it is possible to reduce lead times for products. This is especially important when tendering for contracts, or in consumer product industries where time to market gives a competitive edge. This increased integration does however introduce some complications. For example, in a manufacturing environment, the completion time of the design of a part will define the earliest start time of manufacturing of that part and must be scheduled accordingly. Also constraints in the assembly process mean that components have to be produced in accordance with a specific assembly plan to avoid delays. If space is limited a manufacturing plan can also be used to coordinate the delivery of raw materials for a given component with the start of manufacture of that component. Additionally, certain tasks may not be allowed to commence until others have finished. This leads to complex interdependencies between tasks.

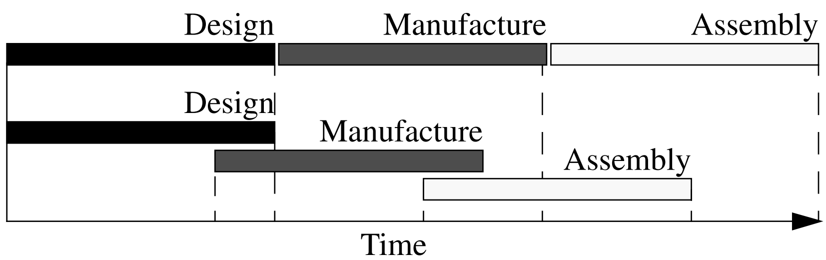

Figure 4. Increased Integration Leading to Reduced Lead Times.

Scheduling can also be used as a analysis tool. For example it may be used to investigate the viability of purchasing an extra machine or to identify and remove system bottlenecks.

All of the aspects mentioned in this section are essential to any useful scheduling process, and they are all addressed in this paper.

THE GENETIC ALGORITHM SCHEDULER

The use of GA's for scheduling tasks was first investigated by Davis [4] in the mid 1980's. He successfully optimised a simplified job shop scheduling problem with a single criterion. Since then there have been many other attempts at problems such as job-shop (Bruns [5], Cheng [6], Fang [7], Gen [8], Nakano [9], Reeves [10], Syswerda [11]), flow-shop (Cleveland [12]), line-balancing (Anderson [13], Kim [14]) and resource allocation (Easton [15], Tanomaru [16]) in different domains. More complex approaches by Liang [17] and Sridhar [18] use a weighted sum approach to combine disparate metrics to give a single fitness measure. Murata [19] uses a random weight system to generate a range of trade-off solutions and Hamada [20] addresses optimization with constraints. However, none of them combine multiple criteria considerations and constraints with the additional complexity of setup times, different machine operating speeds, maintenance periods, selection of alternative machines for certain jobs and complex job constraints and interdependencies.

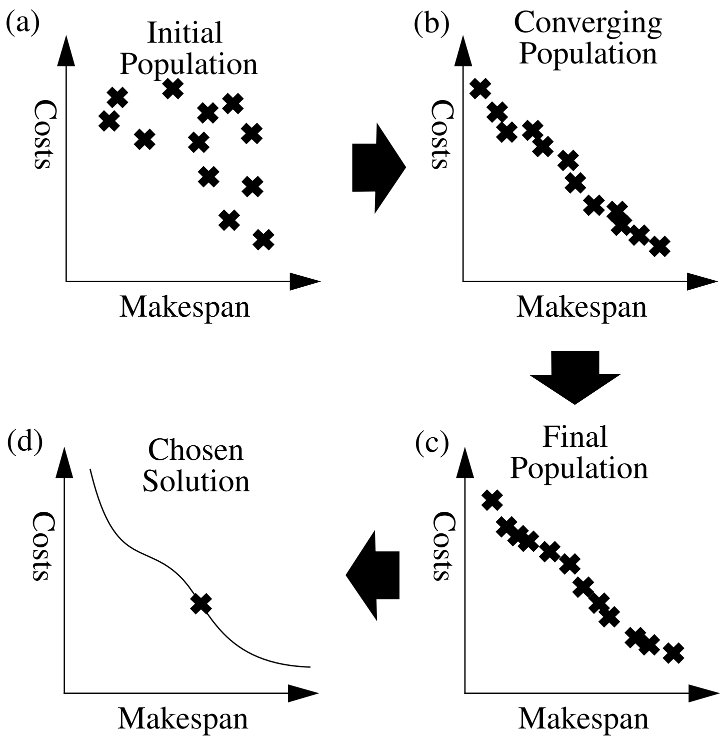

Figure 5. Evolution of the Trade-off Surface.

The Multiple Criteria Genetic Algorithm (MCGA) utilised here can be used to search for optimal solutions over any number of criteria and maintain full separation between each. This separation allows simultaneous maximisation and minimisation of individual criteria, in that it is possible to maximise profits whilst minimising makespan and work in progress. The algorithm, which is explained more fully elsewhere (Todd [21]), is based on the simple GA model (explained earlier) with several modifications. Instead of generating a single near optimal solution the MCGA evolves the trade-off surface. Figure 5 shows this process for the minimisation of two criteria, makespan and incurred cost. Initially we start with a random distribution of points (a), as the search progresses a trade-off surface begins to emerge (b). The GA continues until the surface is reasonably defined (c). From this set of solutions the most desirable solution can be chosen (d). This process is fully demonstrated later in the paper using the shipyard steel shop problem.

Several important aspects are involved in using GA's for scheduling problems. These are representation, the evaluation mechanism and the genetic operators (crossover and mutation).

Representation

The representation of the problem within a string is one of the most important aspects of constructing a GA. If this is not done in a suitable manner the string will not successfully reflect the problem being tackled, therefore leading to poor results. The string must also contain all of the information needed to specify the problem parameters fully. There are two classes of representation in scheduling, direct and indirect encoding. Direct encoding can be decoded straight into a schedule, whereas the indirect approach needs to be processed through some intermediate stage before it is converted into a schedule. Cheng [6] gives a comprehensive summary of these methods.

The scheduler developed here uses an indirect representation in a novel form for ease of decoding and for storing information on machine choices. The most common indirect GA representation used is the encoding of the string as a list of jobs wherein the ordering dictates the output schedule. The ordering of these jobs is then changed within each string and the strings are decoded for evaluation. In order to cope with machine choice, or choosing which task is to be performed by which machine, the string is constructed in a slightly more complex way.

The string is loosely based on the Structured GA (Dasgupta [22]). It is a set of integers which is split at certain points. The string construction is shown in Figure 6 with a simple example where the subscripts are used for clarity. There are three jobs 1, 2 and 3. Job 1 has two stages and jobs 2 and 3 have 3 stages. If we have two machines 0 and 1 we can specify that stages 1_1, 2_1, 2_3, 3_1, must be completed by machine 0 and stages 2_2 3_2 must be completed by machine 1. We may also specify that the remaining two stages, 1_2 and 3_3, can be put on either machine. We now need to make a choice about which machine 1_2

and 3_3 go on. This choice is fully encoded within the string. The string is constructed in sections, the first is the Machine Choice section, there are two stages for which a machine choice has to be made so there are two genes, the first corresponds to stage 1_2 and the second to 3_3. The rest of the string is made up of ordered job lists for each machine. In each of these lists all jobs which could be completed by that machine are listed, hence stages 1_2 and 3_3 will occur on both machine lists. This string is in the form that is manipulated by the MCGA.

For evaluation the string is decoded by separating the string into its constituent parts. The machine job lists are then drawn up. Using the machine choice genes the jobs which the genes correspond to are removed from all unselected machines. In this example machine 1 is chosen in both cases hence 1_2 and 3_3 will be removed from all other machine lists. The constructed machine lists indicate the priority order of the jobs to be tackled, i.e. machine 0 will carry out 2_1 then 1_1 ... etc.

.

List of Jobs :

Job 1:	1_1	1_2	
Job 2:	2_1	2_2	2_3
Job 3:	3_1	3_2	3_3

String Configuration :

Machine Choice		Machine 0						Machine 1			
1	1	2_1	1_1	1_2	2_3	3_1	3_3	2_2	3_2	3_3	1_2

Decoded Machine Lists :

Machine 0 Jobs:	2_1	1_1	2_3	3_1
Machine 1 Jobs:	2_2	3_2	3_3	1_2

Figure 6. String Representation and Decoding.

Evaluation

The decoded machine lists are then evaluated. Evaluation can be carried out via simulation or using an approximate system model. In the case of the MCGA a simulation has been chosen. A simulator provides more accurate results than approximate models. It can also be used to provide additional data such as machine utilisations and accurate job start and finish times. However it does this at the expense of a longer total run time.

The simulator is based on a clock system and will always produce a valid schedule. At every clock interval the system looks at the current state of all the machines. If a machine is idle the ordered job list for that machine is examined and the job closest to the front of the queue that has not yet been completed and is available is processed. Job availability is based on whether all previous stages of that job have been completed and the interdependency requirements have been met. The type of job is then compared with the current setup of the machine. If the two are incompatible the machine undergoes a setup period for a pre-defined number of clock intervals then the job is placed on the machine. The machine in question is then set to busy for as long as it takes to complete that job. Each machine also has an associated time scaling factor. This has been included to allow the system to analyse the effect of modifying machinery or even replacing older slower machines. For example a fast machine may be able to perform a task in half the time of slower machine, hence the fast machine will multiply the standard job time by 0.5. The simulator runs until all tasks have been completed. The results are then compiled for each criteria. Data on makespan, average length of jobs, number of deadlines met, incurred penalty costs, machine utilisations, average workload etc. can all be obtained. However, increasing the number of criteria increases the time taken to converge to good solutions.

Genetic Operators

The genetic operators (crossover and mutation) are the procedures that are carried out to manipulate and re-order the strings. The aim of crossover is to recombine the strings in a meaningful and constructive way. In order to do this the strings are separated into machine choice and individual machine lists (as in Figure 6). However the strings are not decoded into final machine job lists. When crossover is carried out between pairs of strings each of the separated string sections is only combined with the corresponding section in its partner string. The machine choice section is crossed using a simple two point crossover whereas within each of the machine job list sections a modification of the alternating edge operator (Grefenstette [23]) is used. Two-point crossover simply means that two random points are selected within the machine choice string section and the genes between these points are swapped between partners. In the machine job list sections of the string the two-point crossover operator cannot be used as it would introduce repeated jobs. The alternating edge operator is slightly more complex. However it ensures that repeated jobs are not created and jobs are not lost during crossover. When two partners are to be combined the first job of parent 1 is chosen. Then the job directly following this in both parents is found. The next job is chosen randomly to be one of these two jobs and this is then used to find the subsequent job in the same way and so on. If the chosen job is found to be a duplicate then the other of the two jobs is chosen. If this is also a duplicate then next job is chosen as the first job in parent 1's machine job list not yet used. This process is repeated until all the genes have been used. The whole procedure is then repeated for parent 2 using its first job.

Mutation is a much simpler operator. Mutation is used to introduce small random changes into the population in order to maintain diversity and prevent the population from converging to sub-optimal points. Again mutation is performed on separated sections of the string. In the machine choice section, if a gene is chosen it is randomly reset to another machine choice. For example if we have a choice of three machines 0, 2 and 3 for a specific job and the current machine selected is 2 it would change to either 0 or 3. In the machine job list sections a gene selected for crossover will simply be swapped with another randomly selected gene in that section.

THE SHIPYARD PROBLEM

The problem being tackled in this paper is a shipyard plate cutting shop problem. The aim of the exercise is to schedule the work on three machines, a Flame Planer (FP - cuts rectangular plates with weld preparation), a Shell Plate Cutting Machine (PCS - cuts curved plates with weld preparation) and a Dual Plate Cutting Machine (PC2 - cuts profiled plate parts with no preparation). The goals being to reduce the total time taken and comply as fully as possible with a specific assembly schedule. The steel shop is to produce all of the plates for a 24m section of a general cargo ship. This comprises a total of 144 plates of various sizes and cut patterns. The parts to be cut are deck/side/bulkhead plates, shell plates, longitudinal girders, transverse floors and brackets/small parts shown in Figure 7.

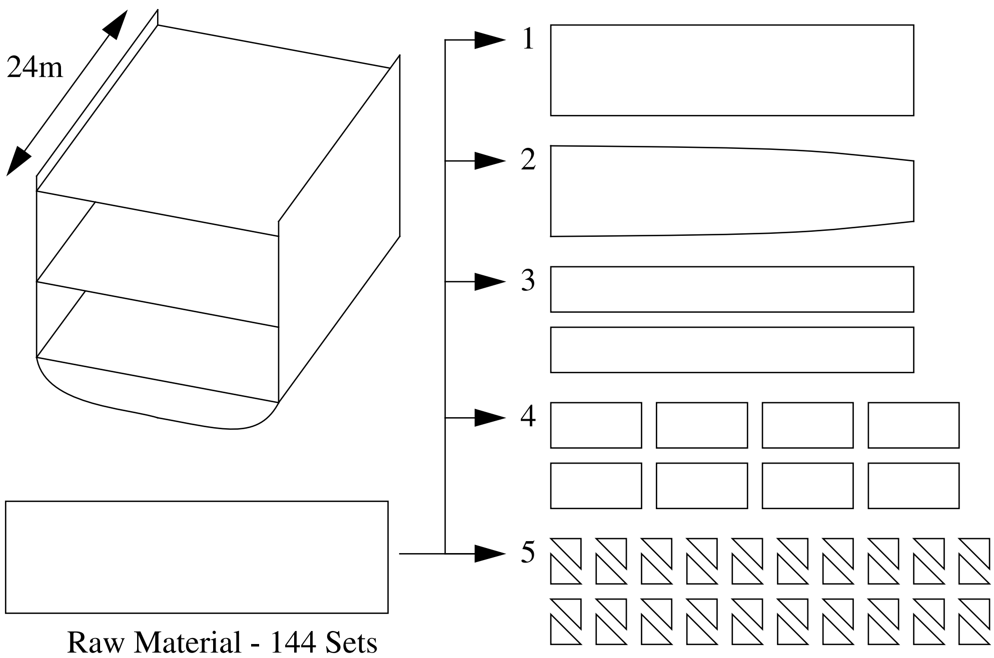

Figure 7. Plates Required.

The approximate time in minutes to process each piece of raw material is shown in Table 1. The complete loading-marking-cutting-unloading process is simulated. For example the components which are cut into more parts will take longer to mark and unload as well as longer to cut.

FP and PC2 operations can also be performed on the PCS hence most of the jobs can be placed on one of a choice of machines. The PC2 can cut two plates simultaneously. The machines also run at different cutting speeds. The FP, PCS and PC2 run at 400, 300 and 450mm/min respectively. This means that jobs can be done faster on some machines than others. The cutting times in the table are shown for a machine cutting at 500mm/min. A crane is needed to load and unload the plates/cut parts. As the crane is a shared resource some jobs may have to wait before they can be placed onto machines.

Table 1. Part Data Table.

Type	Quantity	Loading	Marking	Cutting	Unloading	Machine
1	64	15	5	60	5	FP/PCS
2	20	15	5	60	5	PCS
3	24	15	5	110	10	PCS/PC2
4	12	15	10	145	25	PCS/PC2
5	24	15	40	185	150	PCS/PC2

The closer the cutting can be matched to the subsequent assembly schedule, the less work in progress will be accumulated. Every manufactured part is given a due date obtained from the assembly schedule and an associated cost penalty for missing this due-date. This is proportional to the length of time that it is late. The penalties can be increased for critical parts whose delay will have a greater impact on the time of the assembly process.

RESULTS

The above scenario was run using the MCGA Scheduler. A typical run (Population Size 500, Number of Generations 250) takes just over 4 hours on an Sun UltraSparc. The aim of the problem was to minimise makespan (estimated at around 8500) and total penalty costs. Figure 8 shows a graph of the two criteria against the number of generations. It can be seen from the initial trade off set at generation 0 the quality and number of solutions has increased. It can also be seen that if we had minimised make span alone without the constraints high penalty costs would be incurred. One interesting point to note is that the specified assembly schedule cannot be met within the constraints of the system due to the fact that zero penalty has not been achieved. The assembly schedule could be modified but it is better to specify an ideal case assembly schedule and optimise towards it. By the same reasoning if an assembly schedule can be met by the MCGA tool then it is possible for the demands on the scheduler to be tightened in order to achieve further process gains.

From these graphs specific solutions can be chosen and these are further processed in order to generate process plans. The plans contain information on which jobs are placed on which machines, where the crane is in use, when setups are needed, and idle times for

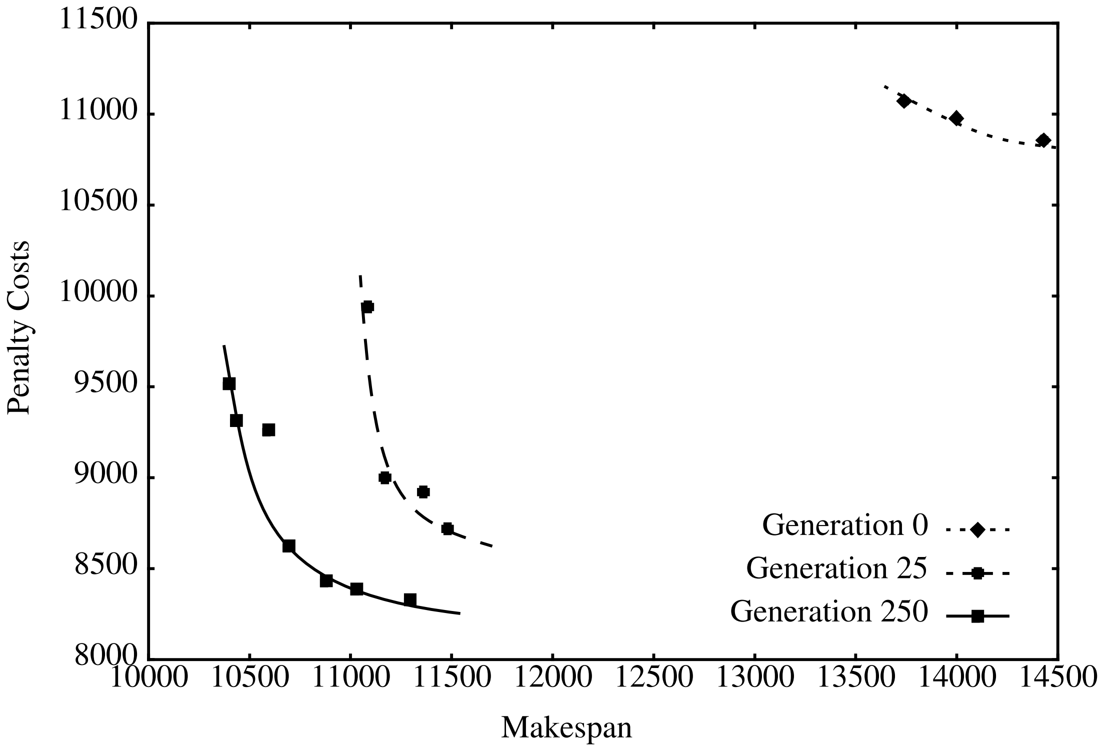

Figure 8. Evolution of Trade-off Solutions.

machines. It also gives information on machine utilisations and further data on job start and finish times and the incurred penalties. From these plans further insights into the process were obtained. The MCGA performed well in achieving a good balance of solutions which tried to optimise on both criteria. It was found that as makespan was minimised deadlines for jobs were discarded in favour of increasing utilisation. The use of crane was also more staggered as the scheduler tried to prevent the crane being required by two jobs simultaneously thereby reducing delays. In the other direction, minimising penalties, one of two things would occur. Either the scheduler would accept a small slippage in a large number of jobs or it would try to meet as many deadlines as possible at the expense of heavy delays on a small number of jobs. For the purposes of assembly the first case is more favourable as it would still allow the same assembly plan but slightly delayed. In the other case changes to the plan would be needed as a small number of components' due dates are delayed significantly therffore disrupting the assembly ordering.

An additional benefit of this tool is that it allows experimentation with various process setups. Further runs were carried out, one with the addition of a second crane, another with an added workstation for manual separation of plates and one with both crane and workstation present. In environments with shared equipment such as cranes it is often the case that bottlenecks form around these resources. It can be seen from Figure 9 that the benefits of adding a second crane are large. This is also demonstrated in the machine

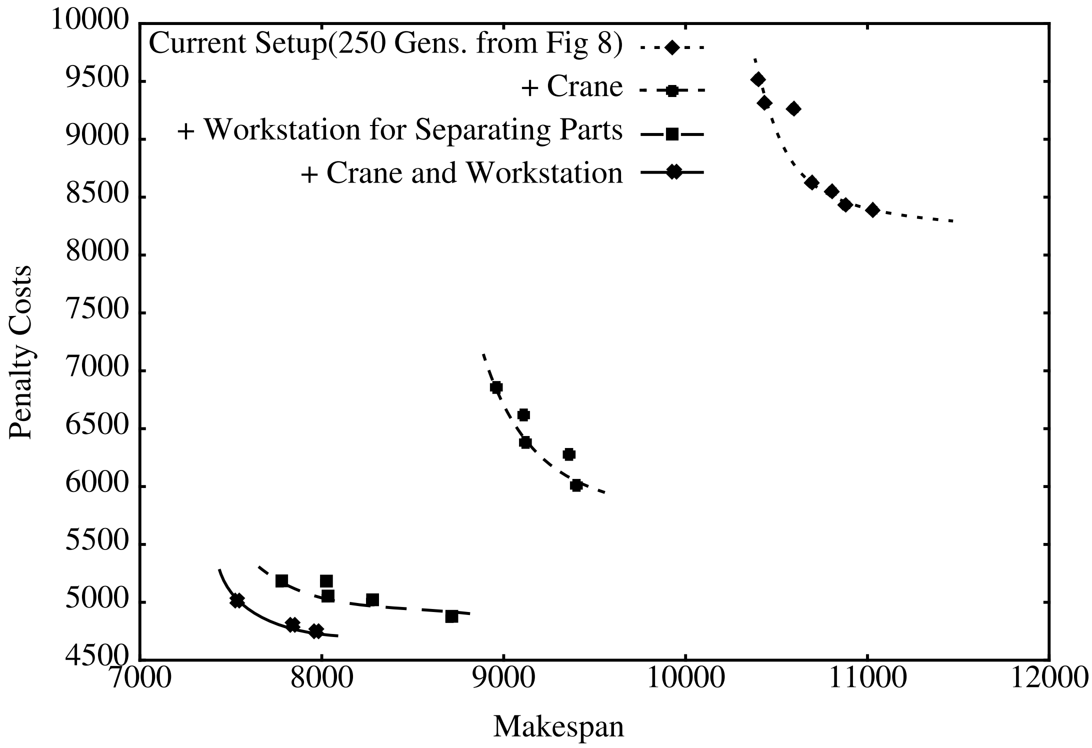

Figure 9. Analysis of Possible Setup Changes.

utilisation plots of Figure 10. In the first case with only a single crane there are many times when machines are waiting for a loading or unloading operation. When a second crane is added the occurrence of waiting machines is vastly reduced thus reducing total time for completion. On both graphs the number of machines in use drops to 2 part way through. This is because the Flame Planer runs out of jobs to perform. This free time could be used to start another project or build up a store of 'ready-cut' parts in anticipation of future work.

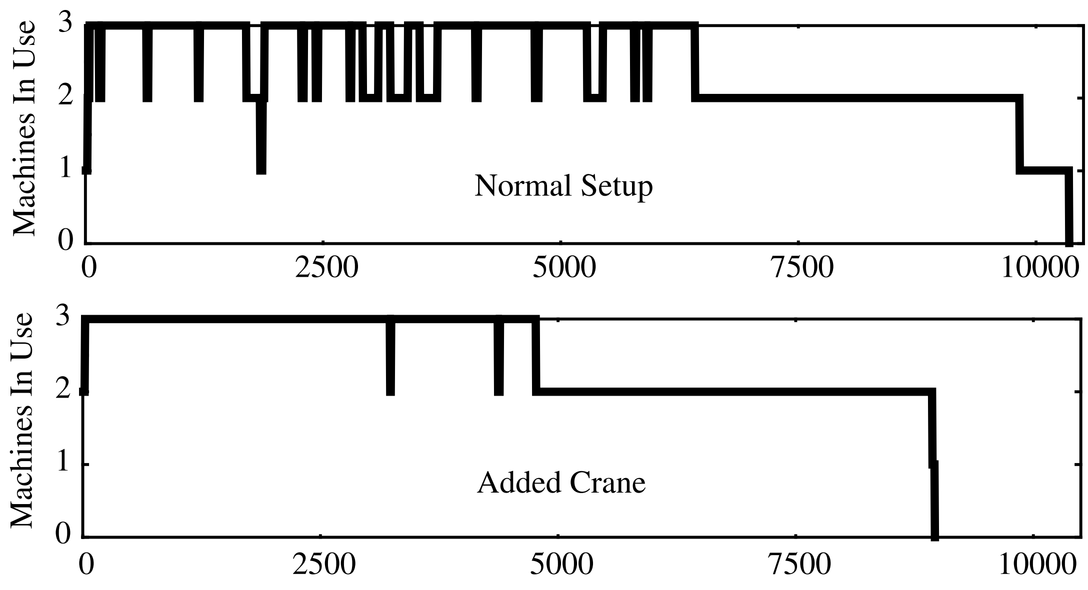

Figure 10. Utilisation of Resources.

Returning to figure 9 further performance gains can be achieved by adding a workstation for separating parts. When this is done parts are not fully cut on the machines instead they are separated on an additional workstation. This saves a lot of crane time as only one plate needs to be unloaded from the machine rather than each of the separated plates. The added flexibility and speed allows the tasks to be completed in much less time with lower penalty costs. This kind of experimentation enables the user to asses the viability of possible machine upgrades or addition of extra machines.

FURTHER WORK

The tool developed works well on the problem at hand. However the MCGA scheduler still has potential for improvement. Fine tuning of the genetic algorithm and further trials on more test problems will improve the system run-time. It is hoped that reducing run-time will eventually allow pseudo real-time scheduling. If a new unexpected job requires processing, which occurs regularly in ship repair work, the scheduler can be reinitiated and re-run to find new process schedules including the new job. Non-linear penalties can also be introduced to prevent the scheduler delaying single tasks for a long time and encourage a distribution of small delays across many jobs. This could be done using 'time squared' penalties so that if a job is delayed for 3 time units it would incur a 9 unit penalty, 4 units a 16 unit penalty and so on.

Another concern is that schedules generated must be robust. This essentially means that if certain jobs overrun slightly they do not throw out the whole schedule and cause long delays. Probabalistic methods can be used to insert random variations of job times into the scheduler and also insert machine breakdowns. By sampling over several simulations the robustness of the schedules can be found.

Additional work also needs to be done to take into account realistic operating conditions. In the current example the cranes are assumed to take no time to traverse the steel shop, this is obviously not true. It has also been assumed that the machines work continuously whereas manpower time considerations must also be take into account. The schedules will have to reflect working hours and holidays to allow efficient use of personnel. In general, the requirement for additional features will depend on the priorities of individual applications.

CONCLUSIONS

This paper has shown that the MCGA Scheduler is a potentially powerful tool for process optimization and investigation. The MCGA scheduler maintains full separation allowing simultaneous maximisation and minimisation over several scheduling criteria. The system selects which machines to place certain jobs on and reorders those jobs to achieve good performance. The system allows deadlines to be recognised for each job along with associated cost penalties. Interdependencies between jobs can also be specified along with jobs with multiple stages. Additionally, the system only requires a fitness score for each criteria and no other information. However, for optimisations with larger numbers of jobs the evaluation times of the simulation increase and hence the GA search will take longer to run.

The scheduler provides information on the shape of the trade-off surface and the correlations between criteria. It also provides full schedule plans containing information on machine utilisations, start and finish times of jobs, idle times, setup periods and so on.

It is expected that with further enhancements and performance tuning the system will become a useful tool in both optimization and analysis of production schedules in other engineering fields as well as shipbuilding.

ACKNOWLEDGEMENTS

This research was funded by the Engineering and Physical Science Research Council (EPSRC) and was carried out within the Engineering Design Centre of the University of Newcastle Upon Tyne, UK. I would additionally like to thank George Bruce of the Department of Marine Technology at the University of Newcastle, for the providing useful feedback and the steel shop data used in this paper.

REFERENCES

1. Holland, J.H., Adaption in Natural and Artificial Systems, University of Michigan Press, 1975 (1st Edition); MIT Press, 1992 (2nd Edition), ISBN 0-262-58213-6.
2. Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing, ISBN 0-201-15767-5, 1989.
3. Muth, J.F. and Thompson, G.L., Industrial Scheduling, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1963.
4. Davis, L., Job Shop Scheduling with Genetic Algorithms, Proceedings of the First International Conference on Genetic Algorithms, pp136-140, Lawrence Erlbaum Publishers, 1985.
5. Bruns, R., Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling, Proceedings of the Fifth International Conference on Genetic Algorithms, pp352-359, Morgan Kaufmann Publishers, 1993.
6. Cheng, R., Gen, M. and Tsujimura, Y., A Tutorial Survey of Job-Shop Scheduling using Genetic Algorithms - I.Representation, Computers and Industrial Engineering, Vol.30, No.4, pp983-997.
7. Fang, H-L., Ross, P. and Corne, D., A Promising Genetic Algorithm Approach To Job-Shop Scheduling, Re-Scheduling and Open-Shop Scheduling Problems, Proceedings of the Fifth International Conference on Genetic Algorithms, pp375-382, Morgan Kaufmann Publishers, 1993.
8. Gen, M., Tsujimura, Y. and Kubota, E., Solving Job Shop Scheduling Problems by Genetic Algorithm, IEEE International Conference on Systems, Man and Cybernetics, Vol.2, pp1577-1582, 1994.
9. Nakano, R. and Yamada, T., Conventional Genetic Algorithm for Job Shop Problems, Proceedings of the Forth International Conference on Genetic

Algorithms, pp474-479, Morgan Kaufmann Publishers, 1991.
10. Reeves, C.R., A Genetic Algorithm for Flowshop Scheduling, Computers in Operations Research, Vol.22, No.1, pp5-13, 1995.
11. Syswerda, G. and Palmucci, J., The Application of Genetic Algorithms to Resource Scheduling, Proceedings of the Forth International Conference on Genetic Algorithms, pp502-508, Morgan Kaufmann Publishers, 1991.
12. Cleveland, G.A. and Smith, S.F., Using Genetic Algorithms to Schedule Flowshop Releases, Proceedings of the Third International Conference on Genetic Algorithms, pp160-169, Morgan Kaufmann Publishers, 1989.
13. Anderson, E.J. and Ferris, M.C., Genetic Algorithms for Combinatorial Optimization: The Assembly Line Balancing Problem, Management Studies Research Paper 11/91, Cambridge University, 1991.
14. Kim, Y.K., Kim, Y.J. and Kim, Y., Genetic Algorithms for Assembly Line Balancing with Various Objectives, Computers and Industrial Engineering, Vol.30, No.3, pp397-409, 1996.
15. Easton, F.F. and Mansour, N., A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems, Proceedings of the Fifth International Conference on Genetic Algorithms, pp360-367, Morgan Kaufmann Publishers, 1993.
16. Tanomaru, J., Staff Scheduling by a Genetic Algorithm with Heuristic Operators, IEEE Conference on Evolutionary Computing, Vol.1, pp456-461, 1995.
17. Liang, S.J. and Lewis, J.M., Job Shop Scheduling Using Multiple Criteria, Proceedings of the Joint Hungarian-British Mechatronic Conference, pp77-82, 1994.
18. Sridhar, J. and Rajebdran, C., Scheduling in Flowshop and Cellular Manufacturing Systems with Multiple Objectives - A Genetic Algorithmic Approach, Planning Production and Control, Vol.7, No.4, pp374-382, 1996.
19. Murata, T., Ishibuchi, H. and Tanaka, H., Multi-Objective Genetic Algorithm and its Applications to Flowshop Scheduling, Computers and Industrial Engineering, Vol.30, No.4, pp957-968, 1996.
20. Hamada, K., Baba, T., Sato, K. and Yufu, M., Hybridizing a Genetic Algorithm with Rule-Based Reasoning For Production Planning, IEEE Expert, pp60-67, October 1995.
21. Todd, D.S. and Sen, P., A Multiple Criteria Genetic Algorithm for Containership Loading, Proceedings of the Seventh International Conference on Genetic Algorithms, East Lansing, MI, Morgan Kaufmann Publishers, 1997.
22. Dasgupta, D. and McGregor, D.R., Digital Image Registration Using Structured Genetic Algorithms, Proceedings of SPIE - The International Society for Optical Engineering, Vol.1766, pp.226-234, 1992.
23. Greffenstette, J., Gopal, R., Rosmaita, B. and van Gucht, D., Genetic Algorithms for the Travelling Salesman Problem, Proceedings of the First International Conference on Genetic Algorithms, pp160-168, Lawrence Erlbaum, 1985.