Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 152 (2004) 215–225
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Production, Manufacturing and Logistics

The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure

P. Pongcharoen, C. Hicks *, P.M. Braiden

Department of Mechanical, Materials and Manufacturing Engineering, Stephenson Building, University of Newcastle, Newcastle upon Tyne, NE1 7RU, UK

Received 6 August 1999; accepted 12 August 2002

Abstract

In this paper, the development of a genetic algorithms based scheduling tool that takes into account multiple resource constraints and multiple levels of product structure is described. The genetic algorithms includes a repair process that rectifies infeasible chromosomes that may be produced during evolution process. The algorithm includes problem encoding, chromosome representation and initialisation, genetic operation, repair process, fitness measurement and chromosome selection. The data structure and algorithm are detailed step by step. The tool generates schedules that minimises the penalties caused by early and late delivery of for components, assemblies and final products. The method is applied using data obtained from a collaborating company that manufactures complex capital goods. It is demonstrated that the schedules produced perform significantly better than those produced by the company using a conventional planning method.

Keywords: Genetic algorithms; Optimisation; Production scheduling; Complex products

1. Introduction

Scheduling may be defined as "the allocation of resources over time to perform a collection of tasks" (Baker, 1974). Scheduling problems in their simple static and deterministic forms are extremely simple to describe and formulate, but are difficult to solve because they involve complex combinatorial optimisation. For example, if n jobs are to be performed on m machines, there are potentially $(n!)^m$ sequences, although many of these may be infeasible due to various constraints. Reeja and Rajendran (2000) stated that most research on job shop scheduling problems ignored assembly relationships arising from product structure. This is a particularly important issue in scheduling the production of complex capital goods.

The objectives of this paper are to:

(i) outline the development of a genetic algorithm based scheduling tool for scheduling

* Corresponding author.
E-mail addresses: chris.hicks@ncl.ac.uk, chris.hicks@newcastle.ac.uk (C. Hicks).

0377-2217/$ - see front matter © 2002 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(02)00645-8

the production of complex products with multiple levels of product structure and resource constraints;
(ii) describe the data structures and the algorithm used;
(iii) apply the tool using data obtained from a capital goods company;
(iv) identify appropriate levels for the genetic algorithm parameters by using a full factorial design of experiments.

2. Special features of capital goods manufacture and problem statement

The product structure of capital goods is normally deep and complex. The manufacture of capital product requires many assemblies, subassemblies and components. A typical product structure for a relatively simple product produced by the collaborating capital goods company is shown in Fig. 1. The root node represents the final product with the leaf nodes representing components. The other nodes represent assemblies and subassemblies.

The manufacturing and assembly processes involved in the production of capital goods are diverse in terms of technology, capacities and production volumes. The process routings are often long, requiring many different types of operation on many machines. Each operation may also consist of a number of activities of varying duration such as set-up, machining and transfer time. Set-up times are dependent upon the characteristics of components and their associated processes. The transfer times between successive operations are determined by nature of the component and by layout considerations. Capacity constraints create continual difficulties as the complex product structure and associated manufacturing routing complexity creates contention on resources. Furthermore, the manufacturing systems are subject to wide variations in demand and the requirement for particular manufacturing resources changes over time. The scheduling problem in capital goods industry is therefore characterised by multiple resources constraints and complex precedence relationships between operation and assembly sequences.

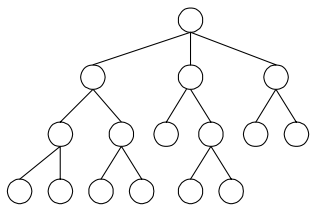

Fig. 1. An example of typical product structures from the collaborating company.

3. Genetic algorithms in production scheduling

The selection of algorithms to solve scheduling problems has found favour in the literature (Jain and Meeran, 1999). Algorithms may be classified into two types: conventional optimisation algorithms, such as integer linear programming (Manne, 1960), dynamic programming (Held and Karp, 1962) and branch and bound (Greenberg, 1968); or approximation optimisation algorithms including simulated annealing (Kolonko, 1999), taboo search (Nowicki and Smutnicki, 1996 and Ben-Daya and Al-Fawzan, 1998) and genetic algorithms (Tsujimura et al., 1997; Todd, 1997). The approximation optimisation algorithms tend to be most suitable for dealing with large combinatorial optimisation problems (Nagar et al., 1995).

Genetic algorithms (GA) are stochastic search techniques for approximating 'optimal' solutions within complex search spaces (Goldberg, 1989; Gen and Cheng, 1997). The technique is based upon the theory of evolution, in which the fitness of an individual determines its ability to survive and reproduce. Many authors including Croce et al. (1995), Tsujimura et al. (1997) and Lee et al. (1997) have applied GA to job shop problems. Tsujimura et al. (1997) examined job shop scheduling problems and produced an operation-based representation of chromosomes in which each gene uniquely indicates an operation. In common with other job shop studies, Tsujimura does not consider assembly operations or their relationship with other manufacturing processes.

In this research, the general procedure for GA (Goldberg, 1989) was modified. The modified GA

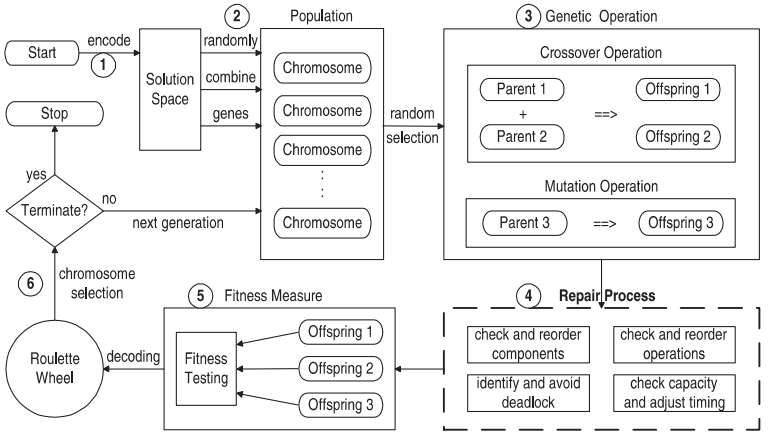

Fig. 2. The structure of modified GA for production scheduling.

program was written approximately 4000 lines of code in Tcl/Tk programming language (Ousterhout, 1994). The algorithm is illustrated in Fig. 2. A repair process was added in order to rectify infeasible schedules that can be produced by genetic operations.

3.1. Problem encoding

Genes may by represented by either numeric (binary or real), or alphanumeric characters. Blazewicz et al. (1996) suggested that the binary chromosome representation is often unsuitable for combinatorial optimisation problem because it is very difficult to represent solutions. In this work, all operations in the problem are encoded into genes represented by alphanumeric strings, which have three parts: a product structure identifier, a product instance identifier and the operation number. Fig. 3 illustrates the product structure identifiers. The root node refers to the product, which in this case has a part number of 1. There are three subassemblies with part numbers 2, 3 and 4 and product structure identifiers 1:2, 1:3 and 1:4. Part number 2 has components 5 and 6 with the product structure identifiers 1:2:5 and 1:2:6. The leaf nodes are components. This coding system uniquely defines the location of each type of part/assembly within the product structure.

However, in the general case it is possible for an assembly to include more than one item of the same type. Fig. 4 represents the situation where the product with part code 1 contains two identical assemblies with the part code 2. The product structure identifiers are the same as shown in

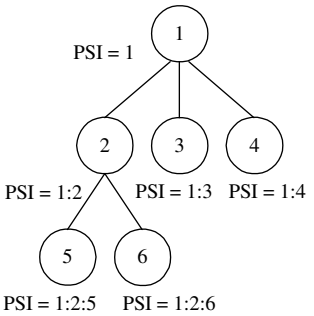

Fig. 3. Product structure identifier (PSI).

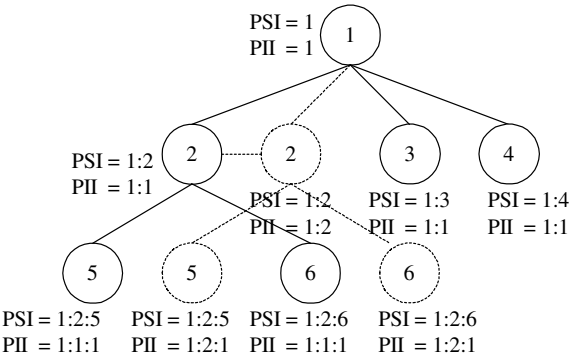

Fig. 4. Coding scheme of product instance identifier (PII).

Part code

Part name

Component / subassembly codes

Component /subassembly quantities

Operation sequence (set of resource codes)

Planned set-up, machining and transfer times

Fig. 5. Data associated with part code.

Product Structure Identifier (PSI)

Part Instance Identifier (PII)

Planned start times

Planned due date

Customer due date

Current operation

Fig. 6. Data associated with product structure and instance identifier.

Fig. 3. However, it is necessary to also consider the product instance identifier that distinguishes the different instances of identical parts.

Fig. 5 identifies the data associated with each type of part. The primary key is the part code that is used to access the data.

Fig. 6 shows the data associated with each instance of each part. The record is referenced by both the secondary keys (product structure identifier and the product instance identifier). The values of these variables are initialised to zero and then set by the GA planning process.

3.2. Chromosome initialisation

All genes are randomly sequenced to generate a chromosome. This can be repeated to generate a population of the desired size. Each chromosome is divided into n sub-chromosomes that represent the sequence of operations for n resources (see Fig. 7). The genetic operations are then performed on the same sub-chromosomes (machines).

3.3. Genetic operation

After a population of chromosomes has been generated, the next stage within the GA is to randomly select chromosomes that will be subjected to crossover and mutation operations. In crossover, the characteristics of two parents are combined to produce an offspring, whilst mutation produces random change in one chromosome. The probability of both must be specified. In this work, one point crossover (see Fig. 8) and inverse mutation (see Fig. 9) were used. The genetic operations are applied within each sub-chromosome to generate operation sequences for each machine.

One point crossover begins by randomly selecting a point within the chromosome that separates the parents into two sections. The first section is directly copied into the child from the first parent, the remaining genes are obtained from the second parent. The process is then repeated in reverse order to produce the second child.

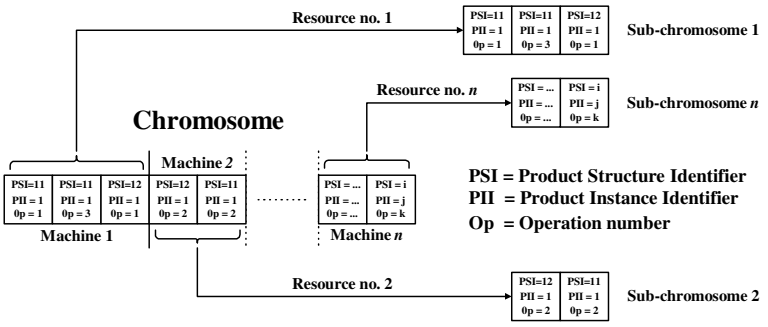

Fig. 7. Sub-chromosome representation.

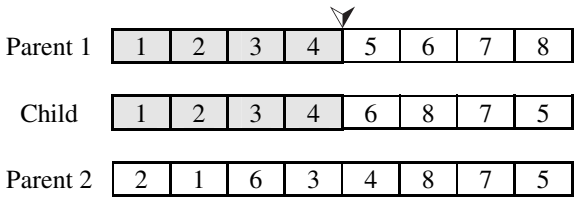

Fig. 8. One point crossover (Goldberg, 1989).

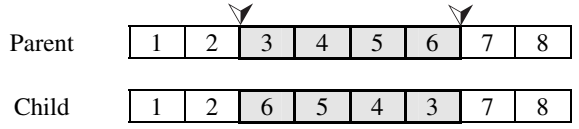

Fig. 9. Inversion mutation (Goldberg, 1989).

Inverse mutation starts by randomly selecting two points. Genes located between those two points are then placed in reverse order.

3.4. Repair process

After the genetic operation is performed, the chromosomes produced may represent infeasible schedules. For example, they may have impossible routings or assembly sequences. The general GA procedure may assign a very low level of fitness to such chromosomes so the probability of them surviving would be minimal. However, when a large proportion of chromosomes are infeasible, it is possible that only a few chromosomes will survive until the next generation. This would reduce the diversity of the populations. If all chromosomes are infeasible they would all survive with equal probability.

Test runs indicated that there was a very low probability of the genetic operations producing feasible schedules directly. A repair process was therefore introduced that consists of four stages.

- *Operation precedence adjustment*: this identifies impossible routings and converts them into a feasible sequence of operations. Operations must be performed in a logical order that is defined by component design and the manufacturing processes used. For example, in Fig. 10, the sequence of operations has the second operation on part 11 before the first operation. This stage of the adjustment process swaps these operations using the copying mechanism shown.

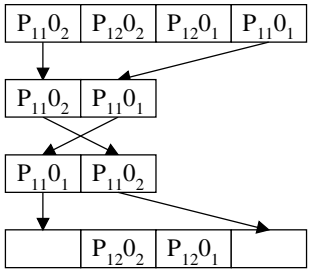

Fig. 10. Check and reorder operations precedence.

The procedure starts by checking number of operations required for the each part. If there is more than one operation, the sequence of operations needs to be checked. If the operations are not in sequence they are copied to a temporary string, they are then reordered in the correct order and then copied back into the chromosome. This procedure is repeated for all parts.

- *Part precedence adjustment*, which ensures that the supply of component parts and subassemblies is co-ordinated with their subsequent assembly. The procedure starts with a chromosome that may represent either a feasible or infeasible schedule. The chromosome consists of a set of sub-chromosomes that represent the sequence of operations on each resource. The part precedence adjustment is based upon the copying mechanism shown in Fig. 11, which is performed on each sub-chromosome. In this particular case, it is supposed to have four parts each of which require one operation. It can be seen that this sub-chromosome is infeasible as the operation on part 1 cannot be performed

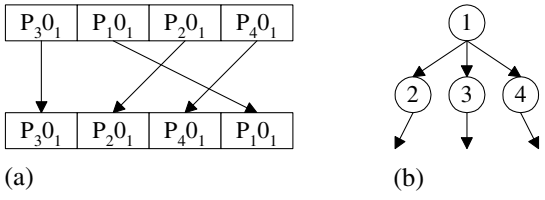

Fig. 11. Check and reorder part precedence: (a) copying sub-chromosome; (b) product structure.

until its components (part 2, 3 and 4) have been completed.

The procedure starts at the beginning of the sub-chromosome and checks the level of each item in the product structure using the product structure and instance identifiers as well as the operation number. Items at the bottom level are copied in the same order that they appear in the sub-chromosome. The procedure then repeats this process for each higher level of the product structure. This approach places operations associated with the lower levels of the product structure at the beginning of the sequence. Operations performed on items at higher levels will be copied to the end of the revised sequence.

- *Timing assignment and finite capacity considerations*: sequences of operations developed generation by generation within GA do not become schedules until all timing assignments are performed. These finite capacity assignments basically make sure that all machines perform only one operation at a time. Semi-active schedules are also introduced to minimise the idle time between operations. In other words, the successive operation should be performed as soon as possible after the predecessor on the same resource has been completed. When a part has operations on more than one resource, the schedule may need modification to ensure that capacity is not exceeded. In this case, the start set-up operation cannot take place until both the previous operation on the part is completed and the previous operation on the same sub-chromosome (resource) is also finished. A delay (idle) time may be introduced between operations due to resource capacity constraint. These timings are calculated using the sequences of operations on each chromosome generated by the GA, together with information on the duration of operations including set-up, machining and transfer time.
- *Deadlock adjustment*: when multiple resources are scheduled, it is possible that machine i is unable to perform an operation, as it is awaiting the part from machine j, whilst machine j cannot perform its operation as it is awaiting a part from machine i. This situation occurs for parts 11 and 12 shown in Fig. 7. The procedure for identifying the deadlock situation starts from the first operation of sub-chromosome 1. Operations that have no precedence relationship can be performed. They are therefore added to a list of legal operations. The logic then moves to the next gene within the sub-chromosome. If this is a second or subsequent operation a check is made to ensure that the previous operation appears in the list of legal operations. If the precedence constraint is satisfied the operation is added to the list. Otherwise the control moves to the next sub-chromosome when the process is repeated. At the end of this process a pointer will identify the first deadlocked operation on each resource. The program will then randomly select a sub-chromosome. The deadlocked operation is then swapped with its successive operation. The process is then repeated until all operations appear in the list of legal operations.

3.5. Fitness measurement

The next stage is to measure the fitness of the chromosomes, which evaluates the goodness of the associated schedule. In capital goods companies, delivery performance is an essential aspect of customer service. It is common for contracts to include severe penalties for late delivery. The early completion of components, assemblies and products results in increased inventory with associated holding costs. Early delivery may also inconvenience the customer. A scheduling method is required that minimises the total penalties due to both earliness and tardiness. It is also necessary to ensure that capacity constraints are not exceeded. Holding costs apply to raw materials, work in progress and finished goods. This includes final products, assemblies, subassemblies and components. The penalties for tardiness only occur when final products are delivered late.

Chromosome evaluation was done using the fitness function given in Eq. (1), which aggregates the penalty cost of both earliness and tardiness. An ideal schedule would have no penalty cost and a fitness value of zero. The objective function or

fitness function used in this work is to minimise total costs:

$$\begin{aligned}\text{Total costs} &= \text{sum of earliness costs for all components} \\ &\quad + \text{sum of earliness costs for all products} \\ &\quad + \text{sum of tardiness costs for all products} \\ &= \sum_{c} P_e E_c + \sum_{p} P_e E_p + \sum_{p} P_t T_p \end{aligned} \quad (1)$$

where subscript c relates to components, subscript p relates to products, $E_c = \max(0, D_c - F_c)$, $E_p = \max(0, D_p - F_p)$, $T_p = \max(0, F_p - D_p)$.

Notations: P_e, penalty rate of earliness (pounds per day); E_c, earliness of component (days); E_p, earliness of final product (days); P_t penalty rate of tardiness (pounds per day); T_p, tardiness of final product (days); D_c due date of component (date); F_c, finish time of component (date); D_p, due date of final product (date); and F_p finish time of final product (date).

3.6. Chromosome selection

The final stage is to select the same number of chromosomes that are included in the population for the next generation. The roulette wheel approach (Goldberg, 1989) was used for chromosome selection with a random number generator over the range 0–1. The probability of survival and number of replicates of a chromosome in the next generation is determined by its fitness. The GA process is repeated until the termination criteria are satisfied.

3.7. User interface

The user interface for the GA scheduling tool is shown in Fig. 12 that shows the schedule as a Gantt chart and Fig. 13 that shows the loading on resources. In the Gantt chart, it can be seen that there are two products that have different product structures. Each horizontal bar represents a part, with individual operations comprising set-up time (red), machine time (yellow), transfer time (blue) and delay time (light grey). In the monochrome figure the set-up, machining and transfer activities appear to be combined as a dark grey band. The vertical lines between the bars denote assembly relationships. In the resource utilisation chart, time is represented by the x-axis, each separate machine is shown on the y-axis, and loading is shown on the z-axis. Each vertical bar therefore represents the loading on a specific machine during a particular time period. It can be seen that loading of all machines does not exceed 100% because the repair process within the GA process has applied a finite capacity constraint.

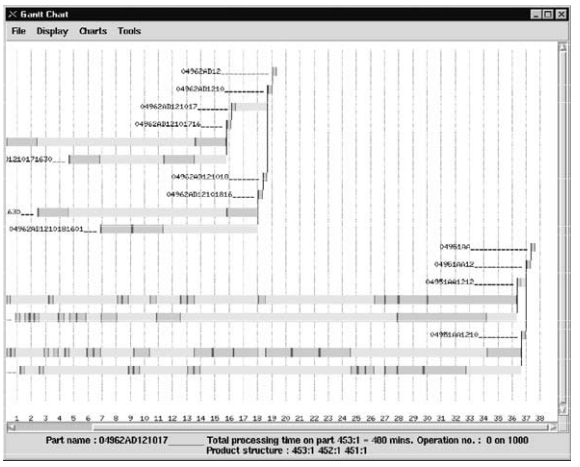

Fig. 12. Gantt chart for the best schedule obtained from GA.

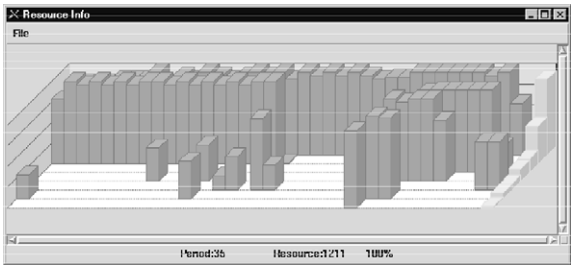

Fig. 13. Resource utilisation associated with the schedule.

4. Genetic algorithms applied to an industrial problem

Production schedules, product structure relationships, process plans and resource loading information were obtained from a collaborating company. Three representative planning problems were chosen (Table 1). All products had four levels

Table 1
The characteristics of the production scheduling problems

Problem sizes	Characteristics of the problems				Penalty cost (company schedule) £1000
	Products	Operations machining/assembly	Resources	Levels (product structure)	
Small	2	25/9	8	4	8.01
Medium	2	57/10	7	4	9.69
Large	2	118/17	17	4	11.30

in the product structure. The small problem involved two different products, with a combined requirement of twenty-five machining operations on eight resources with nine assembly operations. There was interaction and contention on resources. The schedule produced overload on several resources leading to the penalty cost shown. Under infinite capacity conditions it would be possible to achieve zero penalty cost in all cases.

Full factorial experimental design was employed. This is more efficient than one factor at a time experiments, and is necessary to avoid misleading conclusions when interactions may be present. This technique allows the effects of a factor to be estimated at several levels of the other factors, yielding conclusions that are valid over a wide range.

In GA the number of candidate solutions (chromosomes) in the population, the number of generations and the probabilities of crossover and mutation must all be specified in advance. The first two of these determine the execution time. All of the parameters influence the rate of convergence (divergence) and the quality of the solutions found. The influence of problem size, the number of generations, the number of chromosomes within the population and the probabilities of crossover and mutation were all considered.

Table 2
Experimental factors

Factor	No. of levels	Levels
Problem size	3	Small, medium, large
Number of generations	2	20, 40
Number of populations	2	20, 40
Probability of crossover	4	0.6, 0.7, 0.8, 0.9
Probability of mutation	5	0.02, 0.04, 0.06, 0.08, 0.1

The size of solution space is obviously related to the complexity of the problem. Test runs indicated that convergence was achieved within 20–40 generations. These values were therefore used for the levels of this factor. For crossover and mutation typical probabilities that move quickly towards 'optimal' solutions have been found to be 0.6–0.9 and 0.001–0.1 (Todd, 1997). The experimental design is shown in Table 2.

5. Identifying the requirement of the repair process

A typical run was analysed to investigate the effects of the repair process within the GA procedure. This used the large problem from Table 1 with 90% crossover, 18% mutation and a population of 60 chromosomes with 20 generations. The results are shown in Fig. 14. It can be seen that all chromosomes in each generation are repaired by the operation, timing and deadlock adjustment procedures. The number of chromosomes repaired by the part precedence adjustment drops from

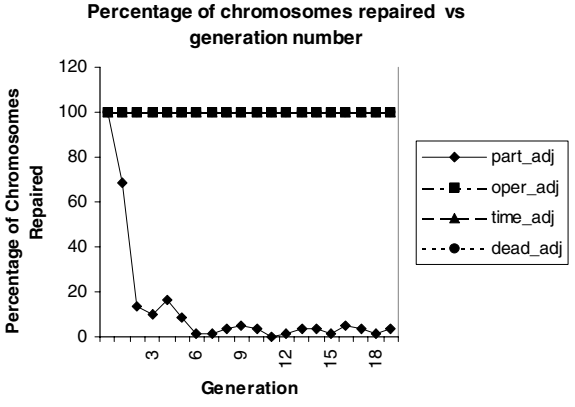

Fig. 14. Percentage of chromosomes repaired vs. generation.

100% in the first generation down to 1% by the seventh generation. In this particular example, the scheduling process is subject to many constraints due to the complexity of process routings, product structure and resource constraints. This demonstrates that the repair process is a critical aspect of the methodology for rectifying infeasible schedules.

6. Results

Analysis of variance (ANOVA) was used to investigate the effects of the main factors and their interactions. Table 3 shows the results of this analysis. For each factor the degrees of freedom (DF), sums of squares (SS), mean square (MS), F value (F) and probability (P) of incorrectly rejecting an important factor (type I error) are given.

For a given confidence level α, which is defined as the acceptable probability that a good model is incorrectly rejected, all factors or interactions with a value of $P \leqslant \alpha$ are statistically significant, whilst other factors may be disregarded.

All main factors, except the probability of crossover, have values of $p \leqslant 0.05$ and are therefore statistically significant, within the ranges considered. The first main factor, which is problem size, had a large effect on penalty cost. This factor also has important interactions with other factors. The quality of solution, measured in terms of the fitness function, is inversely proportional to the problem size.

The next two main factors, the number of generations and the population size, are also significant. The results show the fitness value for a population of 40 is better than that with 20. Likewise, 40 generations produces better results than 20. Execution time increases in direct proportion to both population size and the number of generations.

The fourth main factor, the probability of crossover, is not statistically significant when using the values chosen. The last main factor, the probability of mutation, is statistically significant and better solutions were obtained with values in the range 0.06–0.1.

The investigation of interactions between the factors is one of the main purposes of factorial designs. An interaction between two factors occurs when the effect of one factor differs depending on the level of another factor. It can be seen from Table 3 that there was interaction between problem size and all the other factors, except crossover, since $p \leqslant 0.05$. It was found that, changing from 20 to 40 generations, the penalty cost decreased for

Table 3
Analysis of variance for production scheduling data

Source	DF	SS	MS	F	P
Problem size	2	269.8500	134.9250	1189.86	0.000
Generation	1	2.4186	2.4186	21.33	0.000
Population	1	2.0986	2.0986	18.51	0.000
Crossover	3	0.0093	0.0031	0.03	0.994
Mutation	4	3.4500	0.8625	7.61	0.000
Problem size * population	2	1.6714	0.8357	7.37	0.001
Problem size * generation	2	1.6546	0.8273	7.30	0.001
Problem size * crossover	6	0.4870	0.0812	0.72	0.637
Problem size * mutation	8	5.1579	0.6447	5.69	0.000
Population * generation	1	0.0032	0.0032	0.03	0.867
Population * crossover	3	0.1138	0.0379	0.33	0.800
Population * mutation	4	0.2634	0.0658	0.58	0.677
Generation * crossover	3	0.1061	0.0354	0.31	0.817
Generation * mutation	4	0.7249	0.1812	1.60	0.177
Crossover * mutation	12	0.9135	0.0761	0.67	0.778
Error	183	20.7514	0.1134		
Total	239	309.6736			

* Denotes interaction.

Table 4
Mean and standard deviations of penalty cost for each level of each factor

Penalty cost (£1000)	Factors															
	Problem size			Population		Generation		Probability of crossover				Probability of mutation				
	s	*m*	*l*	20	40	20	40	0.6	0.7	0.8	0.9	0.02	0.04	0.06	0.08	0.10
Mean	1.36	2.09	3.88	2.54	2.35	2.54	2.34	2.44	2.44	2.45	2.45	2.60	2.56	2.39	2.28	2.38
S.D.	0.14	0.17	0.68	1.21	1.06	1.20	1.06	1.14	1.11	1.12	1.20	1.29	1.24	1.13	0.98	1.03
Best value	1.16	1.69	2.51	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16	1.16
Company schedule	8.01	9.69	11.3													
% Improvement	85.5	82.6	77.8													

the large problem, but this had little effect on the small and medium size problems. A similar interaction effect occurred when the size of populations was changed from 20 to 40. The reason for this is that the solution space is far bigger for the large problem. Increasing the number of generations and population, both increase the amount of search that enables the improved solutions to be identified. With the small and medium problem sizes, the solution space is smaller which makes it easier to find the 'optimum' result more quickly with fewer searches. It was also found that the penalty cost decreased when the probability of mutation increased, especially for the large problem. The effect of increasing mutation is to increase the amount of random search, which is beneficial with large search spaces.

The mean and standard deviations of penalty cost for every level of each factor in the experiments are summarised in Table 4. It can be seen that the schedules generated with GA have far lower penalty costs than the company schedules. All of the results generated by the GA represent a significant improvement, with the best results showing an up to 80% reduction in costs. This indicates that the GA method is far more effective than conventional scheduling.

7. Conclusions

Genetic algorithms (GA) have been developed for production scheduling problems for capital goods with many levels of product structure and multiple resource constraints. The algorithm including problem encoding, chromosome representation and initialisation, genetic operation, repair process, fitness measurement and chromosome selection is described in detail. A repair processes was developed to rectify infeasible schedules that may be produced during evolution process. It demonstrated that all chromosomes in each generation are rectifying by operation, timing and deadlock adjustment procedures.

The GA was applied to the data from collaborating company that manufactures capital goods. The GA schedules demonstrated a large reduction in tardiness and earliness costs when compared with those obtained from a company employing a traditional scheduling method. A statistical analysis on factorial experiment showed that problem size, the number of generations, the population size and the probability of mutation were statistically significant. The problem size has major effect on penalties cost. The results also suggest that the larger number of generation and population size have more chance to obtain the better schedule with longer execution time. It also suggests that using the range 0.06–0.1 for the probability of mutation produced better schedule.

References

Baker, K.R., 1974. Introduction to Sequencing and Scheduling. Wiley and Sons, New York.

Ben-Daya, M., Al-Fawzan, M., 1998. A tabu search approach for the flow shop scheduling problem. European Journal of Operational Research 109, 88–95.

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J., 1996. Scheduling Computer and Manufacturing Processes. Springer, Berlin.

Croce, F.D., Tadei, R., Volta, G., 1995. A genetic algorithms for the job shop problem. Computers Operations Research 22 (1), 15–24.

Gen, M., Cheng, R., 1997. Genetic Algorithms and Engineering Design. John Wiley and Sons, New York, USA.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Reading, MA.

Greenberg, H.H., 1968. A branch-bound solution to the general scheduling problem. Operations Research 16, 353–361.

Held, M., Karp, R.M., 1962. A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics 10, 196–210.

Jain, A.S., Meeran, S., 1999. Deterministic job shop scheduling: Part, present and future. European Journal of Operational Research 113, 390–434.

Kolonko, M., 1999. Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research 113, 123–136.

Lee, C.Y., Piramuthu, S., Tsai, Y.K., 1997. Job shop scheduling with a genetic algorithm and machine learning. International Journal of production research 35 (4), 1171–1191.

Manne, A.S., 1960. On the job shop scheduling problem. Operations Research 8, 219–223.

Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research 81, 88–104.

Nowicki, E., Smutnicki, C., 1996. A fast taboo search algorithm for the job shop problem. Management Science 42 (6), 797–813.

Ousterhout, J.K., 1994. Tcl and the Tk toolkit. Addison-Wesley, Reading, MA.

Reeja, M.K., Rajendran, C., 2000. Dispatching rules for scheduling in assembly job shops–Part I. International Journal of Production Research 38 (9), 2051–2066.

Todd, D., 1997. Multiple criteria genetic algorithms in engineering design and operation, Ph.D. thesis, Faculty of Engineering, University of Newcastle upon Tyne, UK.

Tsujimura, Y., Cheng, R., Gen, M., 1997. Improved genetic algorithms for job shop scheduling problems. Engineering Design and Automation 3 (2), 133–144.