The PEGS Scheduling System: A Case Study With Environmental Optimisation

Kieran Greer[1], John Stewart[2,*] and Barry McCollum[2]

1. The QUESTOR Centre, Queen's University Belfast, Belfast BT7 1NN, Northern Ireland.
2. The School of Computer Science, Queen's University Belfast, Belfast BT7 1NN, Northern Ireland.
Email: (k.greer, j.r.stewart, b.mccollum)@qub.ac.uk

ABSTRACT

The PEGS (Production and Environmental Generic Scheduler) program is a new generic scheduling program for manufacturing companies. It includes sort and search algorithms and can operate on the basis of elapsed time or economic values. Program performance is compared to a series of published benchmarks. The case study presented here illustrates the use of PEGS for a flow shop configuration at a local company manufacturing agricultural net-wrapping. The effects of batch size on makespan were studied using a conventional elapsed time basis. The study was then extended to consider optimising for minimum operating costs and including consideration of cost of waste arising from product changes. The program was modified to cope with operating practice in the company. Test results are presented illustrating how makespan and machine utilisation vary with job batch sizes, and the effects on an economic objective and makespan when environmental costs are included.

Introduction

In this paper we describe a new scheduling program recently developed at the Queen's University of Belfast and its application at a local manufacturing company.

The project is funded through the University's environmental research centre, The QUESTOR Centre, with the general aim of producing a practical, generic scheduler for manufacturing operations which will include an option to consider environmental costs when generating a schedule.

The new program, called PEGS, has the following main features:

- finite capacity;
- a choice of objective function (elapsed time based, due date based, economic cost based);
- initial sort heuristics (chosen to best suit the objective function);
- search algorithms (tabu, simulated annealing, beam search);
- a wide range of constraints (e.g. machine, operation, availability of services and staff);
- accounts for enviromental issues (e.g. cost of waste when using cost based objective function).

An exhaustive branch and bound search has been included for testing purposes, but is, of course, too slow to be practical for realistically sized problems.

The program may be used to generate schedules for single or parallel machines, flow shops or job shops. For the latter two cases we have implemented a version of the shifting bottleneck algorithm based on the method described by Adams, Balas & Zawack (1988). For details of tabu search see Hertz, Taillard & de Werra (1992), Morton & Pentico (1993) or Pinedo (2002), for simulated annealing see Pinedo (2002) and for beam search Pinedo (2002) or Valente & Alves (2004). The use

* Corresponding author

of search methods, in addition to the sorting heuristics, incurs much longer run times for the program. We provide some details of typical run times for a range of numbers of search iterations.

The incorporation of environmental issues is achieved by switching from time based to economic based calculations. The user must be able to provide the costs of operating each machine, rather than the time taken to process a particular operation. It is then possible to account for waste arising during product change over through the cost of that waste, entered as a single value which may be the basic material value or some higher value allowing for any processing costs already incurred. For more details see Cenk & Rajgopal's (1996) work on economic lot scheduling and Morton & Pentico's heuristic pricing formulation (1993). The program now finds a schedule which minimises the total cost rather than, say, the makespan. For the case study we use an economic version of makespan with waste costs added. The makespan objective is defined as:

$$C_{max} = max_j\{C_j\} \quad (1)$$

where *Completion time* (C_j) is the time at which activity *j* completes processing.

We modify this equation to the form:

$$EC_{max} = (C_{max} * P) + W \quad (2)$$

where EC_{max} is the economic makespan, P is the processing cost per unit time per machine, and W is the total cost of the waste arising.

Our intention is to make the program commercially available, but it is also a contribution to a larger research project and is part of the work undertaken by the Scheduling and Optimisation Research Group at Queen's (Stewart, 2006).

PEGS Architecture

The PEGS scheduling system uses a relatively straightforward architecture. Machines are grouped into workcentres, where a machine can belong to only one workcentre. The machines in each workcentre may be used to complete the same types of operations. A workcentre thus consists of a group of one or more parallel machines. Each workcentre can process a number of specified operations, and those operations may only be processed in their single specified workcentre. The system can be used to schedule for single machines, parallel machines, job shops or flow shops.

When presented with a scheduling task, the initial step is to use a dispatch heuristic to generate an initial ordering. One of three possible search strategies is then used in an attempt to find an improved ordering. The dispatch heuristics used are selected automatically and depend on the objective type and features selected. The search strategies are tabu search (Hertz *et al*, 1992, Morton and Pentico, 1993, Pinedo 2002), simulated annealing (Pinedo, 2002) and beam search (Pinedo, 2002, Valente & Alves, 2004).

For flow shop and job shop problems it is necessary to identify the workcentre which is the bottleneck – the one which is 'holding up' all the others. This is the workcentre which is currently making the largest contribution to the total objective value. We firstly generate schedules for all workcentres based on initial times. The workcentre with the largest objective is then considered to be the bottleneck. A balancing stage starts where the bottleneck is scheduled first, so its operations are given priority over the other workcentres. When the bottleneck's operations are set the other workcentres are re-scheduled taking account of the new times. Then the second bottleneck is calculated and the balancing repeated with the first bottleneck scheduled first, the second bottleneck scheduled second and then the rest. This is repeated until all of the workcentres have been ordered in terms of bottlenecks and have been balanced and thus scheduled.

The solution at each bottleneck is obtained from a slightly modified version of the scheme suggested in Morton and Pentico (1993). Firstly, a schedule is generated for all operations on the bottleneck as if they were for a single machine. In the next, modified step when calculating completion times (used to calculate the objective) we take into account the number of parallel machines in the workcentre. The operations are assigned to each parallel machine in turn to give a very general idea of possible completion times. This is not the final assignment and does not consider different machine speeds, but gives improved completion times for the single schedule ordering. The operations are then actually assigned to the parallel machines, in the order of the single schedule, but in the most economic manner. We have two algorithms here. The first simply assigns the operation to the machine that can process it first. The second algorithm tries to place similar operations on individual machines. Figure 1 illustrates this basic architecture.

As well as using a search strategy, the schedule is modified to meet a range of constraints that may be specified by the user. Time constraints include operating shift times and any arbitrary time constraint which is interpreted as a block of time when a machine is not available. Machine constraints include energy consumption and operator availability or any other user-supplied machine constraint. Each machine is assigned a value for the machine constraint and the sum of machine values for any moment in time is not allowed to exceed the constraint value. Machine constraints are applied after the schedule is generated. We do this by 'stretching' the schedule to take account of the machine constraints. Time and operation constraints are considered during the scheduling process. Two types of operation constraint have been implemented. An operation can be constrained to be completed before some other operation, but not necessarily on the same machine, or to be both before another operation and on the same machine. If this second case is specified via the constraints interface, then when assigning to parallel machines the constraint must still hold, and both operations will be assigned to the same machine. In the case study section we describe a weak form of this constraint which is used when generating the initial single machine schedules.

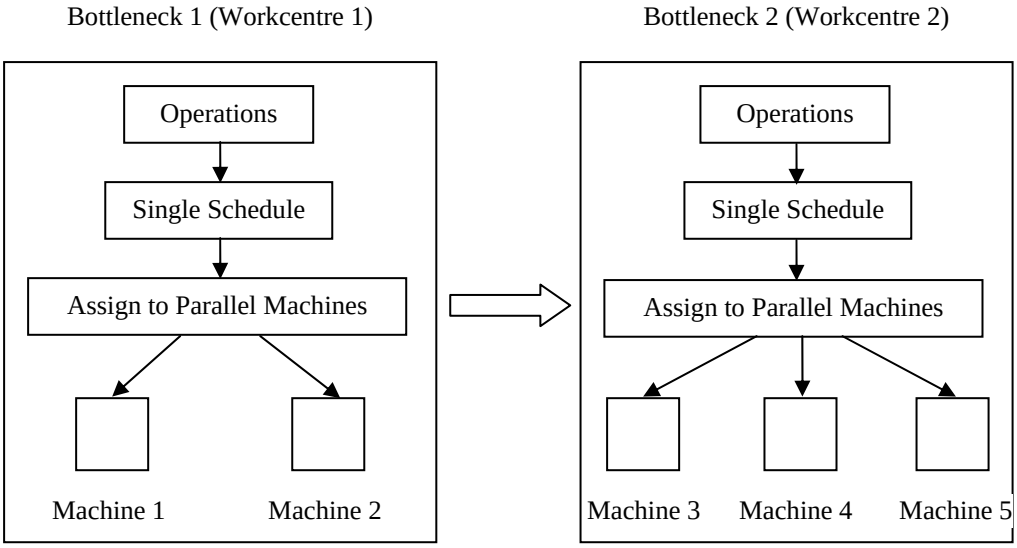

Figure 1. Basic architecture used in PEGS. Each workcentre can be a bottleneck and each bottleneck is balanced in turn. Knowing the operations to be performed on each bottleneck, we first generate a schedule as if there were only a single machine. The operations are then assigned to the parallel machines in this order. The ordering on one bottleneck may influence the ready times on another.

The system has two main windows. The user enters data through the Data Entry window and the schedule is displayed in the Schedule Window. Output comes in the form of a Gantt chart and a

textual description of the schedule. Screen shots of these two windows are given in Appendix A. The Data Entry window has a number of check boxes for features that can be switched on or off. These are listed below.

- The program allows for static or dynamic ready times. For static times, it is assumed that all jobs are ready (may be started) at the current time, or some future fixed start time. Dynamic ready times allow users to specify when individual jobs may be started.

- The program also takes account of machine idle periods. The default is to allow idleness, but there is an option to minimise this, where the schedule with the smallest amount of idleness will be preferred.

- Setup times may be entered for each operation or for families of operations.

- It is possible to specify a preference that operations be completed on the same shift in which they started.

- The amount of wasted product may be entered for each operation or for families of operations.

- Energy tariffs, varying with time of day and calendar, may be entered and included when calculating an objective based on economic values rather than time values.

- We permit operations for the same product to overlap rather than to take place in strict sequence. The default is strict sequential, assuming that each operation requires as input the output from the previous one. However, it will sometimes be the case that operations are independent of each other and can overlap in time on the same group of parallel machines or different groups of machines. Overlapping can be switched on or off for different operations required for a particular job.

We have provided for buffers between each operation which may accumulate stock as it is produced. Users may specify minimum quantities which may be removed from a buffer (see the case study for an illustration of the use of buffers).

We have also implemented a special 'accumulation' operation. Users may group a number of operations from different jobs into a single operation on a notional machine and process them as a single operation. This feature would be useful in a pottery or bakery where different items would be processed together in a single oven. It was actually implemented for a furniture manufacturer which makes various components at one site and then transports them to another for subsequent assembly. The transport step is treated as an accumulation operation.

Program performance

PEGS can handle parallel groups of machines in a job shop or flow shop environment. The program was tested by comparing results with benchmark tests published by Wittrock (1988) and subsequently by Cheng *et al.* (2001) and Phadnis *et al.* (2003) using the same test data. The environment is a flow shop with groups of parallel machines where there are three different groups. Transport time is added to move a magazine of cards from one bank of machines to another. We included this time by adding it as an equivalent processing event beteween the two operations involved. Each author improved on the previous author's best objective values. They wrote algorithms to solve this type of problem with a single objective in mind. The results below are for the generic PEGS system which has not been hand crafted to solve just one problem.

Problem	Current Best	PEGS	% Worse	Lower Bound
Jobs1	760	791	4.1	720
Jobs2	770	818	6.2	715
Jobs3	770	821	6.6	694
Jobs4	785	817	4.0	694
Jobs5	961	1005	4.6	895
Jobs6	667	693	3.9	584

Table 1 Objective values for the 6 scheduling problems of Wittrock (1988). The current best results of Phadnis *et al.* (2003) are shown followed by the PEGS results. The % worse is the amount PEGS is worse than the current best. The final column is the lower bound.

PEGS is unable to find solutions as good as those of Phadnis, but the results seem reasonable for a program which has not been optimised for this particular benchmark. A summary of the performance of PEGS for six other benchmark test sets is shown in Table 2. Our aim is to produce a program which can provide good schedules across a wide range of bench mark data sets, thus increasing the range of applicability. We believe the results presented here are very promising in this respect and represent a first step in achieving our overall goal.

Benchmark sources	Objective	Shop type	Test identity	Average % worse than current best
Adams, Balas & Zawack (1988)	Makespan	Job shop	abz5-9	14.3
Fisher and Thompson (1963)	Makespan	Job shop	mt10	15.5
Lawrence (1984)	Makespan	Job shop	la19-21,la24-25	16.1
Applegate and Cook (1991)	Makespan	Job shop	orb1-orb5	19.1
Purdue (in Dimirkol *et al*, 1998)	Makespan	Flow shop	flcmax_20_15_3,4,6	1.0
Purdue (in Dimirkol *et al*, 1998)	Lateness	Flow shop	fl_20_15_1_1_2, 3, 7, 8, 9, fl_20_15_2_2_5, 6	5.9

Table 2 Comparison of PEGS performance against most recently published optimum objective for a series of benchmarks.

PEGS has been written so that it is possible to distribute the calculations for the scheduling algorithm over a number of networked computers. Tests confirmed that run times for the networked version are close to proportional, but with a small adverse impact on the objective value. Using two computers for a five bottleneck problem produced an objective that was 1% worse than when using one computer but executed in just over half the time. Using three computers produced an objective that was also just over 1% worse but took just over one third of the time.

Case Study

Process description

Our case study arises from a collaboration with a local company making plastic netting for the farming industry. Figure 2 illustrates the key features of their manufacturing process. The first stage is to produce rolls of blown film. Melted polymer is extruded at high speed through a circular orifice to form a continuous cylinder. The cylinder is collapsed to form a flat double sheet which is wound onto large rolls. The second stage takes the rolls of blown film, slits them into ‘threads’ and knits the threads together to form the netting. Two different grades of blown film, known as pillar and inlay, are needed to make the netting.

The company has twelve netting machines and three blown film machines. The netting machines have different widths and, depending on the width of product required, typically produce four or five rolls of netting at a time.

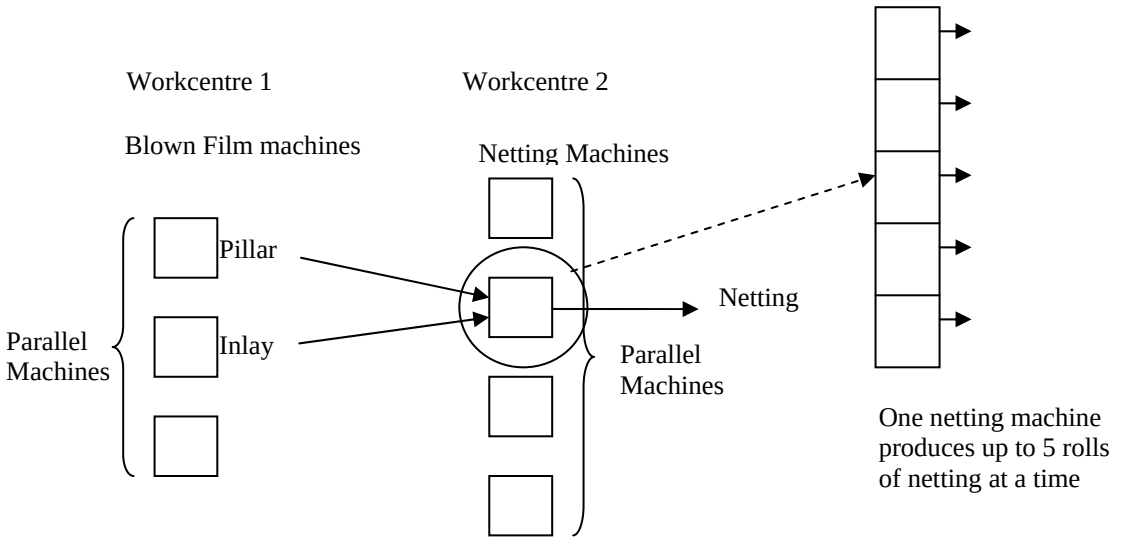

Figure 2. Key features of the manufacturing process showing two workcentres for blown film and netting machines.

The company has a large range of products with over 200 different types. However, many of the products are made from the same blown film grades, so there are fewer of these. Blown film is produced in large rolls which are held in store until required on a netting machine. Netting production is planned to exactly match customer order quantities and this usually means that, at the end of a job on the netting machines, there will be part rolls of pillar and inlay remaining un-used.

When planning their production, company staff calculate how many rolls of each type of blown film are needed and schedule these on the three machines. As the blown film becomes available, the netting machines are set up and production begins. Set up time for the netting machines depends on the product type before and after change over. Because several products may be made from the same type of blown film, when it is convenient, set up time is reduced by following on with a product which uses the part rolls of film already feeding a netting machine.

The company operates a seven-day continuous shift system and wish to maximise the output from the netting machines. Utilisation of netting machines is less than that of the blown film machines – suggesting that the bottleneck to increased production is the manufacture of blown film. However, it is not known whether the sequence of production of either blown film or netting is optimum.

To avoid build up of partly used rolls of blown film, the part rolls are used in preference to new full rolls when a new job is started.

The scheduling problem

For scheduling purposes, we can consider the factory to have two workcentres, one with three blown film machines and the other with 12 netting machines. The sizes, speeds and capacities of the various machines are different, but nearly all products can be processed on any of the parallel machines within a workcentre.

Newly produced rolls of blown film are stored until needed by the netting machines. The store also contains partly-used rolls which have been taken off netting machines at the end of a run. We have

implemented this store as a buffer between the two workcentres. The buffer stock also includes partly-used rolls left on a particular netting machine at the end of a run and which will be used for another product to be made on that machine. We do not differentiate between part rolls in the store or still on a netting machine. However, the required set up time will be less when a product change does not require removal of part rolls and replacement with another grade.

When determining how much blown film must be manufactured for a particular job, any un-allocated stock of film in the buffer is taken into account first. A typical job might require many rolls of blown film to complete, but the netting machines can start as soon as one roll of each of the two necessary types is available in the buffer. Blown film can be produced more quickly than the netting machines can use it, so that it is safe to start a netting machine as soon as one full roll of each type is available in the buffer.

The two grades of blown film, pillar and inlay, needed for any particular product can be produced independently on the blown film machines. But, since production of netting cannot start until both grades are available, they need to be scheduled at about the same time. In order to achieve this we have implemented a weak version of a 'beside' constraint. When the user is entering initial data about which operations may be carried out on which machines, it is possible to specify that one operation should be scheduled close to another.

As part of the initial testing programme, some schedules were created using historical data for orders. These test schedules had frequent changes of product type on the blown film machines – which were necessary to get as many netting machines as possible operating as early as possible.

To illustrate this point consider three jobs to be processed, each requiring different grades of pillar and inlay. A single blown film machine schedule might be j1P, j1I, j2P, j2I, j3P, j3I. As soon as the pillar and inlay are ready for job 1, a netting machine can be started. So repeated switching from pillar to inlay production means that netting production can begin earlier.

However, this revealed a further constraint to be taken into account. Changing product type on the blown film machines, from pillar to inlay and back, is always associated with waste of a significant amount of the polymer feedstock. To avoid this waste, blown film production is scheduled to minimise changes in product type. This is an example of an environmental and an economic constraint. In an initial attempt to meet this constraint we added a method to PEGS which has a strong bias towards assigning operations of the same type to be produced in sequence on the same machine.

If there was only one blown film machine, the new sequence might be j1P, j2P, j3P, j1I, j2I, j3I. Now we see that the first netting machine cannot start until the fourth blown film item, j1I, is produced. If we have two blown film machines, the production sequence could be j1P, j2P, j3P on one of them and j1I, j2I, j3I on the other. This is a superior ordering in terms of earlier starting for netting machines. During program set up, it is necessary to enter the various operations needed to make a product in the sequence in which they should be completed. In this case pillar and inlay must be produced before netting, but there is no ordering between pillar and inlay. We allow users to specify that some operations may overlap – they may be produced at the same time on different machines.

One final feature requested by the company was to be able to fix certain jobs on machines. It may be the case that an operation would generally be run on a specified set of machines (any one in the appropriate workcentre), but occasionally it may be necessary to override the general case and use a specific machine. As well as fixing the machine for the operation it may also be useful to fix the time when the operation will take place. When specifying the operations for each product through the products window, it is necessary to indicate the preferred machines the operations should be run on. This will be the general case. However, it is possible to override this selection by fixing the operations at startup or manually moving the operations to a different machine using the Gantt chart display. This will then override the general case.

Program Tests

Typically, some customer orders are for large numbers of rolls of netting which might require many weeks of processing on a single netting machine. Assuming that this might result in an inefficient schedule, with low average machine utilisation, tests were carried out to determine the impact of job sizes on utilisation (and the makespan and flowtime objectives). The test was carried out using customer orders expected to be processed over a period of five to six weeks. Larger jobs were split into 400, 200, 150, 100 and 50 roll lots to see which might be preferable. The impact on the number of jobs is shown in Table 3. Note that each job involves three operations: make pillar on blown film machine; make inlay on blown film machine; and make netting on netting machine. Makespan and flowtime objectives were tested for the three search strategies. Makespan produced slightly better completion times than flowtime, but values were very close. The nearest neighbour tabu and simulated annealing search strategies were also very similar.

The optimum batch size is clearly around 100 to 150 rolls where the makespan is 33 to 34 days. This compares with 40 to 41 days for the larger batch sizes, an improvement of around 17 %. Netting machine utilisation figures follow a similar pattern with highest utilisation for batch sizes around 100 to 150. The smallest batch sizes of 50 rolls reduces the average run time per operation and set up times become an important disadvantage. Makespan values increase from the optimum to 36 to 37 days. Interestingly, the number of iterations used for the search had little impact on the best objective value found. However the results presented are an improvement on the ordering achieved by the initial sort heuristic, demonstrating that the search step is improving the objective values.

No. of Jobs	**Maximum batch size**	**Number of iterations for each bottleneck run**							
		1,000		**10,000**		**100,000**		**1,000,000**	
		Make-span	**% Util**	**Make-span**	**% Util**	**Make-span**	**% Util**	**Make-span**	**% Util**
21	400	40.5	69	41.5	65.5	41.8	67.5	40.8	68.5
41	200	34.3	72.5	35.1	71	35.5	72	36	71.5
55	150	34.0	73.5	33.5	72.5	33.9	72	33.5	72.5
82	100	34.2	73	34.8	72.5	34.2	71	33.3	69.5
164	50	36.5	65	36.4	65.5	36.8	65	37.1	64

Table 3 Objective values (days) for makespan and netting machine utilisation for varying batch sizes.

Run times for the tests scaled approximately linearly with respect to number of iterations and also number of jobs. The Beam search scaled linearly with respect to the beam width but was more like exponential with respect to the number of jobs. Table 4 shows the relative run times for the tabu search with the makespan objective for test runs of 1,000, 10,000, 100,000 and 1,000,000 iterations per bottleneck.

In this case, with an optimum batch size of 100 to 150 rolls, program run time with 100,000 iterations is about 5 minutes, and with 1,000,000 iterations about 50 minutes.

The company had pointed out that it was important to minimise the number of product changes on the blown film machines as each change is associated with wasting some of the blown film polymer. To investigate this issue, we used the program in economic mode, assigning costs to operate the machines and costs of wastes. The cost data are confidential, so the results presented here are not the actual costs. They have been chosen to illustrate the operation of the scheduler. The values used were £50

per hour to operate each of the machines, £75 cost of waste for product switching between pillar and pillar or inlay and inlay, and £100 cost of waste for switching between pillar and inlay.

No. of Jobs	Maximum batch size	Number of iterations for each bottleneck run			
		1,000	10,000	100,000	1,000,000
21	400	1	10	92	918
41	200	2	18	169	1,685
55	150	3	23	223	2,220
82	100	4	34	324	3,233
164	50	10	67	632	6,298

Table 4 Program run time in seconds for the batch size tests.

No. of jobs	Group similar operations			No grouping, minimises makespan		
	Objective	Makespan	% Util.	Objective	Makespan	% Util.
21	52022	42.8	61.4	52619	41.1	67.6
41	44795	36.7	66.4	48668	35.2	71.7
55	43030	35.1	67.5	48818	33.7	72.6
82	43046	35.1	67.1	52901	34.1	71.5
164	42682	34.8	67.1	65642	36.7	64.9

Table 5. Economic objective values (£), makespan (days) and % utilisation of netting machines comparing simulations with and without enforced grouping of similar operations.

The results are shown in Table 5 where we compare the situation when similar operations are grouped together, which acts to minimise the number of product changes on the blown film machines, with the situation where no grouping is enforced and the outcome should be equivalent to optimising for minimum makespan.

When grouping is enforced, the economic objective value decreases as batch size decreases (and number of jobs increases). Makespan follows a similar pattern with the optimum 34.8 days, compared to the worst case of 42.8 days (an improvement of 19 %). So, even though there are potentially more product changes, because of the enforced grouping minimising the waste, the smallest makespan corresponds with the smallest objective value. Note that the grouping is enforced in the blown film machines only, where the waste is produced. For the netting machines where waste is not considered, the algorithm that does not group operations is always preferred, as it produces a lower makespan.

The comparison with the no grouping case is interesting. For 21 jobs, the objective value is slightly worse and the makespan slightly better. However, as expected, the changing values of the objective and makespan follow the pattern shown in Table 2 – because we are essentially optimising for makespan. The best makespan values are 33 to 34 days, for the 100 or 150 roll batch sizes. However, the economic objective averages £7,800 worse (15%) for these two cases. In strictly economic terms (for the assumed costs) it would be better to use the smaller, 50-roll batch sizes.

The impact of waste costs is illustrated in Table 6 which isolates the costs associated with wastes generated from product changes on the blown film machines. We see from the third column that the economic costs of minimising makespan rises rapidly as the batch size decreases. The schedule will then include many product changes with larger associated waste costs.

No. of jobs	Group similar operations	No grouping, minimises makespan
21	659	3260
41	769	6391
55	725	8334
82	841	11875
164	813	21622

Table 6. Waste cost element of economic objective values (£) comparing simulations with and without enforced grouping of similar operations.

Conclusions

The PEGS scheduler was designed and implemented as a generic program, capable of producing good schedules for a wide range of manufacturing operations. It uses both heuristic sorting algorithms and a choice of search strategies from tabu, simulated annealing and beam search. As part of the initial testing process, we have undertaken case studies with local companies.

In order to satisfy the problems of the study reported here, we had to introduce a number of extra features to the system. We introduced a feature where one operation in a sequence could start not just after the previous operation had finished, but after a user-specified number of items had been produced from the previous operation. We also allow the start time for any particular operation to be constrained to follow the completion of more than just one preceding operation. These preceding operations would be retrieved from a buffer. So, in an assembly operation, the assembly could only take place when all the component parts had been completed (or provided from an external source).

We allow a constraint which forces one operation to follow consecutively after another. This is a strong constraint that also applies when the operations are assigned to parallel machines. We introduced a weak version of this constraint, where the operations are located together in the initial schedule ordering, but could then be separated when moved to parallel machines. The aim of this weak version was to favour the situation where two operations are completed together to allow a third operation that depends on both of them to start. We also allow operations for the same job to overlap on different machines when one operation does not depend on the other. Selection of these features is under the control of the system user.

One of our aims for the scheduler was to allow the user to include environmental factors in the optimisation. We have implemented this by allowing the optimisations to be carried out on an economic rather than an elapsed time basis. As long as appropriate economic values can be assigned to the environmental issues (for example waste arisings or electricity consumption), and to all the operating costs of the factory, then it is possible to generate schedules optimised for minimum costs, which will therefore include consideration of the environment. For waste costs associated with product changes on the machines, we have added a modified algorithm for allocating operations to parallel machines. The new method has a strong bias towards placing similar operations together (in sequence) on each machine. The effect of this is to greatly reduce the waste generated during product changes providing the cost of this waste is significant.

Future Work

We plan to add more environmental and economic features to the system. The scheduling in PEGS is controlled by a central algorithm. We are developing a distributed version which will use intelligent agents.

Acknowledgement

This work was funded through the University's environmental research centre, the QUESTOR Centre, from a grant provided by the EU RTD Centres of Excellence (PEACE II) initiative.

Appendix – Screen Shots of the Program

There are two main windows, which the user can toggle between. The first screen shot shows the main data entry window (Figure A1). Most of the menu items on the upper menu bar are associated with subsidiary data entry windows. Machine and job details are displayed on the grid shown.

When a schedule has been generated it is displayed in the schedule window (Figure A2). The user may modify the schedule by dragging items around on the Gantt chart. The resulting schedule may be displayed in text form and printed in various layouts.

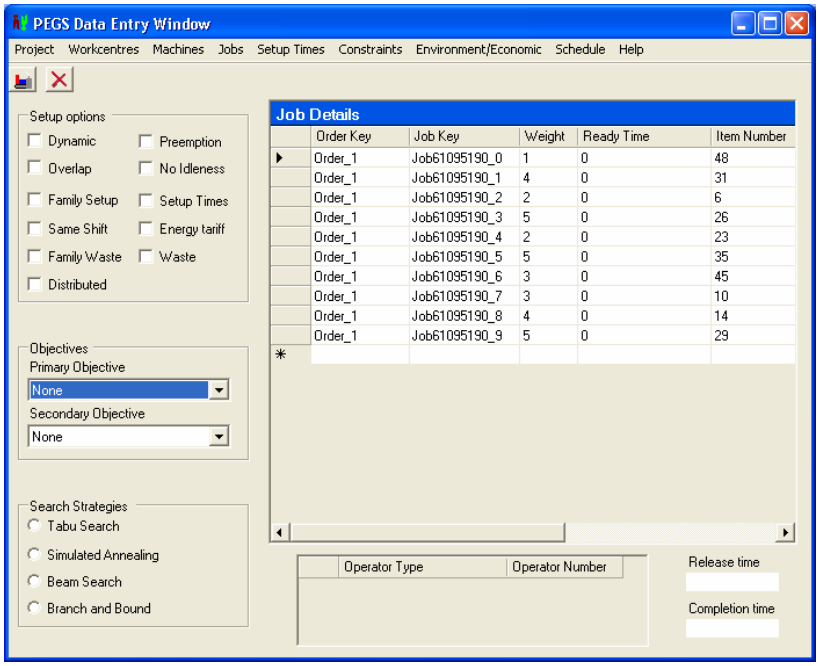

Figure A1. Screen shot of the main Data Entry window. Schedule options are selected on the left, while the grid on the right shows current job details.

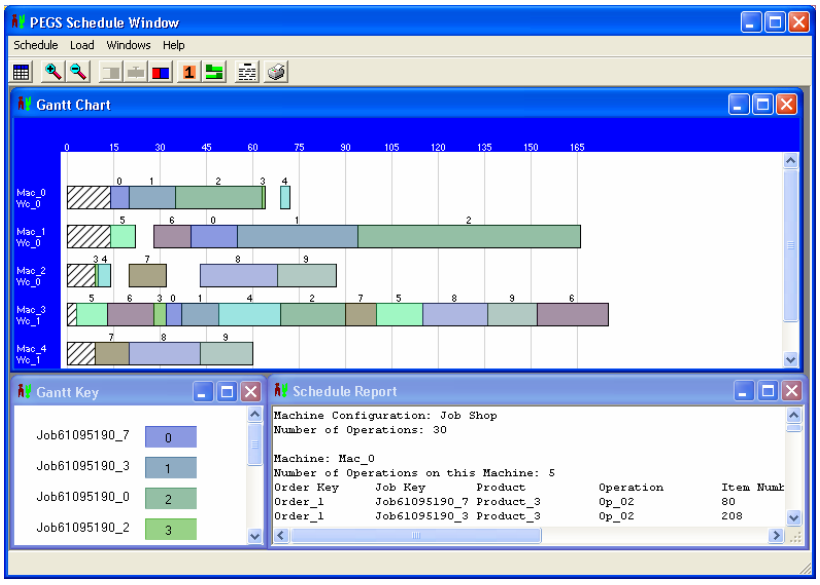

Figure A2. Screen shot of the Schedule window showing part of the Gantt chart and a text listing of the schedule details.

References

Adams J., E. Balas and D. Zawack, (1988). *The Shifting Bottleneck Procedure for Job Shop Scheduling*, Management Science, Vol. 34, No. 3, pp. 391-401.

Applegate D. and Cook W., (1991). *A computational study of the job-shop scheduling instance*, ORSA Journal on Computing Vol. 3, No. 2, pp. 149-156.

Cenk T. and J. Rajgopal (1996), *An Evolutionary Computation Approach to the Economic Lot Scheduling Problem*, Technical Report No. 96-1, University of Pittsburgh, Department of Industrial Engineering, Pittsburgh, PA 15261.

Cheng J., Y. Karuno and H. Kise, (2001). A Shifting Bottleneck Approach for a Parallel-Machine Flowshop Scheduling Problem, Journal of the Operations Research, Society of Japan, Vol. 44, No. 2, pp. 140-156.

Demirkol E., Mehta S. and Uzsoy R. (1998). Benchmarks for shop scheduling problems, European Journal of Operational Research, Vol. 109, pp. 137-141.

Duarte R., Rego C. and Gamboa D., (2006). *A Filter and Fan Approach to the Job Shop Scheduling Problem: A preliminary Study*, paper submitted for publication.

Fisher H. and Thompson G.L. (1963). *Probabilistic learning combinations of local job-shop scheduling rules*, J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, pp. 225-251.

Hertz A., E. Taillard and D. de Werra, (1992) *A Tutorial on Tabu Search*, http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf.

Lawrence S., (1984). *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Morton, T. E. and D. W. Pentico (1993). *Heuristic Scheduling Systems*, Wiley Series in Engineering and Technology Management.

Phadnis S., J. Brevick and S. Irani, (2003). Development of a new heuristic for scheduling flow-shops with parallel machines by prioritizing bottleneck stages, Transactions of the Society for Design and Process Science, Vol. 7, No. 1, pp. 87 – 97.

Pinedo, M. (2002). *Scheduling: Theory, Algorithms and Systems*, Prentice Hall Inc.

Stewart J.R. (2006) Website of the Scheduling and Optimisation Research Group, Queen's University, Belfast. Retrieved 20th January 2006 from http://www.qub.ac.uk/cs/SORG.htm

Valente J. M.S. and R. A. F. S. Alves, (2004). *Beam search algorithms for the early/tardy scheduling problem with release dates*, Faculty of Economics, University of Porto, Working paper, April 2004.

Wittrock R. J., (1988), An Adaptable Scheduling Algorithm for Flexible Flow Lines, Operations Research, Vol. 36, No. 3, pp. 445-453.