Dear Dr/Prof. JohnRea Stewart,

Here are the proofs of your article.

- You can submit your corrections **online** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- For **fax** submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Please return your proof together with the **permission to publish** confirmation.
- Remember to note the journal title, article number, and your name when sending your response via e-mail, fax or regular mail.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.

Please note

Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible**.

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].

If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: www.springerlink.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us, if you would like to have these documents returned.

The **printed version** will follow in a forthcoming issue.

Fax to: +1 347 649 2158 (US) or +44 207 806 8278 (UK) or +91 44 4208 9499 (INDIA)

To: **Springer Correction Team**
6&7, 5th Street, Radhakrishnan Salai, Chennai, Tamil Nadu, India – 600004

Re: Journal of Intelligent Manufacturing DOI:10.1007/s10845-007-0068-y
Comparison of a centralised and distributed approach for a generic scheduling system

Authors: Kieran Greer · JohnRea Stewart · Barry McCollum

Permission to publish

I have checked the proofs of my article and

☐ I have no corrections. The article is ready to be published without changes.

☐ I have a few corrections. I am enclosing the following pages:

☐ I have made many corrections. Enclosed is the complete article.

Date / signature ____

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	Comparison of a centralised and distributed approach for a generic scheduling system	
Article Sub-Title		
Article CopyRight - Year	Springer Science+Business Media, LLC 2007 (This will be the copyright line in the final PDF)	
Journal Name	Journal of Intelligent Manufacturing	
Corresponding Author	Family Name	**Stewart**
	Particle	
	Given Name	**John Rea**
	Suffix	
	Division	The School of Computer Science
	Organization	Queen’s University Belfast
	Address	BT7 1NN, Belfast, Northern Ireland
	Division	School of Electronics, Electrical Engineering and Computer Science
	Organization	Queen’s University Belfast
	Address	BT7 1NN, Belfast, Northern Ireland
	Email	j.r.stewart@qub.ac.uk
Author	Family Name	**Greer**
	Particle	
	Given Name	**Kieran**
	Suffix	
	Division	The QUESTOR Centre
	Organization	Queen’s University Belfast
	Address	BT7 1NN, Belfast, Northern Ireland
	Division	School of Computing and Mathematics
	Organization	University of Ulster
	Address	Newtownabbey, Northern Ireland
	Email	krc.greer@ulster.ac.uk
Author	Family Name	**McCollum**
	Particle	
	Given Name	**Barry**
	Suffix	
	Division	The School of Computer Science
	Organization	Queen’s University Belfast
	Address	BT7 1NN, Belfast, Northern Ireland
	Email	b.mccollum@qub.ac.uk
Schedule	Received	9 December 2007
	Revised	
	Accepted	11 December 2007
Abstract	PEGS (Production and Environmental Generic Scheduler) is a generic production scheduler that produces good schedules over a wide range of problems. It is centralised, using search strategies with the Shifting Bottleneck algorithm. We have also developed an alternative distributed approach using software agents. In some cases this reduces run times by a factor of 10 or more. In most cases, the agent-based program also	

	produces good solutions for published benchmark data, and the short run times make our program useful for a large range of problems. Test results show that the agents can produce schedules comparable to the best found so far for some benchmark datasets and actually better schedules than PEGS on our own random datasets. The flexibility that agents can provide for today’s dynamic scheduling is also appealing. We suggest that in this sort of generic or commercial system, the agent-based approach is a good alternative.
Keywords (separated by '-')	Scheduling - Software agents - Distributed scheduling - Production scheduling - Environmental constraints
Footnote Information	

J Intell Manuf
DOI 10.1007/s10845-007-0068-y

Comparison of a centralised and distributed approach for a generic scheduling system

Kieran Greer · John Rea Stewart · Barry McCollum

Received: 9 December 2007 / Accepted: 11 December 2007

Abstract PEGS (Production and Environmental Generic Scheduler) is a generic production scheduler that produces good schedules over a wide range of problems. It is centralised, using search strategies with the Shifting Bottleneck algorithm. We have also developed an alternative distributed approach using software agents. In some cases this reduces run times by a factor of 10 or more. In most cases, the agent-based program also produces good solutions for published benchmark data, and the short run times make our program useful for a large range of problems. Test results show that the agents can produce schedules comparable to the best found so far for some benchmark datasets and actually better schedules than PEGS on our own random datasets. The flexibility that agents can provide for today's dynamic scheduling is also appealing. We suggest that in this sort of generic or commercial system, the agent-based approach is a good alternative.

K. Greer
The QUESTOR Centre, Queen's University Belfast,
Belfast BT7 1NN, Northern Ireland
e-mail: krc.greer@ulster.ac.uk

Present Address:
K. Greer
School of Computing and Mathematics, University of Ulster,
Newtownabbey, Northern Ireland

J. R. Stewart · B. McCollum
The School of Computer Science,
Queen's University Belfast, Belfast BT7 1NN, Northern Ireland
e-mail: b.mccollum@qub.ac.uk

J. R. Stewart (✉)
School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN,
Northern Ireland
e-mail: j.r.stewart@qub.ac.uk

Keywords Scheduling · Software agents · Distributed scheduling · Production scheduling · Environmental constraints

Introduction

PEGS (Production and Environmental Generic Scheduler) is a generic production scheduling system developed at the Queen's University of Belfast. In addition to conventional time-based optimisation, PEGS calculations may be based on economic values and this allows consideration of any environmental effects to which a cost can be assigned. For example, waste generated at product change-overs or the varying costs of electricity at different times may be included. PEGS allows the use of multiple objectives, setup times and a wide range of constraints. These features mean that the program may be used to generate schedules for many different manufacturing models including single machine, parallel machines, flow shops and job shops. A full description of the program and comparisons of its performance on a variety of benchmark tests are given in Greer et al. (2006).

In this paper, we describe some recent work to develop a distributed algorithm, using software agents, and compare the new version of the program with the original centralised algorithm using some randomised datasets and published benchmarks. For convenience here we will refer to the original version as PEGS and the new version as PEGSAgent.

In PEGS we use search heuristics together with the Shifting Bottleneck algorithm (Adams et al. 1988). In this approach all machines are considered together and a schedule is calculated from a single centralised algorithm. This approach can produce good quality solutions, but today's systems also need to be very flexible. A flexible system may cope with situations where machines are distributed across different sites, with machine breakdowns, or with agile manufacturing methods (He and Babayan 2002). A distributed approach may be preferred because of its greater flexibility. It appears, from the lack of published comparisons, that it may be difficult to achieve the same quality of solution (schedule) using distributed methods. For PEGSAgent we have implemented and tested a distributed approach using agent-based technology. In the remainder of this paper we present some comparisons between the two different approaches. We have not found published details of similar comparisons elsewhere, either considering a generic or a specialised system.

Software agents have now been widely used in scheduling, see Shen (2002) for a summary. An agent-based approach is a distributed approach, where complex problems are broken down into a number of simpler problems which are distributed among several agents. The agents then attempt to solve the simpler problems and their combined efforts will produce a solution to the global problem. Jennings (2000) outlines the advantages of using an agent-based approach to software engineering. There are various features that an agent should possess. One of these is that an agent should be autonomous or semi-autonomous. This means that it has its own internal control system and is able to make independent decisions on behalf of the user. An autonomous agent is also pro-active and may initiate other processes. Agents communicate with each other using an agent communication language. The Foundation for Intelligent Physical Agents, FIPA, is the organisation responsible for creating standards for agent-based technology and its Agent Communication Language is called ACL. Agents use ACL to pass messages to other agents telling them what to do. The message passing protocols are also standardised. Our agents are semi-autonomous in that they generate a schedule independently once they have been asked to do so. They communicate using the standard communication protocols Contract Net and Query Ref (FIPA 2002), plus an even simpler protocol that just asks an agent to do something.

The Contract Net protocol is a standard bidding protocol. It is used to allow agents to bid with each other to provide a service for some other agent. In our case we allow the machine agents to compete with each other to provide a time slot in which to complete an operation belonging to some job. Query Ref is much simpler and is used to send a query from one agent to another and then receive a reply from that agent. We have written our own agent platform, agents and protocols, etc. so that they could be customised to our own specific problem. However, we expect that the protocols used here could equally have been implemented using an open-source generic agent platform.

In the remainder of this paper we describe some related work on centralised and distributed systems (Section "Related work"), the main features of our system (Section "Main features"), the architecture of our centralised (Section "Centralised architecture") and distributed systems (Section "Distributed architecture"), a comparison with other systems (Section "Comparison with other systems"), a comparison of the two program versions on a series of tests (Section "Testing the two approaches"), and some conclusions (Section "Conclusion").

Related work

Search algorithms

PEGS provides a choice from three different search algorithms to find a solution: tabu search (Hertz et al. 1992; Morton and Pentico 1993; Pinedo 2002; Nowicki and Smutnicki 1996), simulated annealing (Pinedo 2002) and beam search (Pinedo 2002; Valente and Alves 2004).

Tabu search and simulated annealing are both nearest neighbour search strategies. They begin with some initial ordering (a schedule) and then make a change to it in the close neighbourhood of that ordering. If the changed ordering is better it is retained as the current best solution and a new modification is made from that one. If the change is not an improvement, then a different modification is made from the original ordering. The program continues making modifications and evaluating the results until a specified number of iterations have elapsed or time has run out. If the only next moves are those which give a worse solution then it is possible to be trapped in a local optimum (a region of the search space where no immediate near neighbour move will provide a better solution, even though there may be better solutions in some other area of the search space). In this case it may be necessary to allow a move to a worse solution in order to move out of a local optimum, so that an even better global optimum may be found. Tabu search and simulated annealing have mechanisms in place to allow this.

In tabu search a record is kept of the last m moves made in a tabu list. Making a reverse of these moves while they are in the list is not allowed. This will prevent repeated cycling between two moves, but may mean it is occasionally necessary to make a worse move (that is not in the list) before it becomes possible to again find better moves. The actual moves made are generated randomly, typically by choosing two operations at random and swapping their positions to produce a new candidate solution. For the first half of the search we swap any two operations, while for the second half we swap just neighbouring operations.

Simulated annealing provides a probability function, where a worse move may be made at certain iterations with a pre-determined probability. This also means that if the solution is trapped in a local optimum there is an opportunity of moving out of it again. Simulated annealing has a 'cooling factor' which is used to determine how often the probability function will allow a worse move. Initially the cooling factor is set quite high and worse moves are permitted more often. As the number of iterations completed increases, the cooling factor is decreased, making it less likely that a move to a worse solution will be allowed. This has the effect of allowing the search strategy to roam widely at the start and then settle on a particular area of the search space as the number of iterations increases. The basic algorithms for these search strategies are described in Pinedo (2002). The tabu search algorithm has been modified to include an aspiration criterion (see Chambers and Barnes 1996). If a move is made that results in a better solution than any found so far, even if it is tabu, the move is still allowed.

Beam search is a breadth-first search that prunes nodes at each level of a search tree. From each solution at one level, all possible next solutions are generated by swapping elements (operations) with one of the elements in the original solution. This produces a new number of solutions at a new level of the search tree. These new solutions can then be expanded in the same way, until all swaps have been considered. The basic algorithm is again taken from Pinedo (2002). Starting from an initial ordering, changes are made by swapping all elements with the first one. This will produce a number of new solutions. Each new solution is evaluated and a certain number, as specified by the beam width, are retained. If, for example, the beam width is 10, only the 10 best new solutions are kept. These new solutions are then expanded again by swapping all elements (except for the first one) with the second element. These new solutions are evaluated again and a beam width number kept, and so on until all of the elements in the solution have been swapped. The algorithm will generate all of the possible solutions, except for those pruned (or rejected) by the beam width. It is also possible to include a filter width, where an initial evaluation of new solutions is made, based on a relatively simple evaluation function, and a filter width number of these are retained. This filter width number is subsequently evaluated using a more sophisticated function and a beam width number of these are retained.

There is also an exhaustive branch and bound algorithm for simple testing that considers all possible solutions. This is, of course, far too slow to use on problems of a realistic size.

The Shifting Bottleneck algorithm

The Shifting Bottleneck algorithm is used for job shop or flow shop configurations. This is a well tried and trusted algorithm that was originally suggested in Adams et al. (1988) and is also in Balas and Vazacopoulos (1998). It is known to be reasonably fast and to produce good quality solutions. Various authors have produced optimal solutions for benchmark datasets using specialised algorithms coupled with the Shifting Bottleneck algorithm. Our system is generic using the search strategies described in Section "Search algorithms" with the Shifting Bottleneck algorithm and thereby producing reasonable results for a wide range of problems.

The Shifting Bottleneck algorithm works by finding the bottlenecks in the system. These are the machines or workcentres responsible for the greatest delay to the schedule. Preference is given to these areas by generating their schedules first. This approach tends to generate schedules most favourable to the bottleneck operations/machines. In the Shifting Bottleneck algorithm, during each iteration, it is necessary to find the worst bottleneck. This is the workcentre that is producing the worst evaluation. A workcentre is a group of one or more machines, all running in parallel, that can process the same operations. A schedule is generated for each workcentre using initial available times. The workcentre with the worst objective is taken as the worst bottleneck. This bottleneck is then optimised first in the next iteration, which means that its allocation of operations to time slots are given preference over all other bottleneck allocations. When the operations have been allocated, the other bottlenecks' allocations are adjusted based on these and optimised in turn. Next the second worst bottleneck is identified as the worst of the remaining bottlenecks. The first and second worst bottlenecks are then balanced together. This means optimising the first worst bottleneck, then the second worst bottleneck (updating allocation times in the process), and repeating until there are no changes in their evaluations or a specified number of iterations have passed. When these bottlenecks have been balanced, the allocation times in the remaining bottlenecks are updated and evaluated in turn to determine the third worst bottleneck. The three worst bottlenecks are then balanced and the process is repeated until all bottlenecks have been balanced.

Together, the search and shifting bottleneck algorithms represent a centralised approach to the scheduling problem. In the next sections we describe an alternative approach using agent technology.

Agent-based systems

There has been a move towards a distributed solution to the production scheduling problem because it is more flexible and adaptable to changes that may occur in day-to-day scheduling. There is now a substantial body of literature on agent-based systems used as distributed production scheduling systems. Babiceanu and Chen (2006) survey the current status of distributed manufacturing systems, including agent-based technology in production scheduling. One advantage of this approach is its ability to cope with dynamic changes such as machine breakdowns, new orders or changes in existing orders; a flexibility sometimes described as agile manufacturing (He and Babayan 2002; Boccalatte et al. 2004). If, for example, each machine is represented by a separate agent, it is easier to simply remove a machine if it breaks down, or keep parts of the schedule the same while allowing other parts to be changed by fixing certain agents and allowing others to be re-scheduled. If it is only necessary to re-schedule part of the system, it can be done much more quickly with an agent-based approach. Shen and Norrie (1998) describe their MetaMorph II system which attempts to address these issues. Yoo and Muller (2002) also address the dynamic job shop scheduling problem. Shen (2002) provides a tutorial which describes the problems that agents address and also lists some systems. Walker et al. (2005) describe an agent-based system that uses evolutionary algorithms to evolve a scheduler rather than the schedule itself. Globally dispersed manufacturing enterprises need to address issues of accessibility, integration and re-configurability (Yen 1998). Planning is usually integrated with the scheduling and changing product specifications or shop floor configurations need to be quickly integrated into the system. To tackle this problem, the planners and schedulers may be involved in some sort of communication. An agent-based approach can prove a good match to this kind of problem as it already relies on multiple inter-agent communications. PEGSAgent is only concerned with the scheduling phase, but an agent-based approach may be more easily extended to cope with planning problems as well.

Another feature of agent-based systems is the potential to introduce a bidding mechanism. This means that it is possible to change the evaluation from a time-based one to an economic-based one, or possibly include time-based and economic-based factors in a single evaluation (however, our centralised system also has some experimental economic objectives that can be used). Dang and Frankovic (2002) introduce a system that is cost-based and copes with a flexible manufacturing environment. Wellman et al. (2003) explore a number of different bidding strategies. Other systems that involve bidding include Lim and Zhang (2002), Boccalatte et al. (2004) and Vancza and Markus (2000). Parunak et al. (1998) describe a comparison between agent-based modelling and equation-based modelling for supply networks and note advantages for each approach.

Main features

This section briefly describes the main features that PEGS provides. These features are available in both the centralised and distributed versions. A demo version of the program is available for downloading from our Environmental Modelling Group website (Stewart 2006).

The program allows for static or dynamic ready times and can attempt to minimise the amount of idleness in the schedule. The program allows setup times to be entered for each operation or for families of operations.

There are a variety of constraints. Operation constraints allow the user to constrain one operation to be beside or before another operation. If one operation is constrained to be beside another, then they are scheduled together on the same machine. Section "Centralised architecture" describes the centralised architecture, where for each workcentre, there is a single machine schedule generated before the operations are assigned to the parallel machines. This single machine schedule determines the order in which the operations are assigned to the parallel machines. We also have a weak form of the beside constraint, where the operations are constrained to be beside each other in the single machine schedule but can then be separated when assigned to the parallel machines. This has the effect of scheduling them at times close to each other but not necessarily on the same machine. Time-based constraints include shift times or any other time period when a machine is not available. If using shift times, it is possible to specify a preference that operations be completed on the same shift in which they started. There are also resource constraints where a maximum available quantity can be specified and the sum of the amounts consumed by the machines in use cannot exceed this. Typical resources include energy used by the machines or the number of operators required to run the machines.

It is also possible to use environmental constraints. The value of waste produced by an operation change-over 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. As waste and energy are measured in monetary units, an objective based on economic values rather than time-based values is then used. It is also possible to use cost-based objectives by entering cost estimates for factors such as the cost to process an operation on a machine per unit time. Then the time-based objectives can be converted into cost-based ones using a number of simple but experimental conversion equations we have developed.

We permit operations for the same product to overlap rather than to take place in strict sequence. The default is strictly 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 also allow the user to fix the operation time or the machine an operation should be run on. We have provided for buffers between each operation which may accumulate stock as it is produced. Users may specify any batch amount by which the items may then be removed from a buffer. 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.

The generated schedules are displayed on an interactive Gantt chart. This is colour-coded with respect to job or operation type. The user can use a key to highlight individual operations in selected jobs and enlarge/shrink the chart. The user can manually move operations to new machines/relative positions and re-schedule, or can fix operation positions and re-schedule round them. The user can also remove operations scheduled before a certain time and re-schedule the rest. This feature is also available from the main window, where the user can enter new job specifications and combine them with an existing schedule, removing operations already completed from the existing schedule. The schedule is also displayed in a textual format. This can be as a report that describes the schedule in relatively simple terms, or you can also have a full description of the scheduling details, that can actually be used to re-construct the schedule.

We have also developed a version of the centralised architecture that can be physically distributed. Each workcentre scheduling process of the shifting bottleneck algorithm can be run on a different machine in parallel, thus speeding up processing time. Results showed that the schedule quality was almost the same, while the scheduling times were much faster. For example, using three computers could speed scheduling up by a factor of nearly three.

Centralised architecture

The PEGS scheduling system uses a relatively straightforward architecture, as illustrated in Fig. 1 (previously given in Greer et al. 2006). The three main algorithms used in the process are labelled as A1.1, A1.2 and A1.3. Machines are grouped into workcentres, where a machine can belong to only one workcentre. Each workcentre can process a number of operations, where an operation can also belong to only one workcentre. Thus the machines in a workcentre can be considered as a group of parallel machines that process the same operations. PEGS uses search heuristics to find a solution. When presented with a scheduling task a number of dispatch heuristics are selected depending on the objective type and features of the problem. Each is then run and the best ordering retained. A search strategy is then used to refine the schedule and can be one of tabu search, simulated annealing or beam search. The system can be used to schedule for single machines, parallel machines, job shops or flow shops.

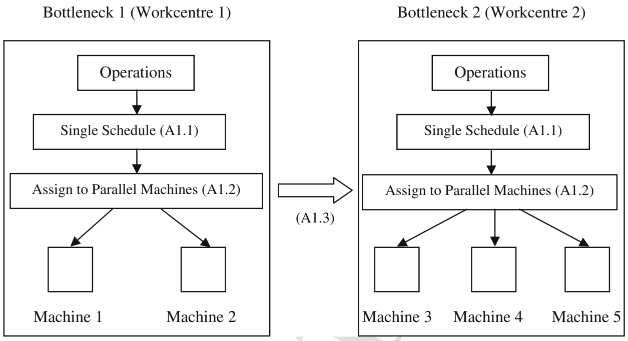

Fig. 1 Basic architecture used in PEGS. Each workcentre can be a bottleneck and each bottleneck is balanced in turn. Knowing the operations to be performed on each bottleneck, we first generate a schedule as if there were only a single machine in the workcentre. The operations are then assigned to the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

For flow shop and job shop problems, the Shifting Bottleneck algorithm is used to generate a solution in the manner described in Section "The shifting bottleneck algorithm". (A1.3 in Fig. 1). The solution at each bottleneck is obtained from a slightly modified version of the scheme suggested in Morton and Pentico (1993). A schedule is firstly generated for all operations on the bottleneck as if it was a single machine (A1.1 in Fig. 1). When calculating completion times (used to calculate the objective) for this step we then take into account the number of parallel machines in the workcentre. The operations are assigned to each parallel machine in sequence 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. For example, if the first two operations in the single machine schedule are assigned to different machines, they may be given the same completion times. The completion times if just considering one machine would of course be different as one would be scheduled after the other. These estimated completion times are then used to calculate the objective for the single machine ordering and the best ordering retained. We expect this technique to be particularly effective when measuring the earliness/tardiness objective (operations should be neither too early nor too late). The operations are then actually assigned to the parallel machines in the order of the single machine schedule, in the most economic manner (A1.2 in Fig. 1). We have two algorithms to do this. The first simply assigns the operation to the machine that can process it first. The second algorithm tries to place similar operations together on individual machines. This second algorithm is good for minimising waste if environmental factors are considered. This architecture is clearly centralised. A single algorithm considers all machines in a workcentre together. The bottlenecks must also all be processed together. This leads to more complex algorithms to process the information, but the information is all present in one place and there is little need for communication between different parts of the system. In the distributed system described in the next section there is substantial need for communication between the agents. So while the distributed solution provides simpler individual problems, there is increased complexity through the communications.

Distributed architecture

In PEGSAgent there are agents to represent machines and jobs. A machine agent represents a single machine. It is responsible for finding the best schedule for that machine. There is also a machine mediator agent that controls some global operations required on all machines. The machine mediator will create the machine agents and also hold global machine information, such as which machine processes which operations. A job agent is responsible for finding the best schedule for the operations in its job. There is also a job mediator agent that controls global operations required on all job agents. The job mediator creates the job agents and also determines which job agent will try to schedule the next operation. The actual scheduling process is conducted between the machine and job agents. The negotiation protocol used here is the Contract Net protocol (FIPA 2002). Figure 2 shows the main architecture used.

There is one main process which is used to generate a schedule and there are then two variations on this. In both variations, the job mediator generates and initialises the job agents and the machine mediator generates and initialises the machine agents. In the first step of the process to generate a new schedule, the job mediator determines the order in which the operations should be scheduled. It does this by generating an ordering for each workcentre using one of the search heuristics for the single machine phase of the Shifting Bottleneck algorithm described above (A2.1 in Fig. 2). The operation placed first in this ordering is then given an *order value* of 1 and so on. These order values are sent to each job agent for each operation (A2.2 in Fig. 2). The job mediator then asks each job agent in turn to return the order value for the next operation that it needs to schedule. The operation with the lowest order value is given permission to be scheduled next (A2.3 in Fig. 2). Note that, within an individual job, the operations are required to be processed in a certain sequence to produce that product, and this cannot be over-ridden by preferences due to order values.

When a job agent is given permission to schedule its next operation, it retrieves the addresses of the machines that can process it and asks each in turn for a schedule time and evaluation. This negotiation is carried out using the Contract Net protocol. In this protocol, a job agent will ask the machine agents for times at which they can schedule its operation. When the machine agents reply, the job agent evaluates all of the replies and chooses the best one. It accepts the best proposal and rejects the rest. The accepted machine agent is then asked to actually schedule the operation at the time that it specified (A2.4 in Fig. 2). The machine agents determine the best time that they can schedule the operation taking account of their existing schedules. A number of factors are used to evaluate a time slot. These include the actual time and date, the evaluation of the objective for the machine if that slot is selected and also some budgetary factors are considered. Budgets are only considered if time and objective evaluations do not indicate a winner. We intend to look further at the budget and bidding processes at a later stage.

In this way, all of the operations are eventually assigned to a machine. When all operations have been assigned, the job mediator asks the machine mediator to combine the partial schedules from each machine agent and return this as the final schedule (A2.5 in Fig. 2). So a search heuristic is used to generate the initial ordering, to determine the order in which the operations are allowed to be scheduled. When a machine is then asked to schedule an operation it uses a local sort to determine the best position. It tries out each possible slot from the earliest legal date to the end of its schedule. It places the new operation in each slot and re-evaluates its objective. The slot that produces the best objective is returned as the proposal from that machine.

The sorting process is part of A2.4 in Fig. 2. One interesting feature to notice about this sorting process is that it is constructive. Beginning from no operations on a machine, the count is incremented one at a time as the schedule is created. If, eventually, there are five operations on a machine,

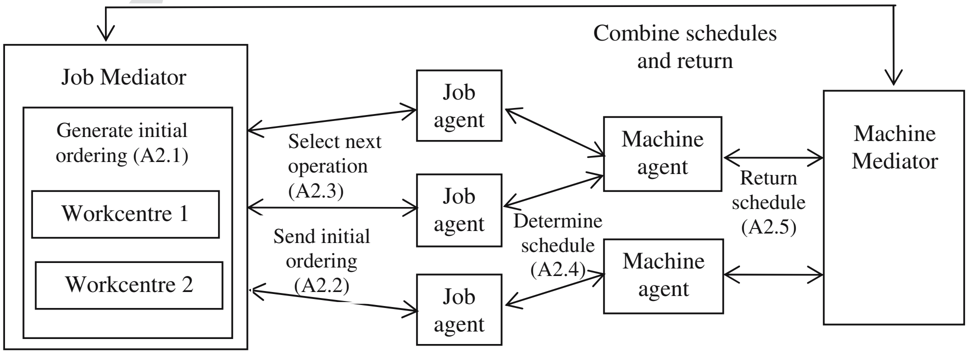

Fig. 2 Diagram illustrating the agent interactions to generate a schedule

the number of sort iterations required may be something like 1+2+3+4+5 for the different scheduling requests on the machine. This is a lot less than using a search strategy which would have to deal with the whole schedule during each iteration and for a possibly large number of iterations. The Shifting Bottleneck approach uses the search strategies and is an iterative process, so searches are performed several times. The agent-based approach is much quicker as there is only one search process at the start. This will be shown to dramatically reduce search times for a job shop environment. The advantage is not so good for other environments such as a flow shop. In this case the centralised approach performs only one search for each bottleneck and so the agent-based approach performs an extra sort on the machines in addition to this. However, for a generic system like PEGSAgent, since scheduling job shops takes a lot longer than scheduling flow shops, the advantage for job shops would out-weigh the time lost on flow shop problems.

Having described the basic scheduling process, we now examine the two variations mentioned above. When a new operation is inserted into a slot in an existing schedule, it may change the times of operations that have already been scheduled. Two approaches have been tried to cope with this. The first approach is to remove all operations whose times are affected and then try to re-schedule them again on some machine. This is similar to the idea of Boccalatte et al. (2004), who give the operation to be re-scheduled a favourable order value to ensure it is re-scheduled relatively soon after being moved. However, we found some problems with this approach. There was a tendency for the scheduling process to cycle, when an operation would be inserted early in an existing schedule. There would then be a ‘knock-on’ effect to move later operations, which would be re-scheduled on some other (or the same) machine. This might then move other operations which follow the new position, which would be re-scheduled on some other (or the same) machine, etc. This would particularly be the case if the operations being scheduled were very similar to each other. To prevent this cycling a maximum iteration count was enforced. If any operations were re-scheduled a maximum count number of times, they would then be added in a place that did not affect any existing schedule. However, this modification resulted in operations being added at the end of the existing schedule. The test section will show that while some reasonable schedules were obtained, another approach proved slightly better.

In a second approach, when an operation was inserted at some intermediate position in a machine schedule, the times of the operations that came after it on that machine were pushed forwards to accommodate the new operation. The times of any dependent operations (due to product sequence requirements) on different machines were also pushed forwards. However, in the case of workcentres with parallel machines, any of the affected operations were also permitted to be re-scheduled on a similar machine so long as it did not affect the existing schedule on that machine. If an operation that was pushed forward could be moved to a more favourable position on a different machine without affecting its existing schedule, then this would be allowed.

Comparison with other systems

The original aims for developing this system included provision to account for environmental factors in the scheduling process. This included optimising with regard to waste and energy, as well as cost. Another aim was to produce a generic system which would be adaptable to a wide range of scheduling tasks. The main manufacturing base in Northern Ireland is SMEs and so a flexible system that could be used in different scenarios, but would not necessarily have to process very large schedules, was the target application. For these reasons, search heuristics were considered acceptable. However, after looking at other systems and through practical experience with manufacturer’s and their particular requirements, many other useful features were added to the system. We also reviewed a range of currently available commercial and academic programs. In comparison to the commercial systems in common use at local companies, our system appears to have several novel features. The constraint options in PEGS are similar to those in many systems, but whereas it is common to use dispatch heuristics for the whole scheduling process, we prefer to use search heuristics for the main process. We have also now implemented the agent-based version. The inclusion of environmental and economic features is also new. We found other systems that offer the overlap or ‘accumulation’ processing options, but none seemed to provide our ‘weak’ beside constraint. In summary, while we found many of the features of PEGS in other systems, we did not find any other system that offered all of the features that we have included.

Having a typical modern graphics-based user interface, we imagine that our system should be relatively easy to use, although help will be required to use the various additional features that are available. This comes in the form of a user manual and online help. The most significant task is in setting up the initial environment. For this a user needs to specify their shop floor layout and the operations that each machine can process. Thus, details of each machine must be entered and then the types, costs and times for processing each operation in each workcentre needs to be specified. Product details can then be entered as a sequence of operations that can be constrained in any allowed way, including the overlap of the processing of individual operations. Processing details are in time units to process a single item. The product details then specify the number of items per operation. Once the initial details are entered however, they can be saved to an XML file and re-loaded each time the system is used. If they are relatively permanent then this process only needs to be done once. The system also tries to be helpful, by providing all stored information in the form of combo-box options when required, and even indicating the order in which to enter information by only enabling the options that are legal for the next stage. To make daily use of the system, the user then only needs to enter the details of the new jobs to schedule. It is also possible to save schedules and re-load them at the next scheduling stage. Old schedules can be updated and re-scheduled with new jobs, where completed jobs will be removed and the new jobs integrated. This allows for some level of continuous scheduling, even though the system is not a dynamic online one. We provide an interactive Gantt chart, giving a clear schedule layout and allowing for interactive re-scheduling.

Testing the two approaches

In this section we describe some tests performed to illustrate the advantages of the two different agent approaches. The tests considered both solution quality and execution times.

Benchmark datasets

The first set of tests compares the quality of the two agent-based approaches and the conventional program using well-known benchmark datasets. The values given for the 'Optimal objective' in Table 1 are the best published results which have been achieved to date. Our generic system does not produce such high quality schedules but it has not been 'tuned' to any particular problem type or dataset. As a consequence, the individual evaluations from PEGS are not so interesting, but the results demonstrate that PEGS works well for a range of different problems. Table 1 lists some datasets which have been used to test the scheduling quality and the best objective values from PEGS averaged over a small number of runs.

Generally, the objective values obtained from PEGS were not as good as the best published values, though there were three cases in the Purdue data where the PEGS objective was better. The Flexible Manufacturing System example is also interesting. This system allows any of the machines to process any of the operations but not all machines process all operations. This was implemented in PEGS as a Parallel Machine configuration (single group of parallel machines) where not all machines processed all operations. On average, the PEGS Shifting Bottleneck algorithm performed just slightly better than PEGSAgent2, while PEGSAgent1 performed worst. This suggests that the second agent-based approach produces good quality solutions compared to our implementation of the centralised approach. All schedules were solved in relatively short amounts of time (just a few seconds), so execution time was not a problem. These results show that while PEGS and PEGSAgent may not produce optimal solutions, they will tend to produce good solutions in most cases.

Random datasets

The three approaches were also tested on randomly generated datasets for flow shop and job shop configurations. The objective measured was weighted makespan (because a search strategy is used for this objective). In all cases the search strategy used was tabu search and each measurement was averaged over three test runs.

Flow shop tests

This set of tests was for three different flow shop configurations. The objective evaluation and the execution times are given in Table 2 for different factory configurations. Flow Shop 1 (FS1) has three workcentres (WC), 20 jobs (J) and 60 operations (Op). Flow Shop 2 has a five workcentres, 30 jobs and 150 operations. Flow Shop 3 has seven workcentres, 25 jobs and 175 operations. Each workcentre contained a number of parallel machines. Flow Shop 1 had 11 machines, Flow Shop 2 had 18 machines and Flow Shop 3 also had 18 machines. Each test run was for 100,000 iterations at each search step.

The results show that execution time is not greatly affected by the extra sort of the agents. Clearly this is a much quicker process than the search process. To extend the analysis to reflect more practical situations, some tests were conducted with heavily constrained data. This data included setup times, waste costs and shift and time constraints. In these cases the agent sort took longer, but, for the size of datasets we are measuring here, it was still measured in seconds. When scaling up to larger datasets, the sort with heavily constrained data would require a more significant amount of extra time.

In two of the three tests PEGSAgent2 produced the best objectives, while in the other test it was PEGSAgent1. This outcome supports a decision to use an agent-based approach. It is possible that the PEGSAgent1 is better in the first set of tests because there are fewer operations. In this case more operations will be 'properly' scheduled before the maximum iteration count is reached and the remaining are placed at the end.

Job shop tests

This set of tests was for three different job shop configurations. The objective evaluation and execution times are given in Table 3 for the different schedule configurations. Job Shop 1 (JS1) has three workcentres (WC), 20 jobs (J) and 80 operations (Op). Job Shop 2 has five workcentres, 30 jobs and

Table 1 List of benchmark datasets used to test the quality of the different approaches

Benchmark dataset[a]	Objective type	Shop type	Optimal objective	PEGS objective	PEGSA1 objective[b]	PEGSA2 objective[b]
abz5	Makespan	Job shop	1234	1296	1352	1307
abz6	Makespan	Job shop	943	1018	981	984
abz7	Makespan	Job shop	655	772	809	774
la19	Makespan	Job shop	842	998	951	951
la20	Makespan	Job shop	902	964	999	975
flcmax_20_15_3	Makespan	Flow shop	4437	4393	4713	4486
flcmax_20_15_4	Makespan	Flow shop	3779	3865	4095	4056
flcmax_20_15_6	Makespan	Flow shop	4144	4128	4227	4163
fl_20_15_1_1_2	Lateness	Flow shop	2833	3025	3033	3033
fl_20_15_1_1_3	Lateness	Flow shop	2322	2608	2789	2489
fl_20_15_2_1_5	Lateness	Flow shop	3651	3692	3728	3633
fl_20_15_2_1_6	Lateness	Flow shop	3360	3507	3516	3419
Day1	Makespan	Flexible flow shop	760	791	826	831
Day2	Makespan	Flexible flow shop	770	826	837	842
Day3	Makespan	Flexible flow shop	770	827	818	833
Day4	Makespan	Flexible flow shop	785	820	857	827
Day5	Makespan	Flexible flow shop	961	986	1031	1031
Day6	Makespan	Flexible flow shop	667	701	706	706
Dataset1	Makespan	Flexible manuf.	420	391	411	441

The optimal objectives and the objective values we found by each approach are given

[a] abz is Adams et al. (1988), la is Lawrence (1984), flcmax are the makespan datasets from Purdue (in Dimirkol et al. 1998), fl are the lateness datasets from Purdue (in Dimirkol et al. 1998), Day is Wittrock (1988) and Dataset1 is Kumar et al. (2003)

[b] PEGSA1 and PEGSA2 refer to the PEGSAgent program operating with the first and second variations described in the text

Table 2 Comparison of objective values and execution times (s) from PEGS variants (100,000 iterations on each search and varying factory configurations)

Dataset		Program version					
		PEGS		PEGSA1		PEGSA2	
		Objective	Time	Objective	Time	Objective	Time
FS1	WC: 3 J: 20 Op: 60	16,124	11	15,987	11	16,115	12
FS2	WC: 5 J: 30 Op: 150	63,109	25	66,366	25	58,938	25
FS3	WC: 7 J: 25 Op: 175	59,121	31	58,728	30	55,100	32

FS is flow shop, WC is number of workcentres, J is number of jobs and Op is number of operations

150 operations. Job Shop 3 has seven workcentres, 25 jobs and 125 operations. Each workcentre consisted of a number of parallel machines. For Job Shop 1 there was a total of 11 machines, for Job Shop 2 a total of 15 machines and for Job Shop 3 a total of 18 machines. Each test run was for 100,000 iterations for each search step.

These results show a clear superiority of PEGSAgent1 and 2 over the centralised approach in PEGS. The agent-based approaches produced better objectives, with the second version being better in all three tests. Considering the execution times, the agent-based approaches are 6–15 times faster than the centralised approach with the greatest advantage associated with the larger problems. These tests also show that for the centralised system (PEGS) the number of workcentres is a more important influence on execution time than the number of operations. The third test had fewer operations than the second but more workcentres. Each workcentre requires its own search and, coupled with balancing of more bottlenecks, the total number of searches is greater. As the agent-based approach only performs one search at the start there are no problems with extra iterations and so it can produce schedules for JS2 and JS3 in much the same time.

From the flow shop and job shop results, it would appear that the second agent-based approach, PEGSAgent2, is the

Table 3 Comparison of objective values and execution times (s) from PEGS variants (100,000 iterations on each search and varying factory configurations)

Dataset		Program version					
		PEGS		PEGSA1		PEGSA2	
		Objective	Time	Objective	Time	Objective	Time
JS1	WC: 3 J: 20 Op: 80	27,917	89	24,053	15	23,373	15
JS2	WC: 5 J: 30 Op: 150	131,710	266	124,658	27	108,075	27
JS3	WC: 7 J: 25 Op: 125	78,802	404	59,636	26	57,788	26

JS is job shop, WC is number of workcentres, J is number of jobs and Op is number of operations

most consistent in terms of quality and execution speed and would therefore be the preferred option.

Conclusions

PEGS is a generic production scheduling system. The initial implementation used a centralised solution based on search strategies and the Shifting Bottleneck algorithm, and has proved to reliably provide good schedules for a wide range of flow shop or job shop problems. However, this technique is necessarily slow. One possible alternative is an agent-based approach and we have suggested a basic framework for using the agents. This includes an initial search and then a much shorter sort on the agents. This method is a similar idea to Boccalatte et al. (2004) but there are several differences. We use a different method for calculating the initial order. They use the job slack time to determine the ideal job start time and also use a probability function to determine if the job agent then issues a call for proposals in the Contract Net protocol to initiate a schedule. The machine agents then order the job proposals based on the bid values. They also re-schedule jobs in a different way, by assigning a higher priority to the jobs being re-scheduled and then re-calculating their bid values. The Contract Net protocol seems to cover their whole scheduling process. Their system is for just-in-time manufacturing where jobs arrive dynamically and are scheduled as they arrive. Our system is currently more suited to a situation where there are a number of jobs that need to be scheduled and they are all scheduled at one time, say at the start of a week.

We have suggested two variations on using the agent-based approach. These both treat the agents as individual entities with no communication between them to try to obtain a better global schedule. The machine agents try to minimise the schedule for their machine only and do not consider the schedule on other machines. This could reasonably be expected to produce good results, for if each machine agent tries to minimise its schedule this will mean the global objective will also be minimised. Shen (2002) writes that there are also some agent systems where the agents communicate with each other in an attempt to minimise some global objective. We have also considered this approach and are testing a third variation that takes this into account. In the third variation, when a schedule is calculated for one machine, the schedules on all machines are updated. All machines then calculate their objectives and these are returned as well as the local machine objective. It is then possible to consider the objectives of all machines. If the worst of these is better than the current best (in terms of its impact on the overall objective), then even if the local objective is worse, this schedule may be preferred.

The test results show that agent methods, compared to our centralised system, can produce schedules that are only slightly worse on benchmark datasets and actually better on our own random datasets. The agents can also save a large amount of processing time, in some cases by a factor of 10 or more for job shops. This would mean that the agent-based system would be able to handle larger numbers of jobs. It seems that agents may not be able to produce optimal solutions but can produce good solutions, though Vancza and Markus (2000) show that near optimal solutions may be achievable in some cases, and we also found this. The lack of information that each agent possesses may make it difficult to produce optimal solutions.

We would suggest that in a generic or commercial system of this type, an agent-based approach would be a good option. This is because these types of systems may be more interested in providing a range of features than in producing optimal solutions. If optimality is not the over-riding issue, the flexibility and speed of the agent-based approach may be preferred for today's dynamic scheduling environment.

A demo version of the program is available for downloading from our Environmental Modelling Group website (Stewart 2006). This includes PEGS and PEGSAgent2. An additional distributed program will allow operation of the distributed PEGS version.

Acknowledgements The project is funded by Invest Northern Ireland's Centres of Excellence programme, through the University's QUESTOR Environmental Research Centre. QUESTOR Environmental Tools is an interdisciplinary project designed to carry out leading edge research which will deliver a new generation of cost-effective and practical tools for analysis and monitoring of the environment. The development of PEGS is part of the QUESTOR Tools programme.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*(3), 391–401.

Babiceanu, R. F., & Chen, F. F. (2006). Development and applications of holonic manufacturing systems: A survey. *Journal of Intelligent Manufacturing, 17*, 111–131.

Balas, E., & Vazacopoulos, A. (1998). Guided local search with shifting bottleneck for job shop scheduling. *Management Science, 44*(2), 262–275.

Boccalatte, A., Gozzi, A., Paolucci, M., Queirolo, V., & Tamoglia, M. (2004). A multi-agent system for dynamic just-in-time manufacturing production scheduling. *IEEE International Conference on Systems, Man and Cybernetics, 6*, 5548–5553.

Chambers, J. B., & Barnes, J. W. (1996). New tabu search results for the job Shop scheduling problem. http://citeseer.ist.psu.edu/chambers96new.html (Last accessed: 15/5/06).

Dang, T.-T., & Frankovic, B. (2002). Agent-based scheduling in production systems. *International Journal of Production Research, 40*(15), 3669–3679.

Dimirkol, E., Mehta, S., & Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research, 109*, 137–141.

(FIPA) Foundation for Intelligent Physical Agents. (2002). FIPA contract net protocol interaction specification, http://www.fipa.org/specs/fipa00029/SC00029H.pdf (Last accessed: 15/5/06).

Greer, K., Stewart, J., & McCollum, B. (2006). The PEGS scheduling system: A case study with environmental optimisation. *The 36th International Conference on Computers and Industrial Engineering (ICCIE'06)*, Taiwan, June 20–23, pp. 1185–1196.

He, D., & Babayan, A. (2002). Scheduling manufacturing systems for delayed product differentiation in agile manufacturing. *International Journal of Production Research, 40*(11), 2461–2481.

Hertz, A., Taillard, E., & de Werra, D. (1992). A tutorial on tabu search. http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf (Last accessed: 15/5/06).

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence, 117*(2), 277–296.

Kumar, R., Tiwari, M. K., & Shankar, R. (2003). Scheduling of flexible manufacturing systems: An ant colony optimisation approach. In: *Proceedings of the Institution of Mechanical Engineers, 217*(Part B), 1443–1453.

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

Lim, M. K., & Zhang, Z. (2002). Iterative multi-agent bidding and coordination based on genetic algorithm. In *Proceeding of 3 Complex Systems, and E-businesses*, Erfurt, 7–10 October, pp. 682–689.

Morton, T. E., & Pentico, D. W. (1993). *Heuristic scheduling systems*. New York: Wiley Series in Engineering and Technology Management.

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

Parunak, H. V. D., Savitt, R., & Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. *Proceedings of Multi-agent systems and Agent-based Simulation (MABS'98)*, pp. 10–25.

Pinedo, M. (2002). *Scheduling: Theory, algorithms and systems*. New Jersey: Prentice Hall.

Shen, W. (2002). Distributed manufacturing scheduling using intelligent agents. *IEEE Intelligent Systems*, pp. 88–94.

Shen, W., & Norrie, D. H. (1998). An agent-based approach for dynamic manufacturing scheduling. In *Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems ((PAAM)-98)*, pp. 533–548.

Stewart, J. R. (2006). QUESTOR environmental modelling research group, Queen's University, Belfast. http://questor.qub.ac.uk/webpages/whatwedo/researchgroups/environmentalmodelling/qcindex.html (Last Accessed 15/10/07).

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

Vancza, J., & Markus, A. (2000). An agent model for incentive-based production scheduling. *Computers in Industry, 43*, 173–187.

Walker, S. S., Brennan, R. W., & Norrie, D. H. (2005). Holonic job shop scheduling using a multiagent system. *IEEE Intelligent Systems, 20*(1), 50–57.

Wellman, M. P., Mackie-Mason, J. K., Reeves, D. M., & Swaminathan, S. (2003) Exploring bidding strategies for market-based scheduling, EC'03, pp. 115–124.

Wittrock, R. J. (1988). An adaptable scheduling algorithm for flexible flow lines. *Operations Research, 36*(3), 445–453.

Yen, B. P.-C. (1998). Agent-based distributed planning and Scheduling in Global Manufacturing. *Proceedings of the 3rd Annual International Conference on Industrial Engineering Theories, Applications and Practice*.

Yoo, M.-Y., & Muller, J.-P. (2002). Using multi-agent system for dynamic job-shop scheduling. *ICEIS 2002, Fourth International Conference on Enterprise Information Systems*.