Dynamic scheduling for complex engineer-to-order products

Christian Hicks, Dongping Song, Christopher Earl

Dynamic scheduling for complex engineer-to-order products

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2005-IJPR-0450.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	05-Apr-2006
Complete List of Authors:	Hicks, Christian; University of Newcastle upon Tyne, School of Mechanical and Systems Engineering Song, Dongping; University of Plymouth, Business School Earl, Christopher; Open University, Design and Innovation
Keywords:	OPTIMIZATION, EVOLUTIONARY ALGORITHMS, DYNAMIC SCHEDULING
Keywords (user):	ENGINEER-TO-ORDER, RESCHEDULING

Dynamic scheduling for complex engineer-to-order products

Hicks C.[1], Song, D.P.[2], and Earl C.F.[3]

[1]University of Newcastle upon Tyne Business School,

Newcastle upon Tyne, NE1 7RU, UK

Tel: 0191 2226238, Fax: 0191 2228600

Email: Chris.Hicks@ncl.ac.uk

[2]International Shipping & Logistics Group, The Business School,

University of Plymouth, PL4 8AA, UK

Email: Dongping.song@plymouth.ac.uk

[3]Department of Design and Innovation, The Open University

Walton Hall, Milton Keynes, MK7 6AA, UK

Email: c.f.earl@open.ac.uk

Keywords: Dynamic scheduling; Engineer to order; Rescheduling; Optimisation; Evolution Strategy.

Abstract:

This paper considers dynamic production scheduling for manufacturing systems producing products with deep and complex product structures and complicated process routings. It is assumed that manufacturing and assembly processing times are deterministic. Dynamic scheduling problems may be either incremental, where the schedule for incoming orders does not affect the schedule for existing orders, or regenerative where a new schedule is produced for both new and existing orders. In both situations, a common objective is to minimise total costs (the sum of work-in-progress holding costs, product earliness and tardiness costs).

In this research, heuristic and Evolutionary Strategy based methods have been developed to solve incremental and regenerative scheduling problems. Case studies using industrial data from a company that produces complex products in low volume demonstrate the effectiveness of the methods. Evolution strategy provides better results than the heuristic method, but this is at the expense of significantly longer computation times. It was found that performing regenerative planning is better than incremental planning when there is high interaction between the new orders and the existing orders.

1. Introduction

Engineer-to-order (ETO) products are manufactured and assembled in low volume to satisfy individual customer's specifications (Song 2001). Typical products include capital goods such as large steam turbines and boilers for the power generation industry, bespoke cranes and oilrigs. A survey of planning practices revealed that engineer-to-order companies use project planning methods for the high level scheduling of projects which usually include construction activities at the customers' sites. Manufacturing Resources Planning (MRP II) is normally used for planning the factory-based manufacture and assembly of major components and systems. Most companies only update their plans on an infrequent basis (Hicks, 1998, Song et al. 2002).

Scheduling was defined by Baker (1974) as the "the allocation of resources over time to perform a collection of tasks". A schedule specifies sequence and timing, normally expressed in terms of a set of start and due times. Dynamic scheduling aims to update an existing schedule by reacting to the occurrence of unpredictable events, such as dynamically arriving orders or machine breakdowns. Dynamic scheduling may be performed periodically or when some particular events occur (Church and Uzsoy 1992, Artigues et al. 2003). There are two types of dynamic scheduling. Incremental scheduling aims to find a schedule for new orders that does not affect the schedules for existing orders, even if the same resources are used. With incremental scheduling, the due dates are fixed, but the lead-time for new orders may be very long because the existing products always take precedence. Regenerative scheduling generates a new schedule for operations in both new and existing orders.

In ETO manufacturing, demand fluctuates considerably and tends to occur in large discrete units (Hicks 1998). In practice, most rescheduling activity is due to order arrivals and other issues such as machine breakdown tend to be relatively unimportant. This paper therefore focuses upon the arrival of new orders. It analyses and compares the use of dynamic scheduling algorithms, based upon heuristics and Evolutionary Strategy, for planning the manufacture and assembly of complex products under finite capacity conditions. Incremental scheduling can produce unacceptably long lead-times for new orders. With dynamic scheduling the arrival of a new order can have a large effect upon the existing schedules, which may result in late deliveries and the imposition of financial penalties. It is therefore necessary to have a scheduling approach which achieves an appropriate compromise between quoting realistic due dates for new orders, whilst ensuring the satisfactory fulfilment of existing orders.

A case study is presented that uses industrial data from a collaborating ETO company. Finite capacity is assumed. The products are represented using hierarchical product structures. The root nodes represent the final products, whilst the leaf nodes represent components. Different products require processing on the same manufacturing and assembly resources, which causes contention for resources and interactions between orders. The performance measure used in this research is total cost, which is the sum of work-in-progress holding costs, product earliness costs and product tardiness costs. Previous research has neglected the rescheduling complex assemblies with such non-regular performance measures (Baker and Scudder, 1990).

The remainder of the paper is organised as follows. In the next section, the dynamic scheduling problem is reviewed and the problem under consideration is mathematically formulated. Incremental planning and regenerative planning problems are then described and tackled in sections 3 and 4 respectively. Heuristic and Evolution Strategy methods are presented to solve these problems. The heuristic method is based on finite loading. Evolution Strategy, a random search method, is used to find optimal planned operation start times. In section 5, the methods are tested through an industrial case study, using data from an ETO company. This is followed by a comparison of incremental planning and regenerative planning in section 6. In section 7, the use of the methods is discussed and other real life factors are considered. Finally, conclusions are made in section 8.

2. Dynamic scheduling

In the literature, the terms 'dynamic scheduling' and 'rescheduling' are often used interchangeably. Dynamic scheduling research has addressed single machine, job shop and project rescheduling. For single machine systems Church and Uzsoy (1992) developed a hybrid event-driven rescheduling strategy that combined periodically time-driven and event-driven modes that responded to dynamic job arrivals. Vieira et al. (2000) described analytical models that predicted the average flow time and machine utilisation. They used a first-in-first-out (FIFO) algorithm to reschedule the new jobs, along with those existing jobs that had not started. Their results showed that the analytical models could accurately predict performance. Cowling and Johansson (2002) presented a dynamic scheduling scheme that traded off the quality of the revised schedule against the production disturbance resulting from changing the planned schedule. A simulation of a single machine system demonstrated that there is a continuum of strategies, allowing the decision maker to trade off utility and stability performance when choosing an appropriate strategy.

In job shop situations Yamamoto and Nof (1985), Li et al. (1993) and Abumaizar and Suestka (1997) investigated rescheduling in response to disruptions caused by machine breakdowns. Matsuura et al. (1993), Jain and Elmaraghy (1997), Chang (1997) and Chryssolouris and Subramaniam (2001) considered dynamic scheduling relating to the arrival of new order arrivals. The general limitation of this job shop research is that it neglects assembly operations, product structure and the need to coordinate the supply of components to meet assembly requirements. These factors are critically important when scheduling products with many levels of assembly.

The majority of research into project planning has focused on static project scheduling, which results in a single plan, with no replanning to take into changes that occur due to uncertainty or disruption (Brucker et al. 1999). Artigues et al. (2003) presented a polynomial activity insertion algorithm and performed computational experiments for both static and dynamic resource-constrained project scheduling problems. The objective was to minimise the total project duration by completing operation activities as early as possible. However, the schedules generated did not lead to the most favourable outcome in terms of costs. Traditional regular performance measures are non-decreasing functions of the job completion times (for example maximum lateness and total weighted flow time). Schedules that are optimised in terms of regular measures do not include any idle time and can be determined from the operation sequence (Brandimarte and Maiocco 1999). Another common way of evaluating schedules is to use total cost, which includes earliness, tardiness and stock holding costs (Baker and Scudder 1990). Total cost is a non-regular performance measure because both job sequencing and timing should be optimised, which is much more difficult than optimising regular measures of performance (Brandimarte and Maiocco 1999).

2.1 Problem formulation

In this research, production schedules were represented by a set of planned operation start times, which denoted the earliest time that the operations could start. The sequence of operations was implied by their timing. When a plan is implemented the actual operation start times may deviate from the plan. After a plan has been executed, the earliness, tardiness and total costs can be evaluated. The following notation will be used:

s_i	– the planned start time of operation i, which is a decision variable;
x_i	– the processing time of operation i;
a_i	– the actual time operation i started to be processed when the plan was implemented;
c_i	– the actual time that the processing of operation i was completed;
d_i	– the due time of operation i, if it is the last operation of a product;
C_i	– the immediate predecessors of operation i in the product structure;
$\rho(i)$	– the operation that immediately follows the operation i in the product structure;
$r(i)$	– the resource which performs the operation i;
$\varphi(i)$	– the operation that immediately precedes the operation i on the resource $r(i)$;
h_i / h_i^-	– the unit time earliness / tardiness costs for operation i.

In order to describe the dynamic situation, additional notation is required. Let Γ_1 denote the total operation set for the existing orders; L_1 denote the set of the last operations of all products in the existing orders (called the product set); and R_1 denote the resource set (the resources used for the existing orders). Similarly, let Γ_2, L_2 and R_2 denote the total operation set, product set and resource set for the new arriving order respectively. Let $\Gamma=\Gamma_1\cup\Gamma_2$, $L=L_1\cup L_2$, $R=R_1\cup R_2$.

The incremental planning problem therefore aims to find an optimal schedule for all operations in Γ_2 (i.e. s_i, $i\in\Gamma_2$) that has the lowest cost in terms of the following cost function:

$$J(\boldsymbol{s}) = \sum_{i\in\Gamma 2\backslash L2} h_i(a_{\rho(i)} - c_i) + \sum_{i\in L2} h_i max(d_i - c_i, 0) + \sum_{i\in L2} h_i^- max(c_i - d_i, 0) \tag{1}$$

s.t. for any $i\in\Gamma_2$

$$a_i = max(s_i, c_{\varphi(i)}, \{c_j \mid j\in C_i\}) \tag{2}$$

$$c_i = a_i + x_i \tag{3}$$

$$a_i \geq arrivalTime \tag{4}$$

$$a_i \geq c_j \text{ or } c_i \leq a_j, \textit{for any } j\in\{j\in\Gamma_1 \mid r(j)=r(i)\} \tag{5}$$

Where ***s*** is a vector of s_i ($i \in \Gamma_2$), and $\Gamma_2 \backslash L_2$ is the difference set of Γ_2 and L_2. In equation (1), the first term is the total work-in-progress holding cost, the second is product earliness cost and the third is the product tardiness cost.

Equation (2) represents the three constraints: planning constraints (i.e. actual start times cannot be earlier than the planned start times); resource constraints (i.e. resources can only perform one operation at a time); and precedence constraints (e.g. an assembly cannot be performed until all preceding operations have been completed). Equation (3) specifies that operations are not allowed to be interrupted. Inequalities (4) and (5) represent two new extra constraints for the operations associated with the new coming orders: i) each operation cannot start before the arrival time; and ii) the existing orders create resource constraints. However, there are two special conditions that may eliminate the constraint given in equation (5). First, if the new order arrival time is later than $max\{c_i \mid i \in \Gamma_1\}$, then there is no interaction between the new order and the existing orders and (5) is satisfied. Second, if $R_1 \cap R_2 = \phi$, i.e. the new order and the existing orders have no common resources, then (5) is also satisfied.

3. Incremental planning

The aim of incremental planning is to generate a schedule for new orders that does not change the production schedules for existing orders. In static scheduling there are three constraints: resource constraints; operation precedence constraints; and due date constraints. With dynamic scheduling two extra constraints arise. Firstly, the initial start time for the new order cannot be earlier than its arrival time. Secondly, the resource constraints depend upon both new and existing orders.

With incremental planning the capacity available may be viewed in terms of ‘time slots’ when the resource is not committed to producing the existing orders. The incoming orders should be loaded to fill in these ‘time slots’ if they are feasible for the arriving operations.

3.1 Problem formulation

To implement incremental planning, the operations in the new order are filled into the idle periods of resources by taking into account the schedules for the existing orders. There are two alternative approaches. Horizontal loading (Vollmann et al. 1992) loads one entire order (the order with the highest priority) for all its operations, then the second highest priority order, and so on. Each component or assembly within a product may be considered as a 'job', which may comprise a series of operations. The job-oriented, or vertical scheduling approach (Yeh 1997) ignores product structure relationships and schedules jobs one at a time. The shortcoming of this approach is that different components and assemblies within the same product can be assigned different priorities, which leads to poorly synchronised plans.

Three methods are presented to deal with the incremental dynamic scheduling problem and are compared using case studies. Heuristics are considered that are based upon forwards/backwards finite loading and priority rules. Theses were based upon Vollman et al. (1992) and Yeh (1997). The third method uses the Evolutionary Strategy (Schwefel, 1995; Schwefel and Back, 1998), which has not been applied to incremental planning problems before.

3.2 Forward incremental planning

Forward incremental planning starts by loading components onto resources. It then moves level by level up through the product structure finishing with the final assembly operations. This approach provides an estimated completion time, which can be confirmed with the customer before the order is started. This process will now be described more formally.

A ready-to-plan operation set is defined as a set that is composed of all operations that have no unscheduled preceding operations. Forward incremental planning (FIP) starts from the new order arrival time. First, it selects the highest priority operation from the ready-to-plan operation set using a specified priority rule. The initial ready-to-plan operation set consists of all the operations in the new order that have no preceding operations. It then finds the first 'feasible' period that the resource required for the first operation is available. To be 'feasible', the period must satisfy two conditions: it needs to long enough for the operation to be performed; and the start time of the period must not be before the arrival time. After finishing the planning of the operation, FIP checks whether its immediately successive operation is ready. If it is, this operation is put into the ready-to-plan operation set. Then FIP selects the next operation from the ready-to-plan operation set and repeats the process. The feasible available period for this operation must not be earlier than the preceding operation's completion time. The period should also be enough to perform the operation. This procedure continues until all the operations associated with the new order have been planned.

Let $\Lambda(r)$ be the set of operations which belong to the existing orders and have been scheduled on the resource r. It is assumed that this set has been sequenced with the earliest start time first. In the following procedure, the notation $\Psi(r)$ denotes the ready-to-plan operation set, $\Phi(r)$ denotes the planned operation set on resource r, and $|\Psi(r)|$ denotes the number of operations in $\Psi(r)$. The detailed steps of the FIP algorithm are:

Algorithm for Forward Incremental Planning (FIP):

Step 1: Initialisation:

1) Set $k = 0$, $\Psi(r)=\phi$ (empty set), $\Phi(r)=\phi$, for $r\in R_2$;

2) Put each operation of the new order that has no preceding operation into the set $\Psi(r)$, where r is the corresponding resource. Set the potential start time of these operations to be the order arrival time;

3) For each set $\Psi(r)$, select its highest priority operation and put it in the first position.

Step 2: For each $r\in R_2$, let i denote the first element in $\Psi(r)$ if $|\Psi(r)|>0$. This step aims to find the first feasible 'slot' for operation i, based upon the information in $\Lambda(r(i))$, starting from the potential start time s_i. The procedure may be described as follows. Suppose $\Lambda(r(i))=\{o(1), \ldots, o(j)\}$ and $s_i<c_{o(j)}$ (if $s_i\geq c_{o(j)}$ go to Step 3), where $o(j)$ is the jth operation in $\Lambda(r(i))$. Then:

1) If $s_{o(1)} - s_i \geq x_i$, then a slot available for operation i has been found, go to Step 3;

2) Otherwise, if $s_{o(u)} - c_{o(u-1)} < x_i$ for $u=1, 2, \ldots, v-1$ and $s_{o(v)} - c_{o(v-1)} \geq x_i$, then the first feasible slot is located between the operations $o(v-1)$ and $o(v)$. Reset $s_i=c_{o(v-1)}$.

Step 3: Use a priority rule to select the highest priority operation within the first element in $\Psi(r)$ for $r\in R_2$ and denote it i.

Step 4: Consider the current operation i, which was selected by Step 3:

1) Set the operation completion time $c_i = s_i + x_i$;

2) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

3) Set $k = k+1$.

Step 5: Consider the operations in $\Psi(r(i))$. Update their potential start times by $s_j = max(s_j, c_i)$, for $j\in\Psi(r(i))$. Use priority rules to select the highest priority operation in $\Psi(r(i))$ and put it in the first position of the set.

Step 6: Consider the immediately successive operation of i, denoted by l. If all subassemblies of operation l have been planned, i.e. for any $j\in C_l$, $j\in\Phi(r(j))$, then:

1) Append the operation l to the set $\Psi(r(l))$ and set $s_l=max\{ c_j : j\in C_l\}$;

2) Use a priority rule to select the highest priority operation in $\Psi(r(l))$ and then put it in the first position of the set.

Step 7: If $k<|\Gamma_2|$, go to Step 2.

Step 8: Stop.

Forward scheduling dispatches the operations as soon as possible. The resultant plan **s** is exactly the same as **a** (i.e. **a**=**s**), where **a** is a vector of the actual processing start times, a_i ($i\in\Gamma_2$). Step 1 satisfies the constraint given in equation (4); step 2 satisfies the constraint given in (5); step 4 implies equation (3); whilst steps 5 and 6 meet the constraint given in equation (2). However, the resultant plan may not be good for several reasons. Firstly, since the plan is designed starting from the new order arrival time, the final products within the new order may be finished much earlier than the due date if the workload is light. Secondly, dispatching jobs as soon as possible may incur holding cost at downstream stages, where the holding cost is higher because value and time have been added to the item. The first disadvantage can be partially overcome by introducing an iterative procedure that advances the planned start time. An optimal initial start time can then be found that reaches an appropriate trade-off between earliness and tardiness costs.

The computational complexity of the FIP algorithm will now be considered. Let n_1 denote the total number of operations in the existing orders and n_2 denote the number of operations in the new order. The worst-case computation times of step 2 are less than $\sum_{r\in R2}|\Lambda(r)|\leq n_1$. The computation times for step 3 ~ step 5 are less than $4n_2$. The computation times for step 6 is $|C_l|+1+|\Psi(r(l))|\leq n_2$ because $C_l\subseteq\cup_r\Phi(r)$. Step 2 to step 6 is repeated n_2 times. Hence, the computational complexity of forward incremental planning is less than $O(n_2^2+n_1n_2)$.

3.3 Backward incremental planning

The backward incremental planning (BIP) procedure is a 'top down' strategy that starts at the product level and then moves level by level down the product structure. It starts from the last operation within the new order and works backwards from the due date. It finds the first 'feasible' available period of the resource required for the last operation of the new order. To be 'feasible' the available period must be: i) long enough to perform the operation; and ii) the processing must be completed before the due date. The next operation whose successive operation has been planned is selected from those operations associated with the new order according to a priority rule. A feasible available period for this operation should be long enough to perform the operation. Processing must also be finished before any succeeding operation's start time. This procedure continues until all operations of the new order are planned. If BIP cannot produce a schedule that completes all the operations before the due date, it is then necessary to advance the due date.

$\Lambda(r)$, $\Psi(r)$ and $\Phi(r)$ have been defined in the previous section. The BIP procedure is symmetrical to FIP, because BIP loads the operations of the incoming orders backwards from the due dates whilst FIP loads the operations of the incoming orders forwards from the arrival dates.

Algorithm for Backward Incremental Planning(BIP)

Step 1: Initialisation:

1) Set $k = 0$, $\Psi(r)=\phi$ (empty set), $\Phi(r)=\phi$, for $r\in R_2$;

2) Put the last operations of the new order that have no successive operation into the set $\Psi(r)$, where r is the required resource. Set the potential completion times of the last operations to be their product due dates;

3) For each set $\Psi(r)$, select the highest priority operation and put it in the first position.

Step 2: For each $r\in R_2$, let i denote the first element in $\Psi(r)$ where $|\Psi(r)|>0$. This step aims to find the first feasible 'slot' for the operation i based on the information in $\Lambda(r(i))$ starting backwards from the potential completion time c_i. The procedure may be described as follows. Suppose $\Lambda(r(i))=\{o(1), \ldots, o(j)\}$ and $c_i>s_{o(1)}$ (if $c_i\leq s_{o(1)}$ go to Step 3):

1) If $c_i - s_{o(j)} \geq x_i$, then the slot available for operation i has been found, go to Step 3;

2) Otherwise, if $s_{o(u)} - c_{o(u-1)} < x_i$ for $u=j, j-1, \ldots, v$ and $s_{o(v)} - c_{o(v-1)} \geq x_i$, then the first feasible slot is located between the operations $o(v-1)$ and $o(v)$. Reset $c_i=s_{o(v)}$.

Step 3: Among the operations composed of the first element in $\Psi(r)$ for $r\in R_2$, use a priority rule to select the highest priority operation and denote it i.

Step 4: Consider the current operation i:

1) Set operation start time $s_i = c_i - x_i$;

2) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

3) Set $k = k+1$.

Step 5: Consider the operations in $\Psi(r(i))$. Update their potential completion times by $c_j = \min(c_j, s_i)$, for $j\in\Psi(r(i))$.

Step 6: Consider the subassemblies of operation i. For each $j \in C_i$:

1) If $|\Phi(r(j))| = 0$, then $c_j = s_i$;

2) If $|\Phi(r(j))| \neq 0$, let z denote the last operation in $\Phi(r(j))$, then $c_j = \min(s_i, s_z)$;

3) Append the operation j to the set $\Psi(r(j))$.

Step 7: Consider the set $\Psi(r)$ whose operation list is changed. That is, for each $r \in \{r(i)\} \cup \{r(j) \mid j \in C_i\}$, select the highest priority operation in $\Psi(r)$ using a priority rule and put it in the first position of the set.

Step 8: If $k < |\Gamma_2|$, go to Step 2.

Step 9: Stop.

With the backward incremental planning procedure, each operation within the new order is scheduled as late as possible, but no later than the due date. Each operation is scheduled at an earlier time if the resource available period is not long enough. In common with the forward incremental planning, step 2 satisfies the constraint given in equation (5); step 4 implies condition in equation (3); steps 5 and 6 meet the constraint in equation (2). If the resultant plan meets the constraint given in equation (4), then it is feasible and **a**=**s**. Otherwise, the product due date in the above procedure should be replaced by a later time (i.e. planned product due date or completion time). In this case the planned product due date is advanced by the planner who would need to consider resource availability and customer requirements. An outer loop is required to find the appropriate planned product due date and the feasible incremental plan, which can be done by gradually increasing the planned product due date until it yields a feasible incremental plan.

The computational complexity of the FIP algorithm will now be considered. Let n_1 denote the total number of operations in the existing orders and n_2 denote the number of operations in the new order. The worst-case computation times of step 2 are less than $\sum_{r \in R2} |\Lambda(r)| \leq n_1$. The computation times for step 3 ~ step 6 are less than $4n_2$. The computation times for step 7 is $\sum_{j \in Ci \cup \{i\}} |\Psi(r(j))| \leq n_2$. From step 2 to step 7 is repeated n_2 times. Hence, the computational complexity of backward incremental planning is less than $O(n_2^2 + n_1 n_2)$. The BIP and FIP have the same complexity because they are symmetrical.

The forwards and backwards incremental planning algorithms only provide a procedure to deal with a single new order (or a group of new orders). If a series of new orders arrive at different times, an outer loop should be added. That is, before the first step, the information on order arrival times and due dates needs to be inserted; before the last step, an extra step is inserted which puts each operation $i \in \Phi(r)$ into $\Lambda(r(i))$ for $r \in R_2$ according to the earliest start time first.

With both heuristic algorithms, the operations are loaded order by order. $\Psi(r)$ and $\Phi(r)$ only consider the operations in the new order that are performed on the same resource. The information about existing orders on this resource is given in $\Lambda(r)$, which remains unchanged during planning a new order. The main advantage of heuristics is their simplicity and their similarity to the manual loading method used with Gantt Charts, which makes them easy for planners to understand.

3.4 Evolution Strategy incremental planning

For the heuristic methods in the above sections, it is not clear how close the results are to the optimum. This section applies the Evolution Strategy method to find a global optimal or near-optimal solution. Evolution Strategy (ES) is a stochastic search optimisation method that is based upon the principle of biological evolution. It is similar to Genetic Algorithms (GA) and also includes an iterative procedure that contains 'selection', 'recombination/crossover' and 'mutation' operators. GAs use either binary, string or real coding representations of the object variables and are suitable for combinatorial problems. ES uses continuous variables (i.e. floating-point variables) to represent a solution and is thus more straightforward for numerical optimisation problems (Schwefel 1995, Schwefel and Back 1998).

The scheduling problem (1) is a numerical optimisation problem on timings (i.e. the vector $\boldsymbol{s}$). The use of ES generates the timings of operations directly and evaluates their costs in a random search. The problem with doing this is that the timings that are generated may not correspond to a feasible schedule. When the schedule is implemented it may not be possible to keep exactly to the timings, i.e. $\boldsymbol{a} \neq \boldsymbol{s}$. Thus evaluating the cost of a generated schedule is not a trivial problem, since the cost function (1) is explicitly expressed in terms of $\{a_i\}$ and $\{c_i\}$. It is necessary to evaluate the cost of some implementation of the schedule, bearing in mind that the mode of implementation chosen may not be the best for that schedule. This section illustrates the outline of Evolution Strategy method first and then describes a detailed procedure to evaluate the timings.

Applying the Evolutionary Strategy to the above scheduling problem is straightforward. The chromosome of an individual schedule is represented by the vector $\boldsymbol{s} = \{s_i,\ i \in \Gamma_2\}$. Figure 1 illustrates the structure of the algorithm.

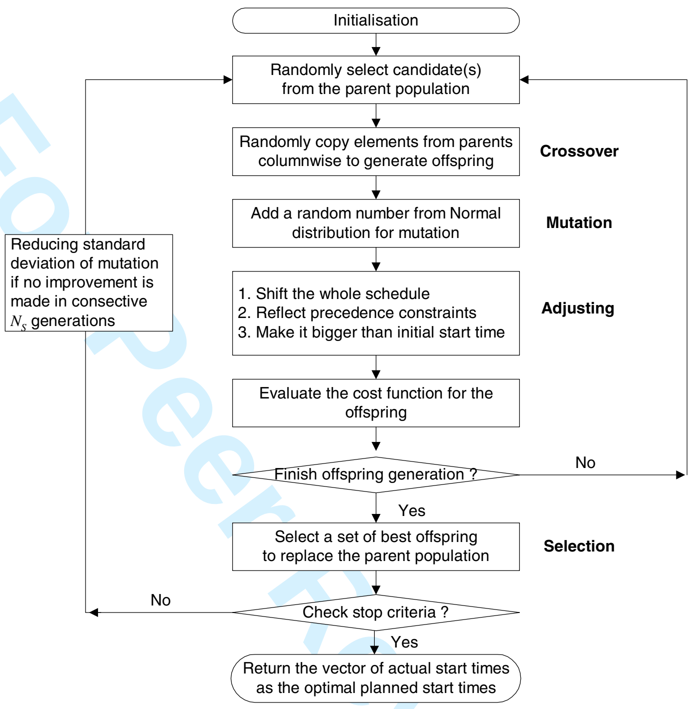
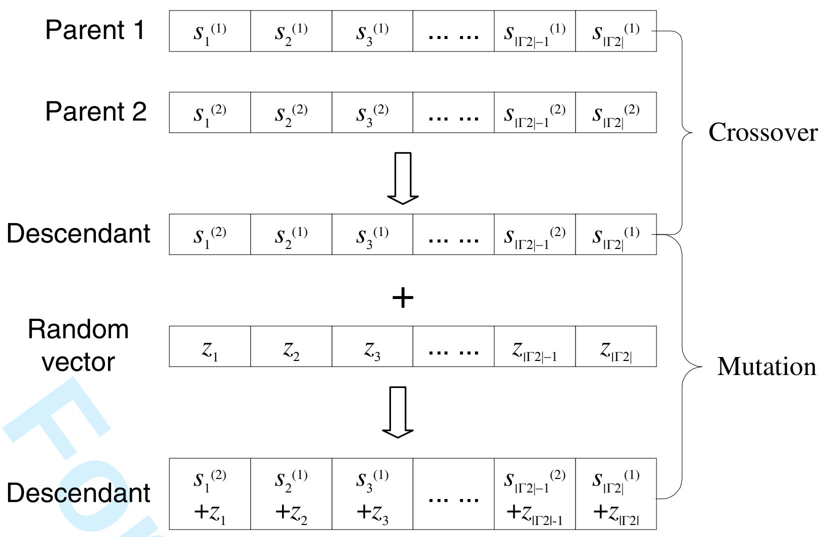

Initialisation
Randomly select candidate(s) from the parent population
Randomly copy elements from parents columnwise to generate offspring
Crossover
Add a random number from Normal distribution for mutation
Mutation
1. Shift the whole schedule
2. Reflect precedence constraints
3. Make it bigger than initial start time
Adjusting
Evaluate the cost function for the offspring
Finish offspring generation ?
No
Yes
Select a set of best offspring to replace the parent population
Selection
Check stop criteria ?
No
Yes
Return the vector of actual start times as the optimal planned start times
Reducing standard deviation of mutation if no improvement is made in consective N_S generations

Figure 1 The Evolution Strategy scheduling method

Each chromosome in the offspring generation is created by crossover on two selected parents. Crossover randomly chooses the starting times of two operations in two parents' schedules and interchanges them, that is, each gene of the descendant is randomly copied from the corresponding gene in one of its parents (see figure 2). Mutation adds a random number from a Normal distribution $N(0, \sigma)$ to all the starting times in a schedule. The standard deviation σ represents the degree of mutation allowed and is reduced by a factor α if there has been no improvement in N_S generations. The crossover and mutation are illustrated in figure 2. An initial value σ_0 is specified. The sizes of the parent and offspring populations are specified as P_P and P_O, respectively. If $P_O > P_P$ then the best P_P chromosomes are selected from the offspring population to form the parents of the next generation.


Parent 1
Parent 2
Descendant
Random vector
Descendant
Crossover
Mutation
+
$s_1^{(1)}$ $s_2^{(1)}$ $s_3^{(1)}$ …… $s_{|\Gamma 2|-1}^{(1)}$ $s_{|\Gamma 2|}^{(1)}$
$s_1^{(2)}$ $s_2^{(2)}$ $s_3^{(2)}$ …… $s_{|\Gamma 2|-1}^{(2)}$ $s_{|\Gamma 2|}^{(2)}$
$s_1^{(2)}$ $s_2^{(1)}$ $s_3^{(1)}$ …… $s_{|\Gamma 2|-1}^{(2)}$ $s_{|\Gamma 2|}^{(1)}$
z_1 z_2 z_3 …… $z_{|\Gamma 2|-1}$ $z_{|\Gamma 2|}$
$s_1^{(2)}+z_1$ $s_2^{(1)}+z_2$ $s_3^{(1)}+z_3$ …… $s_{|\Gamma 2|-1}^{(2)}+z_{|\Gamma 2|-1}$ $s_{|\Gamma 2|}^{(1)}+z_{|\Gamma 2|}$

Where $s_i^{(1)}$ denotes the planned start time of operation i in the parent 1, and z_i is a random number generated for operation i to perform mutation

Figure 2 Crossover and mutation in Evolution Strategy method

After all the offspring have been generated, a ‘repair’ process adjusts infeasible schedules to make them feasible (where feasible means that a solution can be evaluated). Three adjustment operations are performed. Firstly, the whole schedule is shifted by a random time. This is optional but it has been found that it can increase the search speed significantly if the initial solution is far away from the optimum, e.g. if the initial solution was created by MRP backwards scheduling with infinite capacity. Secondly, the planned start times for each operation are adjusted so that they are larger than preceding operations’ planned start times. This reflects precedence constraints. However, this step is also optional since the execution of the evaluation procedure (given below) implies precedence constraints. However, experiments suggest that this step is helpful if the initial solution is not too bad, but may reduce the search speed if the initial solution is far away from the optimum. This can be partially explained by the fact that a poor s_i may result in unsatisfactory planned start times for all those successive operations due to this adjustment. Thirdly, planned start times are adjusted to be larger than the order arrival time.

Evaluating the adjusted schedule is not straightforward. Every operation in the new order can be processed exactly at the planned start times $\{s_i\}$ if there is no contention at resources during its processing duration. In this situation, $a_i \equiv s_i$. However, when there is a queue of operations at a resource then it is necessary to decide which operation to start first. A commonly used rule is to select the operation in the queue with the earliest planned start time (EPST) first. That is, the operation sequence is implied by the planned start times. The detailed procedure for evaluating the cost function associated with a given incremental plan $\{s_i,\ i \in \Gamma_2\}$ without affecting the processing of the existing orders is given as follows. $\Lambda(r)$ was defined in section 2.2, $\Psi(r)$ denotes the 'ready-to-go' operation set (which consists of the operations whose preceding operations have been completed, whose planned start times are earlier than the current time, and whose resource is r), and $\Phi(r)$ denotes the finished operation set on resource r.

An Evaluation Procedure for Evolution Strategy Incremental Planning

Step 1. Initialisation:

1) Set k=0, $\Psi(r)=\phi$ (empty set), $\Phi(r)=\phi$, for $r \in R_2$;
2) Put each operation of the new order that has no preceding operation into the set $\Psi(r)$, where r is the corresponding resource. Set $a_i = s_i$, for $i \in \Gamma_2$;
3) For each $\Psi(r)$, find the operation using the Earliest Planned Start Time (EPST) priority rule and put it in the first position. If $|\Lambda(r)|>0$, let j denote the first operation in Ψ(r) and find the first feasible slot for operation j starting from a_i. Update the potential start time a_i;
4) Set each resource busy/idle flag to be idle for $r \in R_2$.

Step 2: Find the next event among the operations within the first element in $\Psi(r)$ for $r \in R_2$, find the operation that has the Earliest Completion Time (ECT) and denote it by i. Set $c_i = a_i + x_i$.

Step 3: Set resource busy/idle flag. For each $r \in R_2$, if the first operation in $\Psi(r)$ has an earlier start time than c_i, set resource r's busy/idle flag to be busy.

Step 4: Check $\rho(i)$ (i.e. the parent of operation i) is ready or not. If for any $j \in C_{\rho(i)}$, $j \in \Phi(r(j))$, that mean all subassemblies of $\rho(i)$ have already been completed and therefore $\rho(i)$ is ready, go to Step 5; otherwise, $\rho(i)$ is not ready, go to Step 6.

Step 5: Operation $\rho(i)$ is ready. Set $a_{\rho(i)} = max\{\ c_j : j \in C_{\rho(i)}\}$.

If $\rho(i)$ uses the same resource as operation i, then do the following five steps:

(1) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

(2) Append the operation $\rho(i)$ to $\Psi(r(i))$;

(3) Select the EPST operation in $\Psi(r(i))$ and put it in the first position;

(4) Let j denote the first operation in $\Psi(r(i))$ and set $a_j = \max(a_j, c_i)$;

(5) Find the first feasible slot for operation j starting from a_j and update the potential start time a_j.

Otherwise, do the following six steps:

(1) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

(2) Select the EPST operation in $\Psi(r(i))$ and put it in the first position;

(3) Let j denote the first operation in $\Psi(r(i))$ and set $a_j = max(a_j, c_i);$

(4) Find the first feasible slot for operation j starting from a_j and update the potential start time a_j;

(5) Append the operation $\rho(i)$ *to* $\Psi(r(\rho(i)));$

(6) If the busy/idle flag of resource $r(\rho(i))$ is idle, then select the EPST operation in $\Psi(r(\rho(i)))$ and put it in the first position; let j denote the first operation in $\Psi(r(\rho(i)))$ and set $a_j = \max(a_j, c_l : l \in \Phi(r(j)));$ find the first feasible slot for operation j starting from a_j and update the potential start time a_j.

Step 6: Operation $\rho(i)$ is not ready:

1) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

2) Select the EPST operation in $\Psi(r(i))$ and put it in the first position;

3) Let j denote the first operation in $\Psi(r(i))$ and set $a_j = \max(a_j, c_i)$;

4) Find the first feasible slot for operation j starting from a_j and update the potential start time a_j.

Step 7: Set busy/idle flag to be idle for each $r \in R_2$. Consider the operations in $\Psi(r(i))$. Update their potential start times by $s_j = max(s_j, c_i)$, for $j \in \Psi(r(i))$. Using EPST priority rules to select the highest priority operation in $\Psi(r(i))$ and put it in the first position of the set.

Step 8: Set $k = k+1$. *If* $k \le |\Gamma_2|$, go to Step 2.

Step 9: Stop.

In the above procedure, the sub-procedures to find a feasible slot for an operation are the same as those described in the forward incremental planning algorithm (section 2.2). It should be pointed out that the above procedure guarantees that any incremental schedule defined by $\{s_i, i \in \Gamma_2\}$ can be applied and generates the actual operation start and completion times $\{a_i, c_i, i \in \Gamma_2\}$ without changing the processing of the existing orders. The cost corresponding of this schedule can then be evaluated by (1). The evolution strategy procedure is terminated if there is no improvement within a total of N_G consecutive generations or the total number of generations reaches a specified limit N.

4. Regenerative planning

In many cases, it is not appropriate to schedule a new order using incremental planning methods. For example, if the new order is urgent (e.g. if it has a tight delivery date with a heavy tardiness penalty) or large, it is often not sensible or feasible to leave plans for existing orders unchanged. In these situations, it is necessary not only to generate a schedule for the new order but also to regenerate a schedule for the existing orders. For simplicity, the new arriving orders are assumed to have the same priority as existing orders. Their priority can actually be reflected by their due dates and penalty cost coefficients.

4.1 Problem formulation

To formulate the rescheduling problem, some new notation will be used. Let $\Gamma=\Gamma_1\cup\Gamma_2$, $R=R_1\cup R_2$ and $\Gamma'=\{i\in\Gamma_1 \mid a_i \geq arrivalTime\}\cup\Gamma_2$. The set Γ' is a subset of Γ since some operations in the existing orders have started processing before the new order arrives. Without loss of generality, assuming that no product in the existing orders has been finished before the new order arrives (otherwise, simply delete the operations of the finished product from Γ_1 and L_1). Let $L=L_1\cup L_2$.

Regenerative planning aims to find an optimal schedule for all operations in Γ' by minimising the following cost function:

$$J(s) = \textstyle\sum_{i\in\Gamma\setminus L} h_i(a_{\rho(i)} - c_i) + \sum_{i\in L} h_i max(d_i - c_i, 0) + \sum_{i\in L} h_i^- max(c_i - d_i, 0) \qquad (6)$$

s.t. for any $i\in\Gamma'$

$$a_i = max(s_i, c_{\varphi(i)}, \{c_j \mid j\in C_i\}) \qquad (7)$$

$$c_i = a_i + x_i \qquad (8)$$

$$a_i \geq arrivalTime \qquad (9)$$

$$a_i \geq c_j \text{ for any } j\in\{j\in\Gamma_1 \mid a_j < \text{arrivalTime and } r(j) = r(i)\} \qquad (10)$$

It should be pointed out that in (6) Γ is used rather than Γ'. This is because rescheduling could incur extra holding costs for work-in-progress that has already been started or finished. These extra costs must be taken into account in regenerative planning. In addition, the set L is used in (6) rather than L_2 in (1) because products in both existing orders and new orders will be affected by regenerative planning. Equations (7)~(9) are similar to (2)~(4) except that operation i belongs to Γ'. Inequality (10) describes the extra constraints caused by those operations in existing orders that have already started but not finished.

To implement regenerative planning, operations in the existing orders and in the new order will be treated with the same priority. This is similar to the vertical finite loading scheme (Vollmann et al. 1992) and operation-oriented heuristics (Yeh 1997). Vertical loading or operation-oriented heuristics simply selects the next operation from a set of operations waiting in the queue according to a priority rule and then loads it on the machine.

Two methods are presented to deal with this rescheduling problem, which are described in the following two sections. The first is based upon forward finite loading and priority rules (e.g. first come first served). The second is based upon Evolution Strategy, which aims to find an approximate solution to the global optimum by randomly searching the solution space. The definitions are similar to those in section 2.2.

The set of operations that belong to the existing orders and have started on the resource r before the new order arrival time are denoted $\Lambda(r)$. This set is sequenced with the earliest start time first. R' is the set of resources (machines) used by the operations in the set Γ'.

4.2 Forward regenerative planning

The procedure for forward regenerative planning is similar to forward incremental planning. However, in the regenerative planning problem, the operation set that is required to be scheduled or rescheduled is composed of two parts, the operations in the new order and the operations in the existing orders that have been planned but not started, i.e. Γ' defined in the section above.

There are three differences between regenerative planning and incremental planning. Firstly, the initial ready-to-plan operation set consists of operations in the new order that have no preceding operation and operations in the existing orders that are ready to reschedule. Here a 'ready to reschedule' operation means that an operation needs to be rescheduled, together with all of the operations associated with the item's subassemblies (if they exist). These operations may have already been started before the new order arrives. To set the earliest start times for operations in the initial ready-to-plan operation set, it is necessary to take into account operations in the existing orders that have started, but not finished when the new order arrives. Secondly, incremental planning requires an additional step, i.e. to find a feasible capacity slot for each operation. Thirdly, since the current operation may relate to an existing order, its immediate successor operation is ready if, and only if, all its subassemblies have been rescheduled, or have started before the new order arrives. The details of the forward regeneration planning procedure are given as follows.

Algorithm for Forward Regeneration Planning (FRP)

Step 1: Initialisation:

1) Set $k = 0$, $\Psi(r)=\phi$ (empty set), $\Phi(r)=\phi$, for $r\in R'$;

2) Put the initial ready-to-plan operations into $\Psi(r)$, where r is the corresponding resource;

3) Set the earliest start times of these operations to be the order arrival time if the corresponding resources are idle. Otherwise, set the earliest start time to be the completion time of the current operation processing on the resource or its subassembly;

4) For each $\Psi(r)$ $(r\in R')$, select its highest priority operation and put it in the first position.

Step 2: Find the next operation within the operations in the first element in $\Psi(r)$ for $r\in R'$, using a priority rules to select the highest priority operation which is denoted as i.

Step 3: Consider the current operation i:

1) Set operation completion time $c_i = s_i + x_i$;

2) Delete the operation i from $\Psi(r(i))$ and append it to $\Phi(r(i))$;

3) Set $k = k+1$.

Step 4: Consider the operations in $\Psi(r(i))$. Update their earliest start times by $s_j = max(s_j, c_i)$, for $j\in \Psi(r(i))$. Use a priority rule to select the highest priority operation in $\Psi(r(i))$ and put it in the first position of the set.

Step 5: Consider the immediately successive operation of i, denoted by l. If $j\in\Phi(r(j)) \cup \Lambda(r(j))$ for any $j\in C_l$, then:

1) Append the operation l to the set $\Psi(r(l))$ and set $s_l = max\{c_j : j\in C_l\}$;

2) Use a priority rule to select the highest priority operation in $\Psi(r(l))$ and put it in the first position of the set.

Step 6: If $k<|\Gamma|$, go to Step 2.

Step 7: Stop.

The key differences between FRP with FIP are: i) the operation set and the resource set are different, i.e. Γ' and R' in *FRP* compared with Γ_2 and R_2 in FIP. This causes the difference in step 1; ii) Step 2 in FIP is additional, which aims to find the resource availability slot; iii) Step 6 in FIP and step 5 in FRP are different, which is used to judge whether the immediately successive operation is ready to plan.

The forward regenerative planning procedure could produce a schedule in which the new orders for final products may be finished significantly earlier than the due date. This might happen if the due date was set sufficiently late and if the workload is light. This is similar to forward incremental planning. An optimal initial start time can be found that achieved the best trade-off between holding costs and tardiness cost.

The computational complexity of the algorithm will now be considered. Let n_{11} denote the number of regenerative planning operations in the existing orders and n_{12} equal the number of operations that do not need to be rescheduled. Let n_2 denote the number of operations in the new order. The worst-case computation times for step 2 are less than $\sum_{r\in R}|\Psi(r)| \le n_2+n_{11}$. The computation times for step 3 ~ step 4 are less than $2(n_2+n_{11})$. The computation times for step 5 is $|C_l| + |\Phi(r(j))| + |\Lambda(r(j))| + |\Psi(r(l))| \le 2(n_2+n_{11}) + n_{12}$ because $C_l \subseteq \cup_r \Phi(r)$. Steps 2 though to 5 are repeated n_2+n_{11} times. Hence, the computational complexity of the forward regeneration planning is less than $O((n_2+n_{11})^2 + n_{12}(n_2+n_{11}))$. The computational complexity mainly depends upon $n_2 + n_{11}$.

4.3 Evolution Strategy regenerative planning

In order to apply the Evolution Strategy (ES) method to perform regenerative planning, a modified evaluation procedure should be executed to obtain $\{a_i, c_i, i\in\Gamma'\}$ and to evaluate the cost function (given in equation 6). The decision variables that need to be optimised are described by $\{s_i, i\in\Gamma'\}$. The main differences between Evolution Strategy regenerative planning (ESRP) and incremental planning (ESIP) are: i) the initial ready-to-go operation set $\Psi(r)$; ii) the approach for finding resource availability slots; iii) the methods of checking whether an operation's successor is ready to go; and iv) the procedure for setting an operation's potential start time. The evaluation procedure for executing a regenerated plan ***s*** is broadly similar to the procedure used to execute an incremental plan (see section 2.4) and therefore details are omitted. However, specific differences are: i) the operation and resource sets are different, i.e. using Γ' and R' in ESRP instead of Γ_2 and R_2 in ESIP. Therefore, the step 1 in ESIP should include some operations in the existing order; ii) in step 4 of ESIP, $\Phi(r(j))$ should be replaced by $\Phi(r(j)) \cup \Lambda(r(j))$; and iii) in steps 5 and 6 of ESIP, the procedure to find a resource availability slot is removed. The next section applies and tests the ES regenerative planning approach using some case studies.

5. Case studies

In this section, three incremental planning methods (forward incremental planning (FIP), backward incremental planning (BIP) and Evolution Strategy Incremental Planning (ESIP)) and two regenerative planning methods (forward regeneration planning (FRP) and Evolution Strategy regeneration planning (ESRP)), are applied to three case studies that utilise data obtained from a collaborating capital goods company (Song 2001). The characteristics of these scheduling problems are summarised in table 1. The existing order is one of the Company's major engineered-to-order products, which are normally planned well in advance. It represents a large workload and has a complex product structure (see figure 3). Three cases are considered: i) a small size problem with total word load 45 days; ii) a medium size problem a with total work load 106 days; and iii) a large size problem with work load 304 days. In these three cases the new orders share several resources with the existing order.

Order	Product code (Song 2001)	Machining/Assembly operations = total	Existing/New Resources = total
Existing order	228	100/13 = 113	13/0=13
New order: case 1	448	8/9 = 17	1/1=2
New order: case 2	252	46/6 = 52	3/2=5
New order: case 3	312	102/7 = 109	2/3=5

Table 1 Characteristics of the case studies

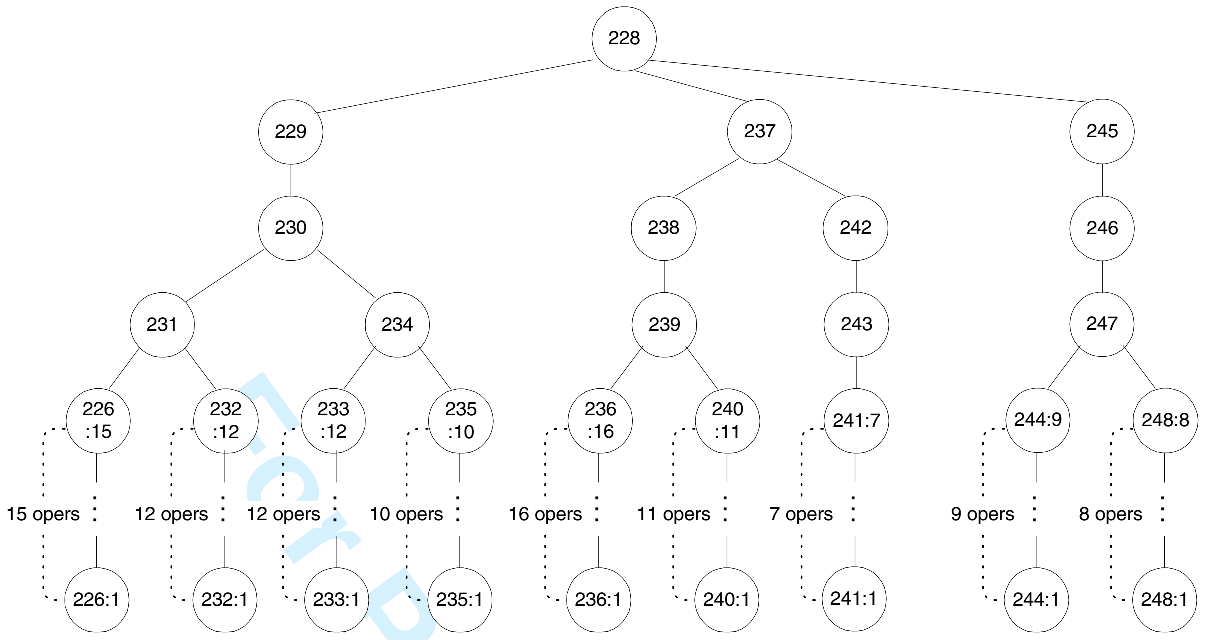


Figure 3 Product structure for the existing product 228 (Song 2001)

In all three cases, the existing order is the same. For simplicity, it is assumed that the existing order is planned using the forward finite loading method and the product is finished exactly at its due date. Its initial start date is 0 and the due date is 144 days. The complexities of the new orders in three cases are different. The cases have a total number of operations of 17, 52 and 109 respectively (i.e. with increasing complexity). The new order may use the existing or new resources, which is described in the last column in table 1.

The work in progress holding cost, product earliness cost, product tardiness cost and total costs are compared. It is assumed that the holding cost coefficient after operation *i* is $h_i = 10 \times$ (sum of all operation times in days already spent on this item). Here the costs use factor 10, which is a rough estimate and is used for illustrative and comparative purposes. Corresponding costs are given in UK pounds. The cumulative increase in holding costs is due to the incremental costs of the operations. It is assumed that the product tardiness penalty coefficient is twice the corresponding product earliness penalty coefficient, i.e. $h_l^- = 2h_l$ for $l \in L$. The costs associated with the existing order are: HC=£1,963,770, PEC=PTC=0, and the total cost is =£1,963,770.

The priority rule used in the heuristic methods is the first-in-first-out/first-come-first-served (FIFO/FCFS) for FIP and FRP, and the last potential completion time last (LCT) for BIP. In each case, two situations are investigated. One has a tight due date for the new order and the other has a relatively loose due date. For the FIP and FRP methods two schedules can be obtained. The first is obtained by starting the incremental planning from the new order arrival time (denoted by FIP and FRP) and the other is the optimal one produced by adjusting the initial start time (denoted by Opt_FIP and Opt_FRP).

The BIP method can yield a feasible schedule if the due date for the new order is loose. Otherwise, it may produce an infeasible solution whose initial start time is earlier than the new order arrival time. However, an optimal feasible solution for BIP can be obtained by adjusting the planned due date (i.e. product completion time) as mentioned in section 2.3. In all three cases that used the BIP method only the optimal feasible schedule is given (denoted by Opt_BIP), since the initial BIP schedule may have been infeasible.

For both the ESIP and ESRP methods, the initial parameters for the Evolution Strategy algorithm were P_P=20, P_O/N=200/200, σ_0=10.00, α=0.90, N_S=10 and N_G=50 for all three examples. These values for the initial parameters were based upon the preliminary results of the case studies presented by Song (2001), which showed that they are appropriate for similar sized scheduling problems.

Example 1. The new order is small in terms of total workload, with a lead time of 17 days and the arrival date is 110 days. Two due dates are considered 140 days (which is tight) and 160 days (which is loose). The cost of the new order for these two situations under FIP, Opt_FIP, Opt_BIP and ESIP planning methods are given in table 2. The table indicates the holding costs (HC), the product earliness costs (PEC), the product tardiness costs (PEC) and total costs for both the existing orders and the new order.

For the tight due date situation (d=140) in table 2, the new order is finished later than the due date and there is no product earliness cost (PEC=0). In terms of total cost, the order of the methods from the best to the worst is: ESIP, Opt_BIP, Opt_FIP and FIP. For the loose due date situation (d=160) in table 2, the new order can be finished before the due date and there is no product tardiness cost (PTC=0). The best to the worst cases in order are: ESIP, Opt_FIP, Opt_BIP and FIP. FIP incurs product earliness cost due to the fact that it starts processing at the new order arrival time and the due date is relatively loose.

		New order				All orders
Due date (days)	Method	HC (£k)	PEC (£k)	PTC (£k)	Total (£k)	Total (£k)
d=140 (tight)	FIP	56.15	0.00	144.80	200.95	2164.72
	Opt_FIP	42.49	0.00	144.80	187.29	2151.06
	Opt_BIP	47.68	0.00	133.73	181.42	2145.19
	ESIP	44.78	0.00	122.51	**167.29**	**2131.06**
d=160 (loose)	FIP	56.15	16.76	0.00	72.91	2036.68
	Opt_FIP	43.21	0.10	0.00	43.31	2007.08
	Opt_BIP	46.44	0.00	0.00	46.44	2010.21
	ESIP	42.55	0.02	0.00	**42.57**	**2006.34**

Table 2 Costs for incremental planning methods in example 1

Regenerative planning will now be considered for the same example. Again the arrival date for the new order is 110 days. The total number of regenerative planning operations is 14 (for the existing order) + 17 (for the new order) giving a total of 31. Two situations are considered with due dates of 140 days and 160 days respectively for the new order. The costs for the existing and new orders for these two situations under FRP, Opt_FRP and ESRP planning methods are given in table 3, which shows that ESRP is the best and FRP is the worst for both the tight and loose due date situations. For the tight due date situation (*d*=140), the new order is finished later than the due date and there is no product earliness cost (PEC=0). Intuitively, it is preferable to start processing as soon as possible in tight due date situations. However, the results show that Opt_FRP is better than FRP. It can be explained by the fact that the existing order has a tight due date and a higher tardiness penalty. Starting the new order too early may cause competition for common resources with the existing order. For the loose due date situation (*d*=160), both the existing order and the new order can be finished on time if they are properly rescheduled (see the result of ESRP in table 3). However, the FRP and Opt_FRP methods incur tardiness costs. This is because the FIFO rule cannot guarantee the existing order will finish on time even if the new order has a loose due date.

Due date (days)	Method	HC (£K)	PEC (£K)	PTC (£K)	Total (£K)
D=140 (tight)	FRP	2125.01	0.00	350.86	2475.87
	Opt_FRP	2045.62	0.00	212.44	2258.06
	ESRP	2005.18	0.00	122.51	**2127.69**
D=160 (loose)	FRP	2125.01	22.31	217.16	2364.48
	Opt_FRP	2007.26	0.00	9.61	2016.87
	ESRP	2003.10	0.01	0.00	**2003.11**

Table 3 Costs for regenerative planning methods in example 1

Example 2. The new order is of medium size in terms of work content, with a lead-time of 18 days and the arrival date is 50 days. Two situations are considered with due dates of 120 days and 150 days respectively for the new order. The costs of the new order for these two situations under FIP, Opt_FIP, Opt_BIP and ESIP planning methods are given in table 4.

		New order				All orders
Due date (days)	Method	HC (£k)	PEC (£k)	PTC (£k)	Total (£k)	Total (£k)
d=120 (tight)	FIP	559.10	0.00	582.81	1141.91	3105.68
	Opt_FIP	400.70	0.00	635.81	1036.51	3000.28
	Opt_BIP	315.57	0.00	610.56	926.13	2889.90
	ESIP	309.36	0.00	609.31	**918.67**	**2882.44**
d=150 (loose)	FIP	559.10	26.60	0.00	585.69	2549.46
	Opt_FIP	400.70	0.10	0.00	400.80	2364.57
	Opt_BIP	314.31	0.00	0.00	314.31	2278.08
	ESIP	307.92	0.00	0.03	**307.95**	**2271.72**

Table 4 Costs for incremental planning methods in example 2

For the tight due date situation shown in table 4, the results have the same pattern as those in table 2. For both the tight and loose due date situations, the order of the methods from the best to the worst is: ESIP, Opt_BIP, Opt_FIP and FIP.

Consider regenerative planning for this example. The total number of regenerative planning operations is 40(for the existing order) + 52(for the new order) = 92 in total. The costs of the existing and new orders for these two situations under the FRP, Opt_FRP and ESRP planning methods are given in table 5.

Due date (days)	Method	HC (£k)	PEC (£k)	PTC (£k)	Total (£k)
d=120 (tight)	FRP	2429.91	0.00	351.29	2781.20
	Opt_FRP	2429.91	0.00	351.29	2781.20
	ESRP	2173.00	0.00	324.79	**2497.79**
d=150 (loose)	FRP	2429.91	214.65	144.59	2789.15
	Opt_FRP	2437.49	119.25	145.11	2701.85
	ESRP	2144.67	0.01	72.29	**2216.97**

Table 5 Costs for regenerative planning methods in example 2

Table 5 shows that ESRP is significantly better than FRP and Opt_FRP. For the tight due date situation, FRP is the same as Opt_FRP. For the loose due date situation, all three methods result in product tardiness costs. This is because avoiding tardiness incurs higher holding costs.

Example 3. The new order has a large work content with a lead time of 15 days and an arrival date is 20 days. Two situations are considered with due dates of 220 days and 250 days respectively for the new order. The costs of the new order for these two situations under FIP, Opt_FIP, Opt_BIP and ESIP planning methods are given in table 6.

For both the tight and loose due date situations in table 6, the order of planning methods from the best to the worst is: ESIP, Opt_BIP, Opt_FIP and FIP. The ESIP and Opt_BIP are quite close, but are much better than Opt_FIP and FIP.

		New order				All orders
Due date (days)	Method	HC (£k)	PEC (£k)	PTC (£k)	Total (£k)	Total
d=220 (tight)	FIP	3684.46	0.00	1013.97	4698.42	6662.19
	Opt_FIP	3683.12	0.00	1013.97	4697.09	6660.86
	Opt_BIP	2849.12	0.00	1094.36	3943.48	5907.25
	ESIP	2855.24	0.00	1013.97	**3869.21**	**5832.98**
d=250 (loose)	FIP	3684.46	404.98	0.00	4089.43	6053.20
	Opt_FIP	3667.88	0.51	0.00	3668.39	5632.16
	Opt_BIP	2849.12	0.00	0.00	2849.12	4812.89
	ESIP	2841.44	0.00	0.12	**2841.56**	**4805.33**

Table 6 Costs for incremental planning methods in example 3

Consider the regenerative planning problem for this example. The total number of regenerative planning operations is 75 (for the existing order) + 109 (for the new order) giving a total of 184. The costs of the existing and new orders for these two situations under FRP, Opt_FRP and ESRP planning methods are given in table 7.

Due date (days)	Method	HC (£k)	PEC (£k)	PTC (£k)	Total (£k)
d=220 (tight)	FRP	5541.08	0.00	859.27	6400.35
	Opt_FRP	5529.77	0.00	844.07	6373.84
	ESRP	5019.28	0.00	816.33	**5835.61**
d=250 (loose)	FRP	5541.08	482.33	0.00	6023.41
	Opt_FRP	5645.90	0.00	175.30	5821.20
	ESRP	4763.41	9.82	0.00	**4773.23**

Table 7 Costs for regenerative planning methods in example 3

Table 7 shows that ESRP is significantly better than FRP and Opt_FRP. In the loose due date case, FRP has no tardiness cost but a high product earliness cost. On the other hand, Opt_FRP has zero product earliness cost with a relatively low tardiness cost.

In general, for the incremental planning problem, the following results can be observed from above case studies:

1) From the best to the worst, those methods are ordered by ESIP > Opt_BIP > Opt_FIP > FIP;
2) The Opt_BIP appears to be a good heuristic method, which has a performance close to ESIP, but the Opt_FIP and FIP are much worse than the ESIP;
3) For the tight due date situation, all the methods have zero product earliness cost and their product tardiness costs are not significantly different;
4) For the loose due date situation, product earliness and tardiness costs are equal or close to zero for ESIP, Opt_BIP and Opt_FIP, but the FIP always incurs product earliness cost;
5) Since PEC and PTC are dependent on the due date situations and they are quite similar for those planning methods, reducing the work-in-progress holding cost is critical in the problems under study. That is, a good planning method should be good at reducing holding cost (HC).

An interesting application of incremental planning is to the environments where resource maintenance should be taken into account. During the resource maintenance periods, no operations are allowed. If the resource maintenance is regarded as an existing scheduled order that will be performed on those resources, then the scheduling problem can be represented an incremental planning problem, which aims to fill the operations in those resource non-maintenance periods.

For the regenerative planning problem, the following results can be observed from above case studies:

1) ESRP is the best and FRP is the worst. As the complexity of the new order increases, the benefit of ESRP increases significantly;
2) For the tight due date situation, all the methods have zero product earliness cost and their product tardiness costs are not significantly different;
3) For the loose due date situation the product earliness and tardiness costs are equal or close to zero for ESRP, but Opt_FRP and FRP always incur relatively high product earliness or tardiness cost.

In terms of computational complexity, the heuristic method is substantially better than the Evolution Strategy method. The algorithms were implemented in Tcl-tk and were run on Sun-sparc computer with a Unix OS 5.7 operating system. Heuristic methods (either forward or backward, incremental planning or regenerative planning) take less than two minutes for all examples. However, the Evolution Strategy method takes many hours and ESIP is better than ESRP. For example 3 with tight due date, the ESIP took 27 hours whilst the ESRP took 52 hours. This is in agreement with intuition since incremental planning only schedules the new orders whilst regenerative planning must also reschedule existing orders.

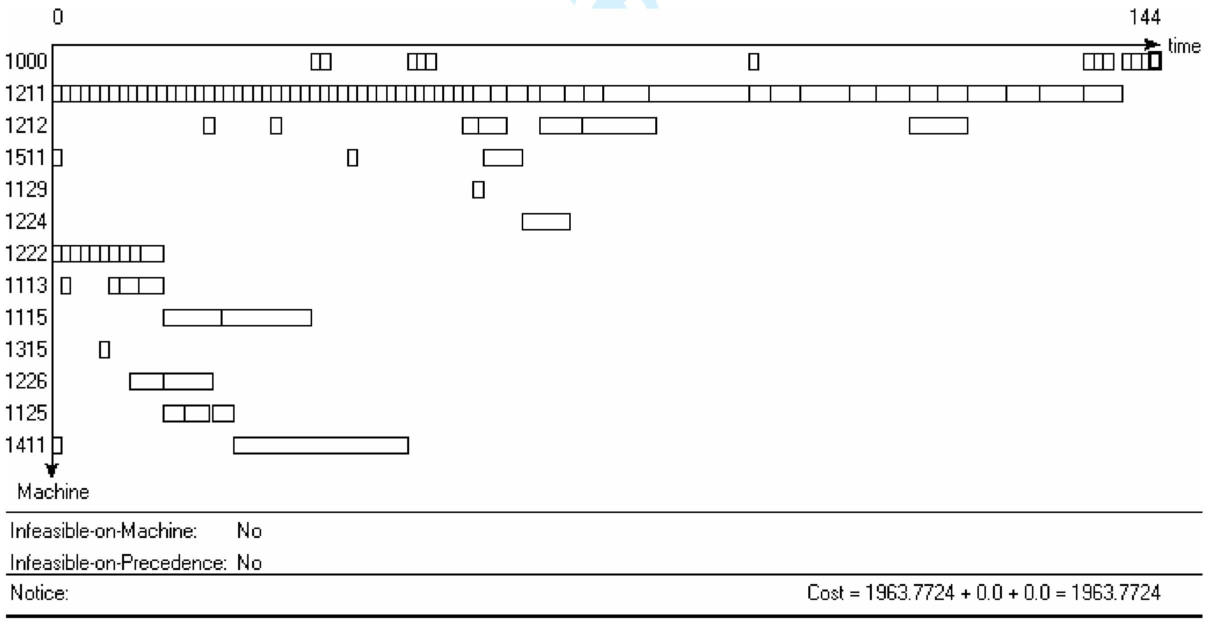
6. Comparison of incremental planning and regenerative planning

This section compares the results of incremental planning and regenerative planning given in the above sections. Only ESIP and ESRP are considered because the Evolution Strategy approach always yields the best result in the above case studies. To make it clear, the total costs for the different examples are summarised in table 8, where the total cost includes the existing and new orders. The total costs for ESIP are obtained by adding 1963.77 (i.e. the total cost of the existing order) to the corresponding values in section 4.

		ESIP		ESRP	
Ex 1	Due date (days)	140	160	140	160
	TotalC (£1,000)	2131.06	2006.34	**2127.69**	**2003.11**
Ex 2	Due date	120	150	120	150
	TotalC (£1,000)	2882.44	2271.72	**2497.79**	**2216.97**
Ex 3	Due date (days)	220	250	220	250
	TotalC (£1,000)	**5832.98**	4805.33	5835.61	**4773.23**

Table 8 Comparison of total costs for ESIP and ESRP in different situations

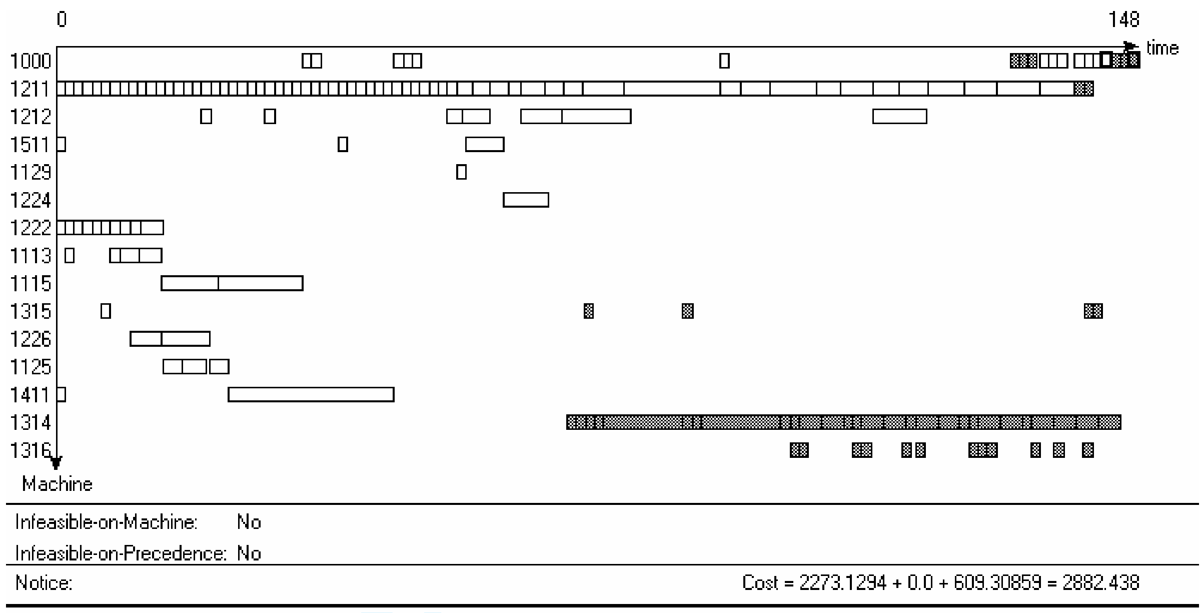
From table 8, it can be seen that ESRP is better than ESIP because ESRP searches a larger solution space. Actually, the solution space for ESIP is a subset of that for ESRP. However, the above table shows that ESIP and ESRP have similar costs in example 1, when the new order is small. This is in agreement with the intuition that if a small order arrives it is preferable to perform net (incremental) planning rather regenerative planning. On the other hand, if the incoming order is large enough, performing regenerative planning is preferred. To illustrate the detailed schedule for a dynamic production system, a Graphical User Interface (GUI) program was developed using the Tcl-tk language. The program can display the operations' durations on each machine as a Gantt Chart. As an example, the Gantt charts of operations for resources for example 2 with *d*=120 days are shown in figures 4, 5 and 6, which correspond to the existing order only, both the existing and new orders under ESIP, and both the existing and new order under ESRP. In figures 4, 5 and 6 the horizontal-axis represents time and the vertical-axis represents the different machines. Each bar or box represents an operation and its duration on the corresponding machine. The second part of the graph indicates the infeasible operations that violate the resource capacity constraints or precedence constraints. The total cost consists of three parts (HC, PET and PTC) and is shown in the right-bottom corner of the figures.



Infeasible-on-Machine: No
Infeasible-on-Precedence: No
Notice: Cost = 1963.7724 + 0.0 + 0.0 = 1963.7724

Note: the operation bar with bold frame represents the final product.

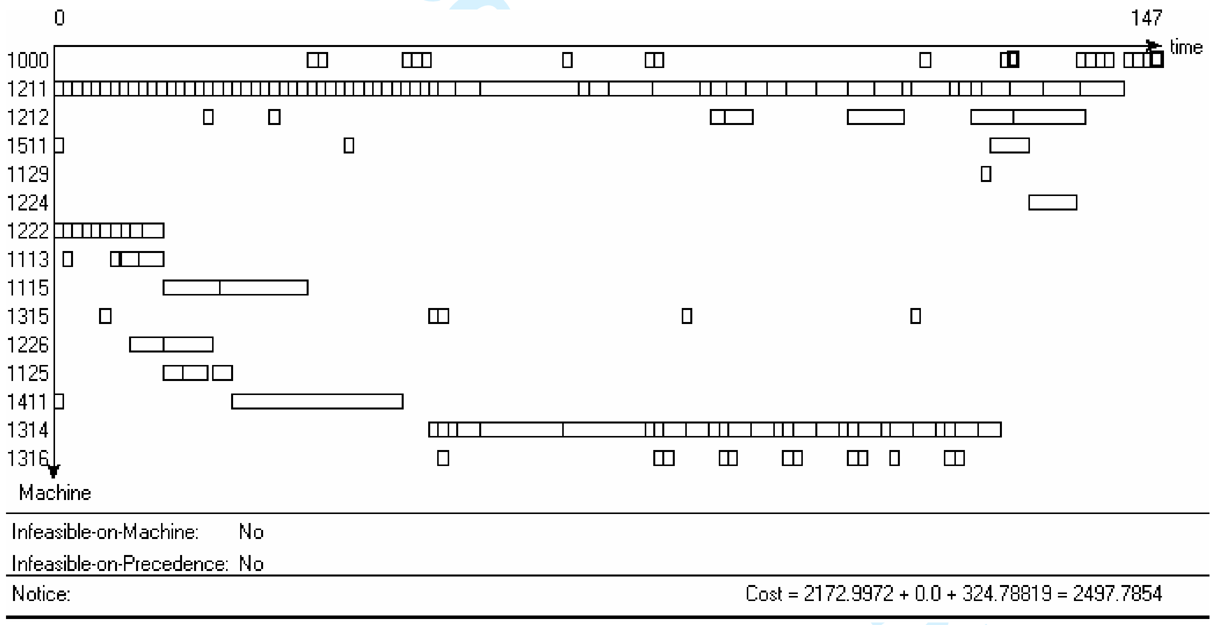
Figure 4 Gantt chart of the existing order



```
Infeasible-on-Machine:     No
Infeasible-on-Precedence:  No
Notice:                                   Cost = 2273.1294 + 0.0 + 609.30859 = 2882.438
```

Note: the shadowed bars represent the operations in the new order.

Figure 5 Gantt chart of the existing and new orders under ESIP



```
Infeasible-on-Machine:     No
Infeasible-on-Precedence:  No
Notice:                                   Cost = 2172.9972 + 0.0 + 324.78819 = 2497.7854
```

Figure 6 Gantt chart of the existing and new orders under ESRP

In this example, the new order competes for machines 1000, 1211 and 1315 with the existing order. Machine 1211 is the bottleneck resource, which is running with full capacity from time 0 to near end of the production (see figure 4~6). Using the ESIP method, the operations for the new order on machine 1211 have to be scheduled at a later time. This makes the new order finish at *d*=148 days (see figure 4). On the other hand, using the ESRP method, the new order can be finished at a relatively early time (see figure 5). Instead, the existing order will be completed later than its due date (*d*=144 days). However, the total tardiness cost is reduced from 609.31 to 324.79 and the total holding cost is reduced from 2273.13 to 2173.00 (figure 5 and 6). Hence, the ESRP approach is significantly better than the ESIP. This example also implies that a large benefit may be achieved by using ESRP if there is a lot of interaction between the new and existing orders.

7. Discussions

This section discusses some other real life factors that might affect the dynamic scheduling and indicates how the tools developed could be adapted.

The factory calendar (or number of shifts), operating hours, overtime work, outsourcing, and subcontracting affect the company's resource capacity. This could be included in both incremental and regenerative planning by readjusting the available resource sets R_1 and R_2, and adding the new relationship between operations and resources. The generated schedules would then reflect the influence of new resource capacity. On the other hand, by iteratively using the tool to perform dynamic scheduling with different resource capacity, it would help companies to identify which resources are in most need and which operations should be outsourced in order to meet due dates or balance workload.

The availability of material may affect the execution of the schedules. In this paper, the material available times are treated as decision variables (denoted by s_i for those operations without preceding operation). Therefore, it is the planner's responsibility to ensure that the materials are available as specified in the schedule. However, in some circumstances, there may be constraints on the material available times, in which case additional constraints should be introduced before performing the dynamic scheduling. To reflect such constraints, the algorithms should be changed slightly during checking the solution feasibility.

The computing time required by the tools developed in this paper may affect the application. For the case studies in the collaborating ETO Company, the products are capital goods and often have a very long lead time (e.g. one year). A few days of planning are acceptable. However, in other cases, the lead-time and due date may be relatively short. The application of the tools to those situations may be justified by: (i) the ES-based procedure could be terminated within the acceptable computing time range. Although it would have less chance of finding the best solution, it still returns the "optimal" solution obtained within the specified computational time. This could be used as guidance or reference point for the planner; (ii)

Computer speed is improving very quickly and the evolutionary algorithms could be implemented via parallel computing. This would help the application of ES-based tools; (iii) If the computing time is really a concern, the proposed heuristic methods (such as opt_BIP and opt_FRP) could be used, which can produce a reasonably good solution with very little computing time.

Sometimes, customers may want to change the due dates. In this situation, the order with a new due date could be excluded from the existing orders and regarded as a new order. Dynamic planning can be performed as described. If some operations in this order have been performed, they should be excluded from the schedule list.

8. Conclusions

This paper has analysed alternative methods for dynamically scheduling complex products that involve many levels of manufacturing and assembly processes with finite capacity constraints. This is a key issue in engineer-to-order companies. Unlike the majority of the literature that has used traditional performance measures, such as make span and maximum tardiness, this research has minimised a non-regular performance measure that includes the combination of lateness and work-in-progress holding costs. ETO companies often produce capital goods (e.g. turbine generators, oil rigs), which are highly customised and required in low volume. Late delivery may incur very high contractual penalty, while the early finishing of subassemblies and final products also incur expensive storage and maintenance costs. Therefore measuring performance in terms of total costs is important for ETO companies.

Incremental and regenerative dynamic scheduling problems have been investigated. The new order arrival time may be random but the operation processing times are assumed to be deterministic. Three methods, Forward Incremental Planning (FIP), Backward Incremental Planning (BIP) and Evolution Strategy Incremental Planning (ESIP), have been developed to deal with the incremental planning problem. The case studies show that in both tight and loose due date situations (for the new order), the ESIP is the best and FIP is the worst. In some situations, BIP has a close performance to ESIP (e.g. the cost reduction percentages achieved by ESIP from BIP in three examples with loose due dates are 8%, 2% and 0.3% respectively, where the cost refers the total cost for the new order only). As far as the computational time concerned, the heuristic methods are much better than evolution strategy.

To deal with regenerative planning problems, two methods, Forward Regeneration planning (FRP) and Evolution Strategy Regeneration Planning (ESRP), have been presented. The ESRP is much better than the FRP. As the complexity of the new order increases, the benefit of ESRP increases significantly.

Comparing the ESRP with ESIP, the results show that ESRP provides a better performance than ESIP but at the cost of more computational effort. More benefit may be achieved by using ESRP if the new order has a lot of interaction with the existing orders. However, if the new order is small, it appears that ESIP and ESRP produce schedules with close to optimal costs in both tight and loose due date situations.

The paper makes three main contributions. First, both heuristic and random search methods have been developed and used to solve realistic dynamic engineer-to-order scheduling problems with non-regular performance measures. Second, the methods were compared using case studies based on industrial data and their advantages and disadvantages. Third, the results provide planners with useful insights into choosing appropriate methods in different situations.

Further research will extend this approach to a wider class of rescheduling problems which include uncertainties such as stochastic manufacturing and assembly processing times and failure-prone machines.

9. References

Abumaizar, R.J. and Suestka, J.A., Rescheduling job shops under disruptions. *International Journal of Production Research*, 1997, 35(7), 2065-2082.

Artigues, C., Michelon, P. and Reusser, S., Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 2003, 149, 249-267.

Baker, K.R., *Introduction to sequencing and scheduling*, 1974 (Wiley & Sons: New York).

Baker, K.R. and Scudder, G.D., Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 1990, 38(1), 22-36.

Brandimarte, P. and Maiocco, M., Job shop scheduling with non-regular objective: a comparison of neighbourhood structures based on a sequencing/timing decomposition. *International Journal of Production Research*, 1999, 37(8), 1697-1715.

Brucker, P., Drexl, A., Moring, R., Neumann, K. and Pesch, E., Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 1999, 112(1), 3-41.

Chang, F.P.R., Heuristics for dynamic job shop scheduling with real time updated queuing time estimates. *International Journal of Production Research*, 1997, 35(3), 651-665.

Chryssolouris, G. and Subramaniam, V., Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, 2001, 12, 281-293.

Church, L.K. and Uzsoy, R., Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 1992, 5, 153-163.

Cowling, P. and Johansson, M., Using real time information for effective dynamic scheduling. *European Journal of Operational Research*, 2002, 139, 230-244.

Hicks, C., *The Application of Computer Aided Production Management Systems in Make-to-Order / Engineer-to-Order Heavy Engineering Companies*, PhD Thesis, University of Newcastle upon Tyne,1998.

Jain, A.K. and Elmaraghy, H.A., Production scheduling/rescheduling in flexible manufacturing system. *International Journal of Production Research*, 1997, 35, 281-309.

Li, R.K., Shyu, Y.T. and Adiga, S., A heuristic rescheduling algorithm for computer-based production scheduling systems. *International Journal of Production Research*, 1993, 31(8), 1815-1826.

Matsuura, H., Tsubone, H. and Kanezashi, M., Sequencing, dispatching and switching in a dynamic manufacturing environment. *International Journal of Production Research*, 1993, 31(7), 1671-1688.

Schwefel, H.P., *Evolution and Optimum Seeking*, 1995 (Wiley: New York).

Schwefel, H.P. and Back, T., Artificial evolution: how and why? in *Genetic Algorithm and Evolution Strategies in Engineering and Computer Science*, Edited by Quagliarella,D., Periaux,J., Poloni,C. and Winter,G., pp1-20, 1998, (John Wiley & Sons: England).

Song, D.P., *Stochastic models in planning complex engineer-to-order products*, PhD Thesis, University of Newcastle Upon Tyne, UK, 2001.

Song, D.P., Hicks, C. and Earl, C.F., Product due date assignment for complex assemblies, *International Journal of Production Economics*, 2002, 76(3), 243-256.

Vieira, G.E., Herrmann, J.W. and Lin, E., Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies. *International Journal of Production Research*, 2000, 38(8), 1899-1915.

Vollmann, T.E., Berry, W.L. and Whybark, D.C., *Manufacturing Planning and Control Systems*, 1992 (Irwin: Homewood, IL).

Yamamoto, M. and Nof, S.Y., Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 1985, 23(4), 705-722.

Yeh, C.H., A fast finite loading algorithm for job oriented scheduling. *Computer and Operations Research*, 1997, 24(2), 193-198.