Optimisation of Integrated Process Planning and Scheduling Using a Particle Swarm Optimisation Approach

Yanwu Guo, Weidong Li, a R Mileham, G W Owen

Optimisation of Integrated Process Planning and Scheduling Using a Particle Swarm Optimisation Approach

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0837.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	23-Nov-2007
Complete List of Authors:	Guo, Yanwu; University of Bath, Mechanical Engineering Li, Weidong; Coventry University, Faculty of Engineering and Computing Mileham, A R; University of Bath, Department of Mechanical Engineering Owen, G W; University of Bath, Department of Mechanical Engineering
Keywords:	ARTIFICIAL INTELLIGENCE, SCHEDULING, EVOLUTIONARY ALGORITHMS, PROCESS PLANNING
Keywords (user):	

Optimisation of Integrated Process Planning and Scheduling Using a Particle Swarm Optimisation Approach

Y.W. Guo[1*], W.D. Li[2], A.R. Mileham[1], G.W. Owen[1]

[1] Department of Mechanical Engineering, University of Bath

Claverton Down, Bath, BA2 7AY, UK

[2] Department of Engineering and Manufacturing Management, Faculty of Engineering and Computing, Coventry University

Priory Street, Coventry, CV1 5FB, UK

* Email: y.guo@bath.ac.uk Fax: (44) 0122538 6928

ABSTRACT

Traditionally process planning and scheduling are two independent essential functions in a job shop manufacturing environment. In this paper, a unified representation model for Integrated Process Planning and Scheduling (IPPS) has been developed. Based on this model, a modern evolutionary algorithm, i.e., the Particle Swarm Optimisation (PSO) algorithm has been employed to optimise the IPPS problem. To explore the search space comprehensively and to avoid being trapped into local optima, the PSO algorithm has been enhanced with new operators to improve its performance and different criteria, such as makespan, total job tardiness and balanced level of machine utilisation, have been used to evaluate the job performance. To improve the flexibility and agility, a replanning method has

been developed to address the conditions of machine breakdown and new order arrival. Case studies have been used to verify the performance and efficiency of the modified PSO algorithm under different criteria. A comparison has been made between the result of the modified PSO algorithm and those of the Genetic Algorithm (GA) and the Simulated Annealing (SA) algorithm respectively, and different characteristics of the three algorithms are indicated. Case studies show that the developed PSO can generate satisfactory results in optimising the IPPS problem.

Keywords: Integrated Process Planning and Scheduling, Particle Swarm Optimisation, Genetic Algorithm, Simulated Annealing, Replanning

1. Introduction

In job shop and batch manufacturing, both process planning and scheduling are responsible for the efficient allocation and utilisation of manufacturing resources. Process planning, as defined by Chang and Wysk (1985), is the act of preparing detailed operation instructions to transform an engineering design to a final part. In this process, the decision of which manufacturing resources to select is usually made based on the objective of achieving the correct quality, the minimal manufacturing cost and ensuring good manufacturability. Scheduling is the function of assigning manufacturing resources to parts and their operations indicated in process plans in such a way that the competition and conflict for the resources can be resolved. The objectives often relate to balanced level of machine utilisation,

minimised makespan and total tardiness. Usually, a process plan is determined before the actual scheduling with no regard for the scheduling objectives and with the assumption of all the resources are available. However, this sequential arrangement of the two functions ignores their close relationship. The two functions are interrelated because both of them take part in the assignment of machines to production tasks (Moon and Seo 2005). If a process plan is prepared offline without due consideration of the actual shop floor status, it may not be an optimal solution due to a heavily unbalanced resource assignment, or even become unfeasible due to changes or constraints in the manufacturing environment. Meanwhile, with the different objectives of these two functions, it is difficult to produce a satisfactory result in their sequential executions. As thus, Integrated Process Planning and Scheduling (IPPS) has been proposed, aiming to increase production feasibility and optimality by combining both the process planning and scheduling problems (Huang et al. 1995).

Over the last decade, a number of research efforts have been made to solve the IPPS problem. An earlier review of different approaches can be found in Tan and Khoshnevis (2000). The most recent works can be generally classified into two categories: the enumerative approach and the simultaneous approach (Li and McMahon 2007). In the enumerative approach (Zhang et al. 2003, Tonshoff et al. 1989, Sormaz and Khoshnevis 2003, Aldakhilallah and Ramesh 1999), alternative process plans for each part are first generated. A schedule can then be determined by iteratively selecting a suitable process plan from the alternative plans to replace

the current plan until a satisfactory performance is achieved. The simultaneous approach (Moon et al. 2002, Moon and Seo 2005, Kim et al. 2003, Yan et al. 2003, Zhang and Yan 2005, Li and McMahon 2007) is based on the idea of finding a solution from the combined solution space of process planning and scheduling. In this approach, the process planning and scheduling are both in dynamic adjustments until specific performance criteria are satisfied. Although this approach is more effective and efficient in integrating the two functions, it also enlarges the solution search space significantly.

To facilitate the optimisation process, some optimisation approaches based on modern heuristic algorithms and Artificial Intelligence (AI) technologies, such as the Genetic Algorithm (GA) (Morad and Zalzala 1999, Kim et al. 2003, Moon and Seo 2005, Zhang and Yan 2005), Simulated Annealing (SA) algorithm (Zhang et al. 2003, Li and McMahon 2007), Tabu search algorithm (Yan et al. 2003) and Agent-based approach (Wong et al. 2006), have been developed in the last decade and significant improvements have been achieved. However, the following issues are still outstanding:

(1) Most of the developed systems are unable to address the dynamic changes in shop floors effectively, such as routine machine maintenance, machine breakdown and new order arrival. Any occurrence of these situations will probably make the current schedule infeasible and result in a need to replan the schedule. The replanning process is more complex and time consuming due to the changed situations and manufacturing resource constraints.

(2) Both process planning and scheduling are *NP*-hard (Non-deterministic Polynomial) combinatorial optimisation problems. Compared to the problem of operation sequencing optimisation for a single part (Guo et al. 2006), there are two major difficulties in IPPS, (1) the search space of IPPS is much bigger than that of operation sequencing of a single part, and (2) the optimisation of IPPS becomes more complicated as the number of parts increases and there are more complex manufacturing constraints (such as operation precedence constraints and manufacturing resource constraints). All of these will increase the computation time dramatically.

As thus, it is necessary to develop an adaptive and unified model for the IPPS problem and an efficient optimisation algorithm. Particle Swarm Optimisation (PSO) is a modern evolutionary computation technique based on a population mechanism (Kennedy and Eberhart 1995). It has been motivated by the simulation of the social behaviour of individuals (particles). The PSO algorithm was initially developed for continuous optimisation problems. Recently, there has been successful research focused on discrete problems such as the Travelling Salesman Problem (TSP) (Wang et al. 2003, Pang et al. 2004, Onwubolu and Clerc 2004), operation sequencing problem (Guo et al. 2006) and the scheduling problem (Jerald et al. 2005). In this paper, a new PSO-based optimisation algorithm for the IPPS problem has been developed. A case study with computational experiments to test

the algorithm on two groups of jobs is demonstrated, and a comparison between the result of the PSO algorithm and that of previous work is presented.

2. Representation of the IPPS

2.1 PSO algorithm

The PSO algorithm was inspired by the social behaviour of bird flocking and fish schooling (Kennedy and Eberhart 1995). Three aspects will be considered simultaneously when an individual fish or bird (particle) makes a decision about where to move: (1) its current moving direction (velocity) according to the inertia of the movement, (2) the best position that it has achieved so far, and (3) the best position that its neighbour particles have achieved so far. In the algorithm, the particles form a swarm and each particle can be used to represent a potential solution of a problem. In each iteration, the position and velocity of a particle can be adjusted by the following formulae that take the above three considerations into account. After a number of iterations, the whole swarm will converge at an optimized position in the search space.

$$V_i^{t+1} = w * V_i^t + c_1 * Rand() * (P_i^t - X_i^t) + c_2 * Rand() * (P_g^t - X_i^t) \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2)$$

$$X_i = (X_{i1}, X_{i2}, ..., X_{iN}) \quad (3)$$

$$V_i = (V_{i1}, V_{i2}, ..., V_{iN}) \quad (4)$$

Here, i is the index number of particles in the swarm; t is the iteration number; V and X are the velocity vector and the position vector of a particle respectively.

For an N-dimensional problem, V and X can be represented by N particle dimensions as formulas (3) and (4) show. P_i is the local best position that the ith particle has achieved so far; P_g is the global best position that all the particles have achieved so far; w is the inertia weight to adjust the tendency to facilitate global exploration (smaller w) and the tendency to facilitate local exploration to fine-tune the current search area (larger w); $Rand()$ returns a random number in [0,1]; c_1 and c_2 are two constant numbers to balance the effect of P_i and P_g.

2.2 Definition of IPPS

The IPPS problem can be defined as:

Given a set of n parts, and each of which has a number of operations. The parts are processed on m machines with alternative manufacturing plans (machines, tools and Tool Approach Directions-'TADs'). The objective of the problem is to select suitable manufacturing resources and sequence the operations so as to determine a schedule in which the precedence constraints between operations can be satisfied and the corresponding objectives can be achieved.

Figure 1 is used to illustrate this problem. For instance, there are 3 parts that can be machined by 3, 2 and 3 operations on 3 machines, respectively. For the different parts, there are precedence constraints among the operations to machine them (Part1: Oper1→Oper2→Oper3, Part2: Oper4→Oper5, Part3: Oper6→Oper7→Oper8). When all these 8 operations are sequenced (Oper1→Oper4→Oper2→ Oper6→Oper3→Oper7→Oper8→Oper5 as shown in

Figure 1) and the manufacturing resources are specified (machines, tools and TADs), the schedule can be determined accordingly. The optimisation problem is to determine the operation sequence and select the manufacturing resources so as to achieve the optimisation objectives (Makespan in Figure 1, for example) whilst maintaining the schedule and process planning feasible.

(Here inserts Figure 1)

2.3 Representation of IPPS

To apply the PSO algorithm to the optimisation of the IPPS problem, two issues have to be handled first:

(1) Encode a solution (here a solution refers to a sequence of operations) to produce a particle.

As shown in Figure 2, each operation is modelled as a particle dimension of the PSO algorithm, and the detailed information is listed in Table 1. Several new variables, including *Mac_time*, *Change_time*, *Machine_s_time* and *Machine_e_time*, are added to record and track the time related to the execution of the operation so as to determine the time allocation on the machines. Here, a position variable and a velocity variable are used to represent the position and velocity of an operation, respectively. As shown in Table 2, the array variable *Oper[n]* represents a solution that consists of *n* ParticleDimensions (operations). A particle can be initialised in the following steps:

- All the operations are given an Operation_id from 1 to *n*.
- Machine_list, Tool_list and TAD_list applicable for each operation are specified, and a machine, tool and TAD are randomly selected from the three lists to execute the operation.
- Mac_time, Change_time, Machine_s_time and Machine_e_time are set as 0 initially.
- A random position between [0, 1] and a random velocity between [-1, 1] are initialised for each ParticleDimension in the particle. The sequence of operations is determined by the relative values of their positions.

(2) Decode the particle to get a solution.

In each iteration, when all the ParticleDimensions in a particle have been updated, the sequence of all the operations to machine the parts can be determined by the relative positions of the ParticleDimensions (Cagnina et al. 2004). As discussed in Section 2.2, when the sequence for all the operations is generated and the manufacturing resources are selected, the assignments of specific operations and machines are determined and therefore the schedule is obtained. By using a number of iterations to update the positions and velocities of the particle dimensions in each particle, an optimised sequence (i.e., an optimized solution) can be achieved eventually.

(Here inserts figure 2)

(Here insert table 1 and table 2)

A feasible solution of the IPPS problem must comply with precedence constraints that come from geometrical and technological considerations. The precedence constraints between operations are usually classified into six types: (1) fixture interaction, (2) tool interaction, (3) datum interaction, (4) thin-wall interaction, (5) material-removal interaction, and (6) fixed order of machining operations. The definitions and illustrations of these constraints can be seen in works of Li et al. (2004) and Guo et al. (2006).

The IPPS problem can be modeled as an extension of the operation sequencing optimisation problem relating to a single part (Li et al. 2004, Guo et al. 2006) into a multiple parts with the IPPS objectives. When the process plans of all parts are generated and the manufacturing resources are specified, it is required to determine the schedule based on this information and calculate the makespan, total tardiness, etc. Here, three evaluation criteria of the IPPS problem can be calculated as follows.

Makespan: $Makespan = \underset{j=1}{\overset{m}{Max}}(Machine[j].Available\_time)$.

Here $Machine[j].Available\_time$ is the time when machine is available for next operation.

Total job tardiness: The due date of a part is denoted as DD, and the completion moment of the part is denoted as CM. Hence,

$$Part\_Tardiness = \begin{cases} 0 & if\ DD\ is\ later\ than\ CM \\ CM - DD & Otherwise \end{cases}$$

Balanced level of machine utilization: the Standard Deviation concept is introduced here to evaluate the balanced level of machine utilization (assuming there are m machines, and each machine has n operations).

Total machining time of Machine[j]:

$$Machine[j].Utilization = \sum_{i=1}^{n} (Operation[i].Mac\_T), (j = 1,..,m)$$

Mean machining time of all the machines:

$$\chi = \frac{\sum_{j=1}^{m} (Machine[j].Utilization)}{m}$$

Standard Deviation to the Mean machining time:

$$Utilization\_Level = \sqrt{\sum_{j=1}^{m} (Machine[j].Utilization - \chi)^2}$$

3. The PSO algorithm with replanning ability

The above approaches for the IPPS problem do not consider the possibility of making dynamic changes in shop floors, such as routine machine maintenance, machine breakdown and new order insertion to the current schedule to meet the deadlines. Any occurrence of these situations will probably make the current schedule unfeasible and require the replaning of the whole schedule. In this research, two types of changes are considered, namely machine breakdown and new order arrivals. The following will discuss these two situations respectively.

3.1 Machine breakdown

If a machine breaks down, it will not only affect the part being machined on it, but also make other parts that are going to be executed on this machine unfeasible. Suppose $Machine[j]$ breaks down at time T_b, and the reparation of the machine requires time T_p. The following assumptions are made:

- The replanning generates a schedule from the next available time for $Machine[j]$, $j = 1,2,...m$.
- The available time of the machine that breaks down is $Machine[j].Available\_time = T_b + T_p$.
- The breakdown of $Machine[j]$ does not affect the current operations of other machines. If an operation $Oper[i]$ is being executed on $Machine[k](k \neq j)$ when $Machine[j]$ breaks down, then the available time of the $Machine[k](k \neq j)$ can be computed as follows: $Machine[k].Available\_time = Oper[i].Machine\_e\_time$.
- If no job is being processed on $Machine[k](k \neq j)$ when $Machine[j]$ breaks down, then the available time of the $Machine[k](k \neq j)$ can be computed as $Machine[k].Available\_time = T_b$.
- If there is a part being machined when the machine breaks down, it does not destroy the part and only the operation disturbed needs to be re-executed in one of two ways: a) to be machined on the current machine after it is repaired, and b) to be rescheduled to be executed on other machines.

- Only the operations that have not been executed and the operation being executed on the broken down machine need to be rescheduled from the machine available time obtained previously.

With the above assumptions, it can be seen in Figure 3, when machine 2 breaks down at time T_b, the available times for three machines are T_1, T_2 and T_3 respectively.

(Here inserts figure 3)

Therefore, with these assumptions, the replanning of the scheduling problem can be resolved by two changes applied to the PSO algorithm described before:

1) Reduce the operations range to the operations that have not been executed.
2) Initialise the operations (particle dimensions) and machines with the new generated available time.

3.2 New order arrival

Compared to machine breakdowns, the situation of the arrival of a new order is less complex. Suppose the new part arrives at time T_a. The following assumptions are made:

- The replanning generates a schedule from the next available times for $Machine[j]$, $j = 1,2,...m$.

- If an operation $Oper[i]$ is being executed on $Machine[j]$ when the new part arrives, then the available time of the $Machine[j]$ can be computed as follows: $Machine[j].Available\_time = Oper[i].Machine\_e\_time$.

- If no job is being processed on $Machine[j]$ when a new part arrives, then the available time of the $Machine[j]$ can be computed as: $Machine[j].Available\_time = T_a$.

- Only the operations that have not been executed and the new operations that are required to machine the new part need to be rescheduled from the machine available time obtained previously.

With the above assumptions, it can be seen in Figure 4, when a new order arrives at time T_a, the available times for three machines are T_1, T_2 and T_3 respectively.

(Here inserts figure 4)

Therefore, with these assumptions, the replanning of the scheduling problem can be resolved by two changes applied to the PSO algorithm described before:

1) Increase the operations range, including the operations of old parts that have not been executed and the operations that are required to machine the new part.
2) Initialise the operations (particle dimensions) and machines with the new generated available time.

3.3 Method to improve the efficiency of the algorithm in replanning

It is required to reduce the computation time for generating a new schedule quickly when encountering the above situations. However, the process of replanning will take more time, especially when adding new orders as it will increase the search space and there is a need to keep the schedule feasible with the consideration of more precedence constraints. As presented above, the critical step to replan the schedule is the initialisation of the particle (all the operations need to be scheduled). Furthermore, the old schedule generated was feasible and optimised whilst complying with all the precedence constraints before the situation occurred. For efficiency, it is better to minimise changes to the existing plan as some allocated resources may already be in place, e.g., tools and materials taken to machines in advance. Therefore, the strategy has been used to update the old schedule with some modifications as a new particle:

- For the situations of machine breakdown, it is possible to initialise a particle by three steps: a) delete the operations that have been executed in the old schedule, b) keep the *velocity* and *position* values to keep the sequence of the operations, and c) change the corresponding available time for the machines.

- For the situations of new order arrival, it is possible to initialise a particle by the following steps: a) delete the operations that have been executed in the old schedule, b) keep the *velocity* and *position* values to keep the sequence of the operations in the old schedule, c) add the operations that are required to

machine the new part to the end of old schedule, d) initialise the newly added operations by selecting alternative manufacturing resources and set the *position* and *velocity* values, and d) changing the corresponding available time for the machines.

With this method, the optimised sequence in the old schedule is mostly kept and this saves a large amount of computation, and hence reduces the time for replanning the schedule.

4. The Modified PSO Algorithm

A traditional PSO algorithm can be applied to optimise the IPPS problem in the following steps:

(1) Initialisation:

- Set the size of a swarm, e.g., the number of particles "*Swarm_Size*" and the maximum number of iterations "*Iter_Num*".
- Initialise all the particles in the method introduced in section 2.3. Decode every particle (solution) to get the schedule of the particle and then calculate the corresponding criteria of particle (the result is called *fitness* here) as described in section 2.3.
- Set the local best $P_i[n]$ and the global best P_g with the best *fitness*.

(2) Iterate the following steps until *Iter_Num* is reached:

- For each particle in the swarm, and each ParticleDimension, (i.e., operation in particle), update ParticleDimension's velocity and position values according to formulas (1) and (2), i.e., *Oper[1].Position*, *Oper[2].Position*, …, *Oper[n].Position*.
- Decode the particle into a solution in terms of new position values and calculate the *fitness* of the particle. Update the local best $P_i[n]$ and the global best P_g if a lower *fitness* is achieved.

(3) Decode global best P_g to get the optimised solution.

However, the traditional PSO algorithm introduced above is still not effective in resolving the operation sequencing problem. There are two major reasons for this:

(1) Due to the inherent mathematical operators, it is difficult for the traditional PSO algorithm to consider the different arrangements of machines, tools and TADs for each operation, and therefore the particle is unable to fully explore the whole search space.

(2) The traditional algorithm usually works well in finding solutions at the early stage of the search process (the optimisation result improves fast), but is less efficient during the final stage. Due to the loss of diversity in the population, the particles move quite slowly with low or even zero velocities and this make it is hard to reach the global best solution (Stacey et al. 2003). Therefore, the whole swarm is prone to be trapped in a local optimum from which it is difficult to escape.

To solve these two problems and enhance the ability of the traditional PSO algorithm to find the global optimum, new operations, including mutation, crossover and shift, have been developed and incorporated in a modified PSO algorithm. Meanwhile, considering the characteristics of the algorithm, the initial values of the particles and P_g (the global best position of all the particles in formula (1)) have been well planned. Some modification details are depicted below.

(1) New operators in the algorithm

- Mutation. In this strategy, an operation is first randomly selected in a particle. From its candidate machining resources (Machine_list[], Tool_list[] and TAD_list[]), an alternative set (machine, tool, TAD) is then randomly chosen to replace the current machining resource in the operation. This operator enables the PSO algorithm to select the alternative Machine-Tool-TAD candidates so the optimisation can be preceded. The probability of applying this strategy is defined as P_m.
- Crossover. Two particles in the swarm are chosen as Parent particles for a crossover operation. In the crossover, a cutting point is randomly determined, and each parent particle is separated as left and right parts of the cutting point. The positions and velocities of the left part of Parent 1 and the right part of Parent 2 are reorganised to form Child 1. The positions and velocities of the left part of Parent 2 and the right part of Parent 1 are reorganised to form Child 2. The probability of applying the crossover is defined as P_c.

- Shift. This operator is used to exchange the positions and velocities of two operations in a particle so as to change their relative positions in the particle. The probability of applying the shift is defined as P_s.

(2) Escape method for P_g

- During the optimisation process, if the iteration number of obtaining the same best fitness is more than 10, then the mutation and shift operations are applied to P_g to try to escape from the local optima.

5. Case Studies and Discussions

Two experiments are used here to verify the efficiency of the PSO algorithm for the IPPS problem. The first experiment is used to compare the efficiencies of the PSO, GA and SA algorithms. The second experiment is used to verify the replanning ability of the PSO algorithm under machine breakdown and new order arrival conditions. For simplification, the parameters of the PSO algorithm recommended in Guo et al.'s work (2006) are used in the PSO algorithm for experiments in this paper.

5.1 Experiment 1

The example parts and manufacturing resources from Li and McMahon (2007), which were for comparing GA and SA on optimisation of process planning, are used here to verify and compare the efficiencies of the PSO, GA and SA approaches. Two groups of parts are used for the experiment.

Group 1:

The first group consists of three parts that are taken from the works of Shah, *et al.* (1995) and Zhang, *et al*. (1997). The specifications of the parts are shown in Table 3.

(Here inserts Table 3)

Two criteria are used here as the optimisation direction, i.e., the makespan and the balanced machine utilisation.

The optimisation processes and results of the PSO algorithm and influence of applying new operators are shown in Figures 5 and 6 respectively (the mutation operator is not included because it is an essential operator to enable optimisation to select alternative Machine-Tool-Tad candidates). From these two figures, it can be seen that the PSO can optimise Makespan and Balanced Level of Machine Utilisation for Group 1 successfully, and the performance of the PSO is improved with these two new operators (The PSO with both crossover and shift converges faster and can achieve better result than the PSO without crossover especially). Figure 7 shows an optimised result for Makespan in Gantt chart. The optimised schedule for a minimised Makespan can be achieved after nearly 3000 iterations and the optimised schedule for Balanced Level of Machine Utilisation can be achieved more quickly, after 200 iterations.

(Here insert Figures 5, 6 and 7)

Two other evolutionary algorithms, GA and SA developed by Li and McMahon (2007), are used to compare the optimised results, computation efficiency and robustness. Figures 8 and 9 show the optimisation results of GA, SA and PSO for

two objectives respectively. The optimisation results are based on 5000 iterations for each algorithm. The population of the GA and the PSO are both set as 200.

(Here insert Figures 8 and 9)

Makespan:

As Table 4 shows, the SA takes 59 minutes to finish 5000 iterations, so Figure 8 shows the results after an 8 minute run. As shown in Table 4 and Figure 8, with the same time period, the SA and the PSO can achieve better results than GA, but the SA is not as robust as the GA and PSO. For 20 random consecutive trials, the SA optimise successfully in 14 trials (Here the optimisation is regarded unsuccessful if no better solutions than the initialised solution can be found in 5 minutes after the starting of the optimisation process), the PSO and the GA can optimise successfully in all 20 trials.

(Here insert Tables 4 and 5)

Balanced Level of Machine Utilisation:

From Table 5 and Figure 9, it can be seen that all three algorithms can achieve the optimised results in all 20 consecutive trials, whilst the GA and the SA algorithms approach the optimised result more quickly.

Group 2:

Eight parts taken from (Li and McMahon, 2007) have been used to test the algorithm under more complex conditions. The relevant specifications of the parts

are given in Table 6. The above two objectives have been used again, and the optimisation results are shown in Figures 10 and 11. It can be seen that the PSO can optimise Makespan after nearly 4000 iterations and Balanced Level of Machine Utilisation after 3000 iterations.

(Here inserts Table 6)

(Here insert Figures 10 and 11)

The comparisons of the GA, SA and PSO designed for Group 1 are used to compare the results, efficiencies and robustness for group 2 as well.

Makespan:

As shown in Table 7 and Figure 12, with the same time period, the PSO and the SA can achieve better results than the GA. However, for 20 random consecutive trials, the SA can only proceed with successful optimisation in 6 trials, the PSO and the GA can proceed with successful optimisation in all 20 trials.

(Here insert Tables 7 and 8)

(Here insert Figure 12 and 13)

Balanced Level of Machine Utilisation:

From Table 8 and Figure 13, it can be observed that all of the algorithms can achieve good results, while different characteristics are shown due to the inherent mechanisms of the algorithms. The SA is much "sharper" to find optimized solutions than the GA and the PSO. The SA can achieve better results than the GA and the PSO. However, in 20 random consecutive trials, the SA can only proceed

with successful optimisation in 6 trials but the GA and the PSO can proceed with successful optimisation in all 20 trials.

Summary of GA, SA and PSO algorithms

As discussed in above section and in Guo et al.'s work (2006), the GA, SA and PSO algorithms are used to optimise the operation sequencing problem and the IPPS problem. All of them can yield good results, but they have different characteristics. The GA and the PSO are both population based algorithms except the SA. Therefore, the optimisation processes of the GA and the PSO take a longer time than that of the SA (Guo et al. 2006). It can also be observed that the PSO needs to adjust the particle dimensions by updating the velocities and positions of them due to its intrinsic mechanism so that it needs more computation time than the GA. For the optimisation results, the SA and the PSO both outperform the GA in all the above case studies. As the complexity of the problem increases (for example when optimising IPPS problems), the SA can achieve better results than the GA and the PSO in the case studies described above. But as the complexity of the problem increases, the SA is not as robust as the GA and the PSO. This is probably because the SA is not population based, so that the initial plan does not have enough diversity to enable it to search the space successfully. Also as the complexity of the problem increases, it can be seen that the optimisation speed advantages of the GA and the SA over the PSO diminish. It is well known that simple mathematic operations run much faster than other position changing operations. This can

probably be attributed to the fact that each iteration of the PSO algorithm uses mainly simple mathematical operators that can be finished in a shorter time than for the GA and the SA algorithms with mainly complex position changing operators. In constraints handling, the GA and the SA can use the adjustment method developed by Li et al. (2002) to keep the plan feasible, but the PSO can only use the penalty method to enable the results to comply with the constraints due to its intrinsic mechanism. The above discussion is illustrated in Table 9.

(Here inserts Table 9)

5.2 Experiment 2

The parts in group 2 have been used to test the replanning ability of the PSO developed for IPPS under machine breakdown and new order arrival conditions. In this experiment, as new order arrivals and machine breakdowns occur, it is appropriate to set the total tardiness as the main objective, so as to make comparison under these two conditions.

1. First planning

The 8 parts in group 2 consist of a total of 59 operations. Here the Due Date (DD) is set as 2700.0. Table 10 shows the first scheduling results of the complete time for individual parts in group 2. It also can be seen from Figure 14, the process can be optimised to achieve the DD for all the parts.

(Here inserts Figure 14)

(Here inserts Table 10)

2. Condition of new order arrival

At time 1000.0, a new order arrives (part 9 in this experiment which is the same as part 1) and the corresponding DD is set as 3500.0. At the time 1000.0, 18 operations have been finished and 41 operations are left. With part 9 added as a new part, 7 operations are then inserted into the total operation list which includes 48 operations. The individual available time for all the machines is shown in Table 11. The optimisation result is shown in Figure 14 and the individual complete time for all the 9 parts after replanning is shown in Table 12.

(Here insert Tables 11 and 12)

3. Condition of machine breaks down

At time 1500.0, machine 3 breaks down (repair time 300.0). Table 13 shows the available times for different machines. At that time, 16 operations have been finished and only 32 operations are left. The optimisation result is shown in Figure 14 and the individual complete time for all the 9 parts after replanning is shown in Table 14.

(Here insert tables 13 and 14)

Because the algorithm will not continue the optimisation when it achieves the lowest value in terms of the objective (here total tardiness=0), it can find the earliest

complete date for parts by reducing the DD. For example if DD(part 1-8) = 1500.0, DD(9) = 2500, the planning results are shown in Tables 15, 16 and 17:

(Here insert tables 15, 16 and 17)

From this case study, it can be seen that the modified PSO algorithm has the ability to replan when new order arrival and machine breakdowns occur. Figure 14 shows that with the method discussed in section 3, the replanning time can be reduced and the computation efficiency can be improved significantly.

6. Conclusions

Efficient and adaptive integration of process planning and scheduling has become imperative in order to optimise the decisions of allocating manufacturing resources in a job shop/batch manufacturing environment. To realise this, it is required to consider the dynamic changes of the shop floor's situation and adopt a more efficient and adaptive algorithm to optimise it.

In this research, the IPPS problem has been defined and a modified PSO algorithm has been used to optimise it. Solutions to the IPPS problem are encoded into PSO particles to intelligently search for the best sequence of the operations through leveraging the optimisation strategies of the PSO algorithm. To explore the search space more effectively, new operators, i.e., mutation, crossover and shift have been developed and incorporated to produce a modified PSO algorithm with improved performance. In order to react to the dynamic changes of shop floor situation (machine breakdown and new order arrival), the method to equip the

algorithm with replanning ability has been proposed. The GA and SA algorithms have been used to verify and compare the performance of the modified PSO algorithm with experiments of two groups' parts. It is shown that the PSO algorithm can obtain a satisfactory optimisation result for the IPPS problem and can execute replanning efficiently. The characteristics of the GA, SA and PSO algorithms have been given and for these cases. The PSO algorithm has been shown to outperform both the GA and SA in applications by considering the computation efficiency, optimality and robustness. At this point in time the conclusions are limited by this computational experience, and more theoretical analysis needs to be made in future. The PSO algorithm has shown the significant improvement in performance by applying the crossover operator taken from GA. Therefore it is possible to introduce other new operators and inspirations from other algorithms in future. With the population based characteristics, a bounded rationality mechanism which is used in social science and economics can also be applied in future to improve the performance of the algorithm further.

7. Acknowledgements

This work is funded by the Innovative design & Manufacturing Research Centre (IdMRC) and the Department of Mechanical Engineering at the University of Bath.

References:

1. Aldakhilallah K.A. and Ramesh R., Computer-Integrated Process Planning and Scheduling (CIPPS): intelligent support for product design, process planning and control, International Journal of Production Research, 1999, 37(3), 481-500.
2. Cagnina L., Esquivel S. and Gallard R., Particle Swarm Optimization for Sequencing Problems: A Case Study, In 2004 Congress on Evolution computation (CEC2004), Portland, USA, 2004, 1, 536-541.
3. Chang T.C. and Wysk R.A., An introduction to Automated Process Planning Systems, Prentice-Hall Inc., Englewood, Cliffs, New Jersey, USA, 1985.
4. Guo Y.W., Mileham A.R., Owen G.W. and Li W.D., Operation Sequencing Optimization using a Particle Swarm Optimisation Approach, 2006, Proceedings of the Institution of Mechanical Engineers, Journal of Engineering Manufacture, Part B, 220(B12), 1945-1958.
5. Huang, S.H., Zhang, H.C. and Smith, M.L., A progressive approach for the integration of process planning and scheduling. IIE Trans., 1995, 1, 456-464.
6. Jerald J., Asokan P., Prabaharan G. and Saravanan R., Scheduling optimization of flexible manufacturing systems using particle swarm optimization algorithm, International Journal of Advanced Manufacturing Technology, 2005, 25, 964-971.
7. Kennedy J. and Eberhart R., Particle Swarm Optimization, In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 1995, IV, 1942-1948 (IEEE Service Centre, Piscataway, New Jersey).

8. Kim, Y.K., Park, K. and Ko, J., A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Computers & Operations Research, 2003, 30, 1151-1171.

9. Li, W.D. and Mcmahon, C.A., A Simulated Annealing-based Optimization Approach for Integrated Process Planning and Scheduling, International Journal of Computer Integrated Manufacturing, 2007, 20, 80-95.

10. Li W.D., Ong S.K. and Nee A.Y.C., Optimization of process plans using a constraint-based tabu search approach, International Journal of Production Research, 2004, 42(10), 1955-1985.

11. Moon Chiung, Kim Jongsoo and Hur Sun, Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Computers & Industrial Engineering, 2002, 43, 331-349.

12. Moon Chiung and Seo Yooho, Evolutionary algorithm for advanced process planning and scheduling in a multi-plant, Computers & Industrial Engineering, 2005, 48, 311-325.

13. Morad, N. and Zalzala, A., Genetic algorithms in integrated process planning and scheduling. Journal of Intelligent Manufacturing, 1999, 10, 169-179.

14. Onwubolu G.C. and Clerc M., Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization, International Journal of Production Research, 2004, 42(3), 473-491.

15. Pang W., Wang K.P, Zhou C.G. and Dong L.J., Fuzzy Discrete Particle Swarm Optimization for Solving Travelling Salesman Problem, In Proceedings of the

Fourth International Conference on Computer and Information Technology (CIT'04), Wuhan, China, 2004.

16. Sormaz, D. and Khoshnevis, B., Generation of alternative process plans in integrated manufacturing systems. Journal of Intelligent Manufacturing, 2003, 14, 509-526.
17. Stacey A., Jancic M. and Grundy I., Particle Swarm Optimization with Mutation, In 2003 Congress on Evolution Computation (CEC2003), 2003, 2, 1425-1430.
18. Tan, W. and Khoshnevis, B., Integration of process planning and scheduling – a review. Journal of Intelligent Manufacturing, 2000, 11, 51-63.
19. Tonshoff, H.K., Beckendorff, U. and Andres, N., FLEXPLAN: A concept for intelligent process planning and scheduling. Proceedings of the CIRP International Workshop, Hannover, Germany, 1989, 319-322.
20. Wang K.P., Huang L., Zhou C.G. and Pang W., Particle Swarm Optimization for Travelling salesman problem, In Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, China 2003.
21. Wong T.N., Leung C.W., Mak K.L. and Fung R.Y.K., An agent-based negotiation approach to integrate process planning and scheduling, International Journal of Production Research, 2006, 44(7), 1331-1351.
22. Yan, H.S., Xia, Q.F., Zhu, M.R. Liu, X.L. and Guo, Z.M., Integrated production planning and scheduling on automobile assembly lines. IIE Transactions, 2003, 35, 711-725.
23. Zhang F, Zhang Y.F and Nee A.Y.C., Using genetic algorithms in process

planning for job shop machining, 1997, IEEE Transactions on evolutionary computation, Vol.1, No.4.

24. Zhang, X.D. and Yan, H.S., Integrated optimization of production planning and scheduling for a kind of job-shop. International Journal of Advanced Manufacturing Technology, 2005, 26, 876-886.

25. Zhang, Y.F., Saravanan, A.N. and Fuh, J.Y.H., Integration of process planning and scheduling by exploring the flexibility of process planning. International Journal of Production Research, 2003, 41(3), 611-628.

Table 1 Class definition of a particle dimension (an operation)

Class ParticleDimension: an operation	
Variable	Description
Operation_id	The id of the operation
Part_id	The id of part to which the operation belongs
Machine_id	The id of a machine to execute the operation
Tool_id	The id of a cutting tool to execute the operation
TAD_id	The id of a TAD to apply the operation
Machine_list[ ]	The candidate machine list for executing the operation
Tool_list[ ]	The candidate tool list for executing the operation
TAD_list[ ]	The candidate TAD list for applying the operation
Mac_time	The machining time for this operation
Change_time	The change time required for this operation including tool change, set-up change and machine change
Machine_s_time	The start machining time of executing this operation
Machine_e_time	The end machining time of executing this operation
Position	The position value of the operation
Velocity	The velocity value of the operation

Table 2 Class definition of a particle (a solution)

Class Particle: a solution	
Variable	Description
Oper[*n*]	Define a process plan Oper[*n*] based on the above class-ParticleDimension. *n* is the number of operations in the plan
TC	Total Cost of the plan
APC	Additional Penalty Cost of violating constraints in the plan

Table 3 The technical specifications for the part in Group 1 (Li and McMahon 2007).

Parts	*Numbers of Operations (with Numbers of Alternative Machining Plans for Each Operation)*	*Numbers of Constraints*
1	20 (9, 9, 9, 8, 12, 12, 6, 12, 3, 4, 12, 12, 3, 4, 4, 3, 6, 12, 3, 4)	58
2	16 (8, 12, 9, 9, 18, 8, 6, 8, 9, 4, 9, 9, 8, 18, 4, 60)	10
3	14 (9, 9, 36, 16, 36, 24, 27, 36, 24, 6, 8, 8, 6, 8)	51

Table 4 The comparisons of GA, SA and PSO of Makespan for Group 1

Algorithm	Time for 5000 iterations	Robustness (successful optimisation trials out of 20 trials)
GA	19 min 40 sec	20
SA	59 min	14
PSO	7 min 40 sec	20

Table 5 The comparisons of GA, SA and PSO of balance level of machine utilisation for Group 1

Algorithm	Time for 5000 iterations	Robustness (successful optimisation trials out of 20 trials)
GA	16 min 15 sec	20
SA	45 sec	20
PSO	3 min	20

Table 6 The technical specifications for the part in Group 2 (Li and McMahon 2007)

Part	Number of Operations (with Numbers of Alternative Machining Plans for Each Operation)	Number of Constraints
1	7 (9, 9, 27, 8, 8, 9, 36)	11
2	8 (9, 9, 36, 18, 27, 8, 27, 18)	11
3	7 (9, 9, 36, 36, 18, 6, 6)	10
4	9 (9, 9, 27, 6, 36, 36, 6, 18, 18)	18
5	7 (9, 9, 36, 36, 36, 18, 6)	13
6	9 (9, 9, 36, 27, 18, 6, 27, 6, 18)	20
7	5 (9, 27, 27, 18, 9)	5
8	7 (9, 9, 27, 36, 36, 6, 6)	13

Table 7 The comparisons of GA, SA and PSO of Makespan for Group 2

Algorithm	Time for 5000 iterations	Robustness (successful optimisation trials out of 20 trials)
GA	16 min 45 sec	20
SA	45 min	6
PSO	7 min	20

Table 8 The comparisons of GA, SA and PSO of Balance level of machine utlisation for Group 2

Algorithm	Time for 5000 iterations	Robustness (successful optimisation trials out of 20 trials)
GA	16 min 45 sec	20
SA	22 min	6
PSO	7 min 30 sec	20

Table 9 The comparison of GA, SA and PSO algorithms

Algorithm	Population based	Optimisation result (out of 10)	Optimisation speed	Constraints handling	Robustness
GA	Yes	6	Fast but get slow when complexity of problems increases	Adjust Penalty	Robust
SA	No	9	Faster but get slow when complexity of problems increases	Adjust Penalty	Not robust when complexity of problems increases
PSO	Yes	8	Fast	Penalty	Robust

Table 10 Complete time for individual part after optimisation

Part	1	2	3	4	5	6	7	8
Time	2272	1958	2665	2355	2350	2505	2690	2697

Table 11 Machines available time when new order arrives

Machine	1	2	3	4	5
Available time	1000	1120	1078	1116	1000

Table 12 Complete time for individual part after replanning when new order arrives

Part	1	2	3	4	5	6	7	8	9
Time	2221	2513	2051	2659	2660	2689	2344	2654	3458

Table 13 Machines available time when machine 3 breaks down

Machine	1	2	**3**	4	5
Available time	1500	1560	**1800**	1574	1500

Table 14 Complete time for individual part after replanning when machine 3 breaks down

Part	1	2	3	4	5	6	7	8	9
Time	1845	2651	2497	2484	2613	2659	2334	2634	3357

Table 15 Complete time for individual part after first planning

Part	1	2	3	4	5	6	7	8
Time	1313	1914	2399	2242	1775	1488	1123	2095

Table 16 Complete time for individual part after replanning after new order arrives

Part	1	2	3	4	5	6	7	8	9
Time	1415	1738	2038	2226	1710	1572	1123	2539	2546

Table 17 Complete time for individual part after replanning when machine 3 breaks down

Part	1	2	3	4	5	6	7	8	9
Time	1415	1918	2063	2310	1844	1692	1123	2295	2502

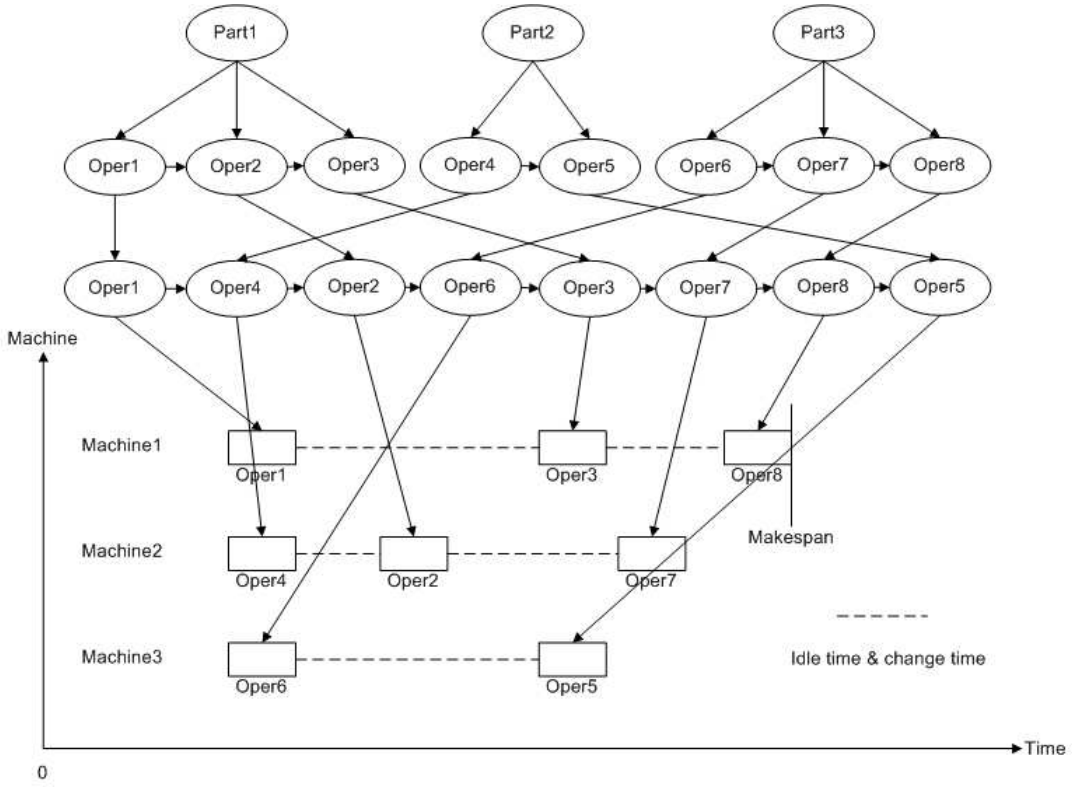

Figure 1 Illustration of the IPPS problem

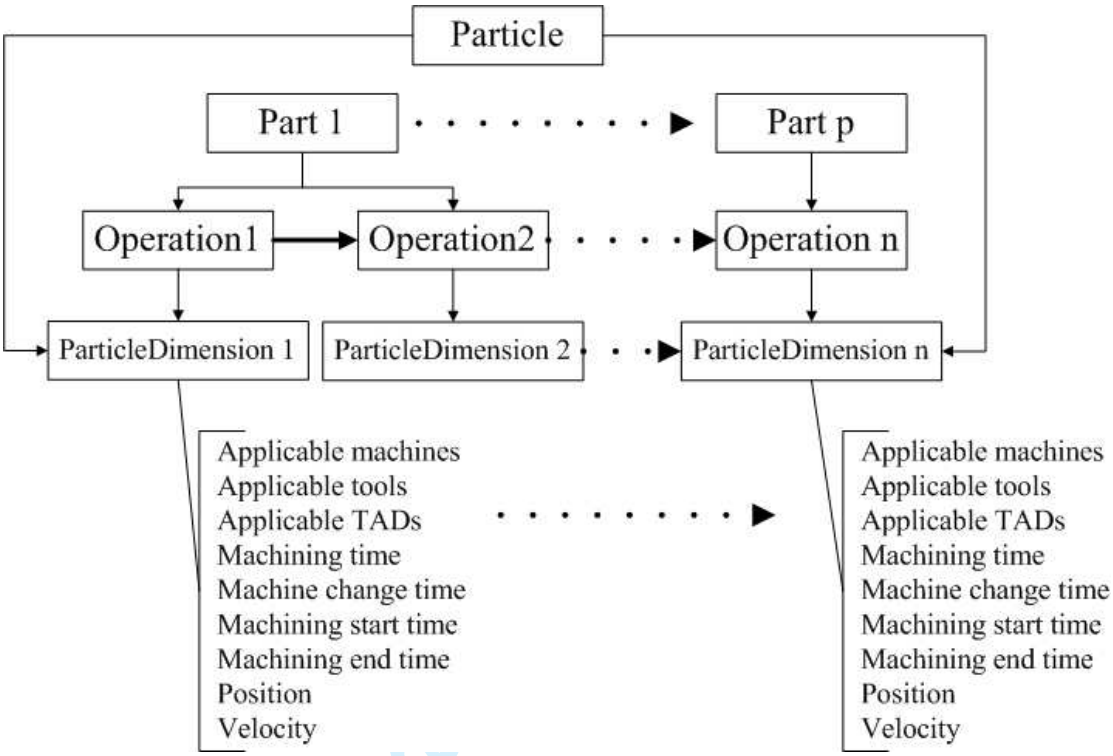

Figure 2 Representation of a solution for IPPS problem

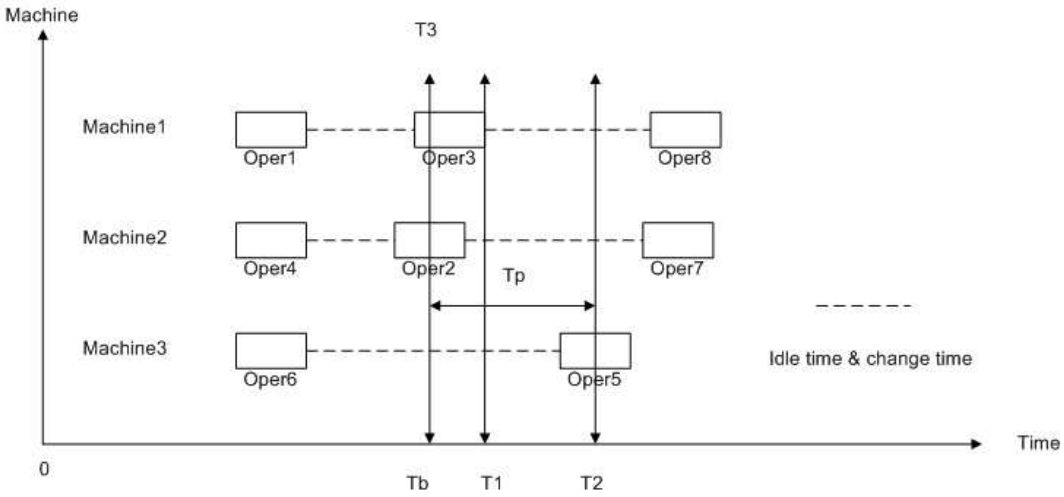

Figure 3 Determination of machines available times when machine2 breaks down

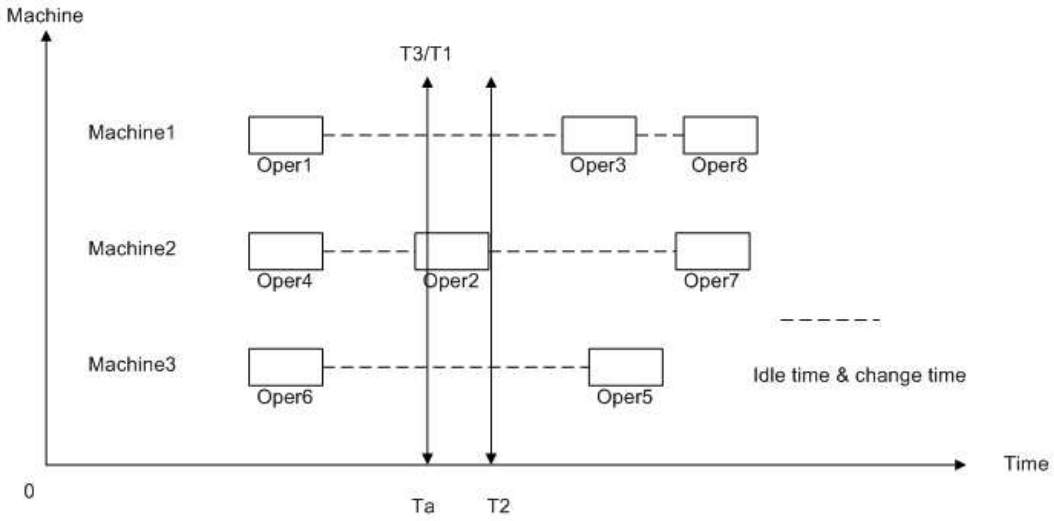

Figure 4 Determination of available times for machines when new order arrives at T_a

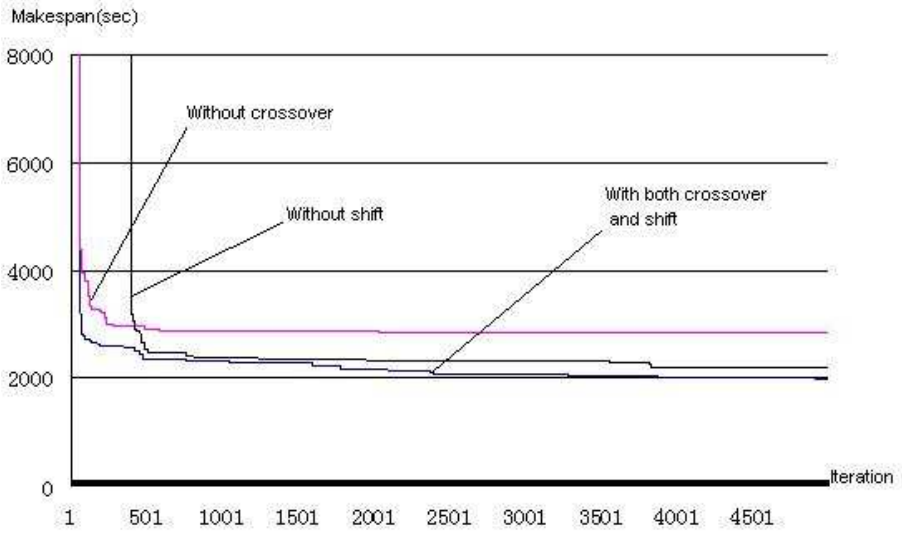

Figure 5 The optimisation results of Makespan for Group 1

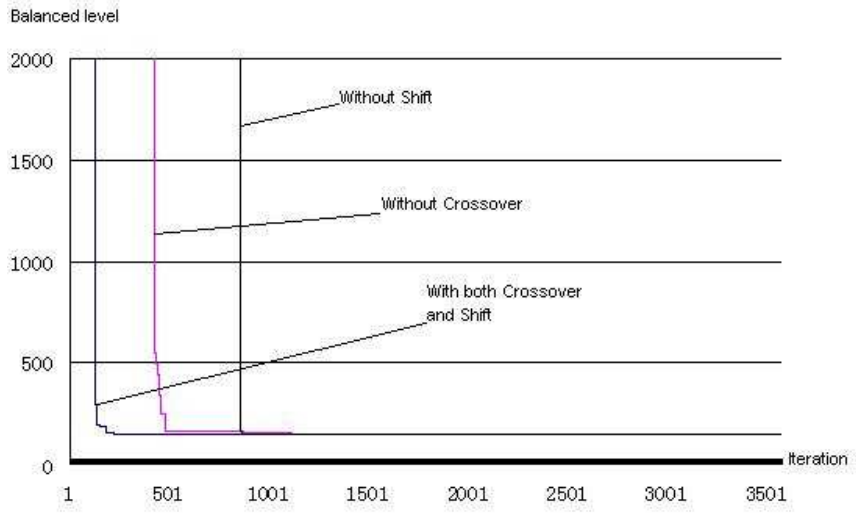

Figure 6 The optimisation results of Balanced Level of Machine Utilisation for Group 1

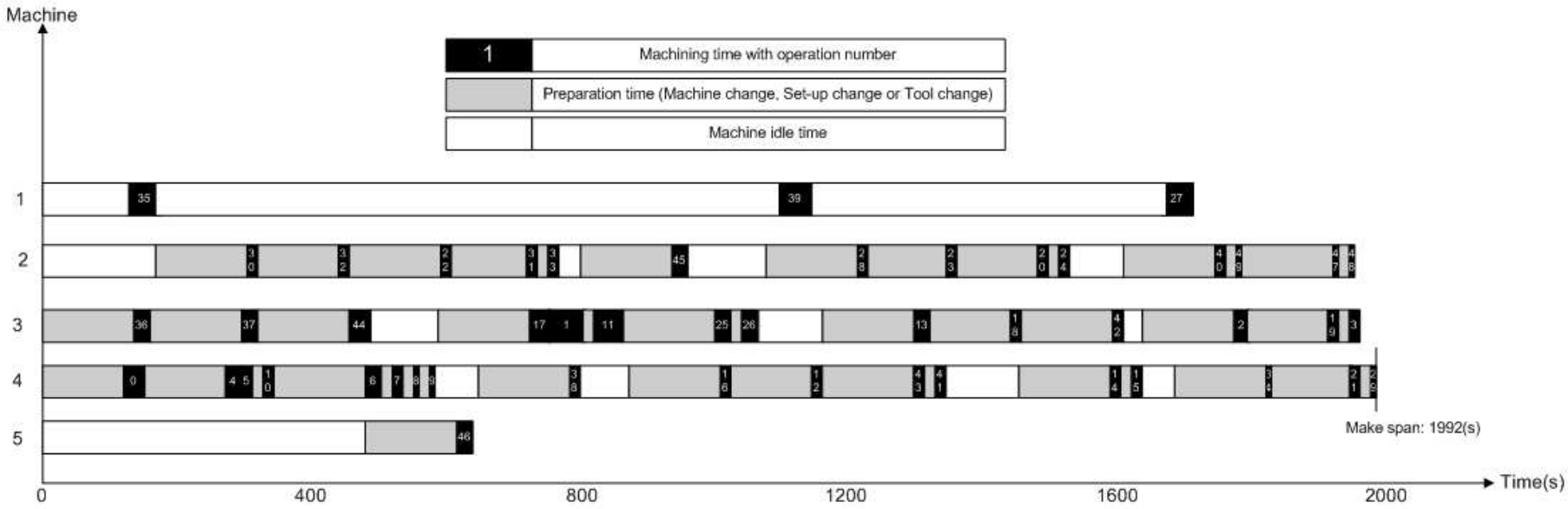

Figure 7 The optimisation result of Makespan for Group 1 in Gantt chart

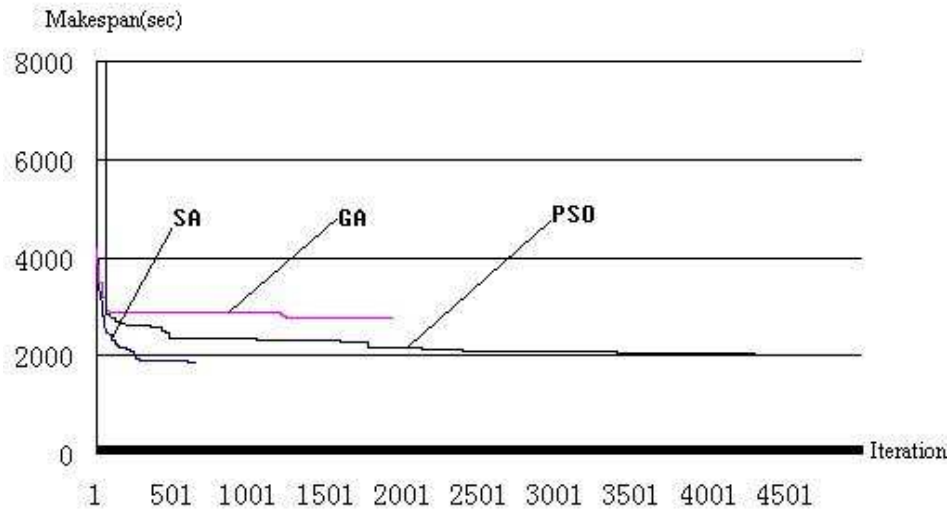

Figure 8 Comparison of GA, SA and PSO of Makespan for Group 1 (8 mins' run)

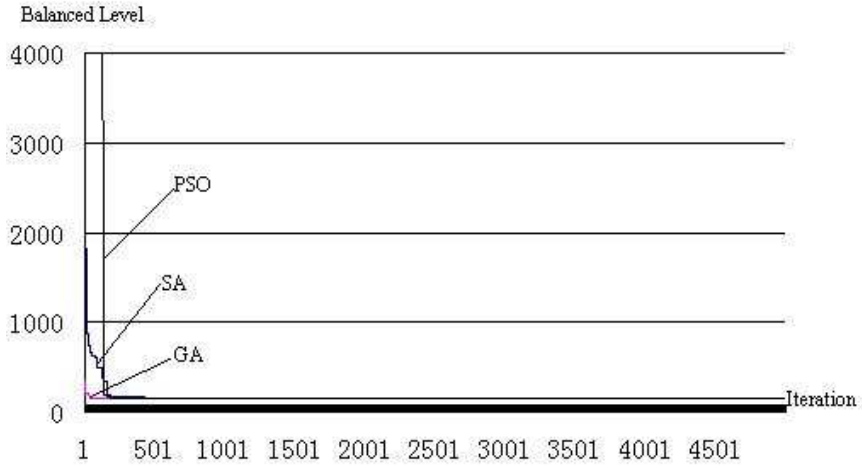

Figure 9 Comparison of GA, SA and PSO of Balanced Level of Machine Utilisation for Group 1

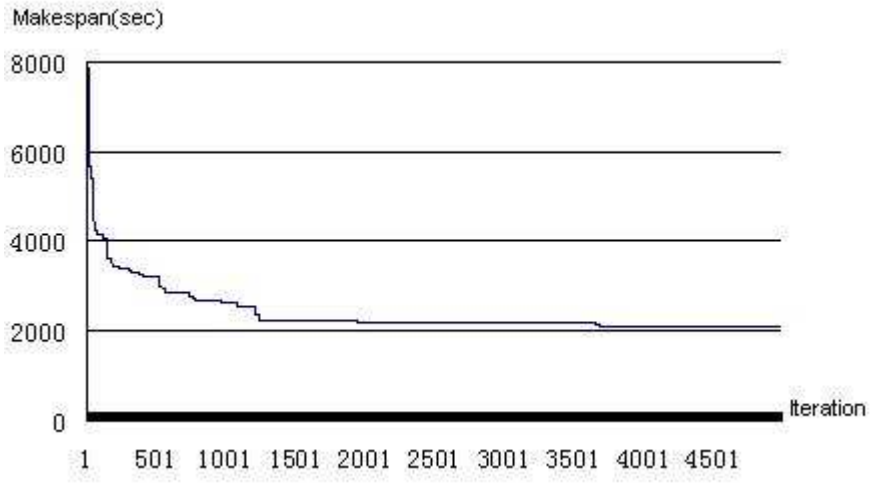

Figure 10 The PSO optimisation result of Makespan for Group 2

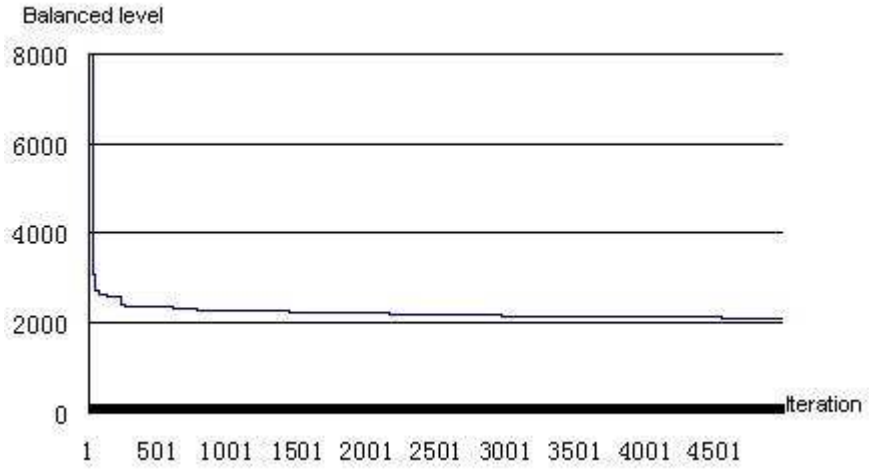

Figure 11 The PSO optimisation result of Balanced Level of Machine Utilisation for Group 2

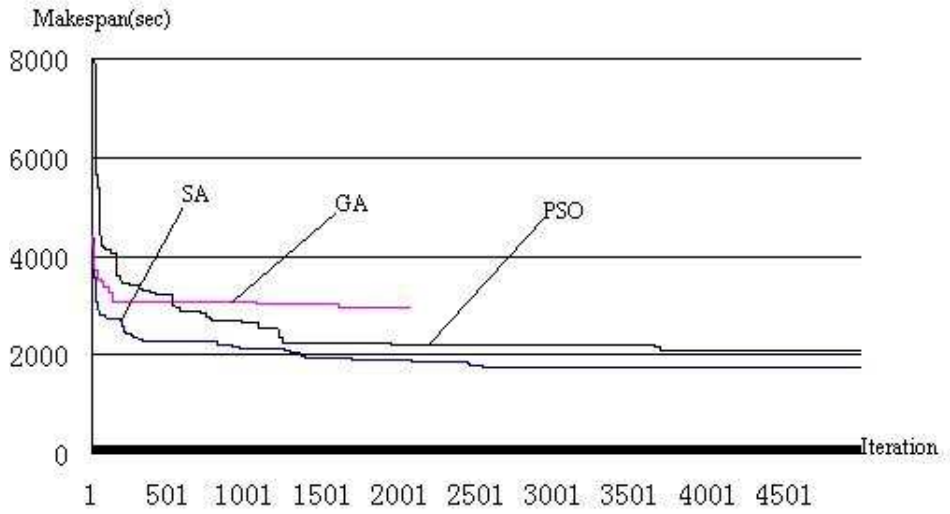

Figure 12 Comparison of PSO, GA and SA of Makespan for Group 2 (in 7 min)

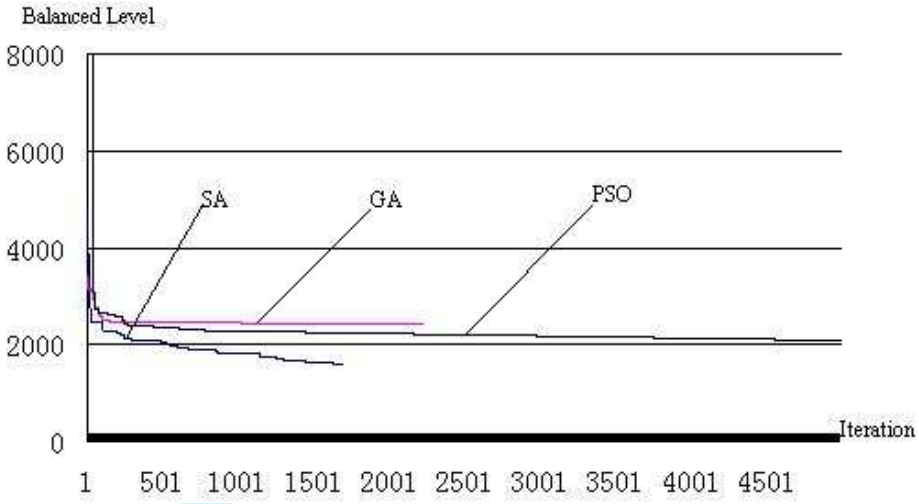

Figure 13 Comparison of PSO, GA and SA of Balanced Level of Machine Utilisation for Group 2

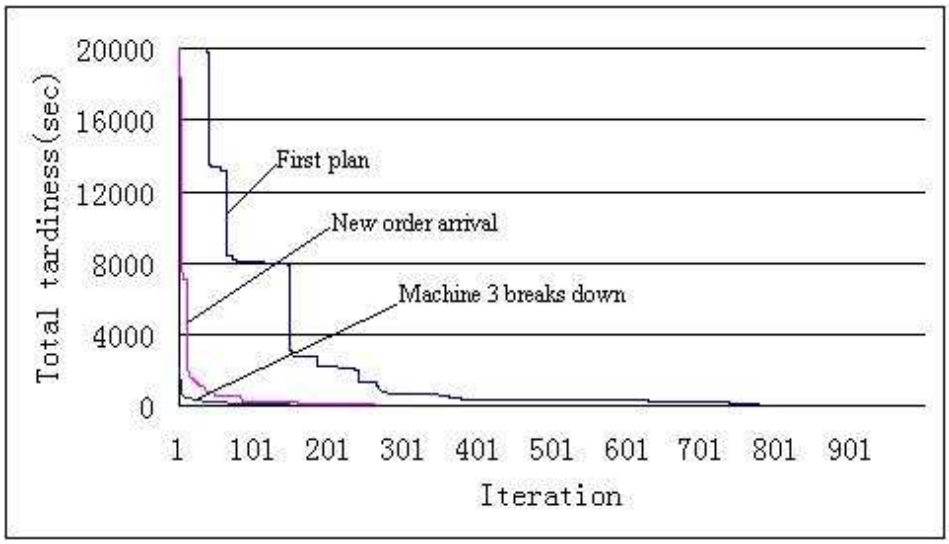

Figure 14 Results of optimisation for first planning, replannings after new order arrival and machine breaks down