Abstract Number: 025-0782

Abstract Title: The Hybrid-PSO approach in production scheduling with parallel machines problem: case in DRAM module industry

Authors: Li-Chih Wang [1], Professor, Department of Industry Engineering and Enterprise Information, Tunghai University, Taiwan

Shu-Fen Li[2*], Ph.D. student, Department of Industry Engineering and Enterprise Information, Tunghai University, Taiwan

Yueh-Ching Jai[3], M.S. student, Department of Industry Engineering and Enterprise Information, Tunghai University, Taiwan

Authors' email: wanglc@thu.edu.tw[1]; fennieli@thu.edu.tw [2*]; j7782727@gmail.com[3]

POMS 23rd Annual Conference

Chicago, Illinois, U.S.A.

April 20 to April 23, 2011

Abstract

This paper proposes a hybrid Particle Swarm Optimization (HPSO) approach, which is combined with variable neighborhood search (VNS) algorithm, to solve a hybrid flow shop scheduling with parallel machine problem. A DRAM module industry case is selected to illustrate the effectiveness of the HPSO model.

Key words: production scheduling, hybrid flow shop scheduling, particle swarm optimization, variable neighborhood search

1. Introduction

The SMT product technology has applied in manufacturing PCB, DRAM

module, and other electronic components. This study deals with the production scheduling problem in DRAM module assembly line, multi-types of DRAM modules are manufactured by multiple identical machines and require a set of components. Production scheduling problems of SMT product line have been extensively studied in recent years, including grouping, sequencing a given set of boards for a machine, and component switching problem. Grouping deals with separating the boards (or several products) to families first, and then allocating to machines, the class of partitioning problems. On the other hand, sequencing a given set of boards for a machine, needs to consider the limitation of magazine capacity since the job sequence will influence the component switching time and the total makespan. Besides, some researches focused on component switching problem. The switching of components, involves removing the unnecessary components and uploading other components, may be necessary at the beginning for each new job. Component switching, a complex problem because of the sequential dependent set-up time, should be planned prudently due to the influence on set-up time of a machine and total makespan.

It is generally accepted that grouping problem and scheduling problem are NP-hard problems, therefore in past researchers solved the above mentioned problems individually. It is easy to find the clustering method for group technology literature (Kusiak, 1990; Islam and Sarker, 2000; Baykasoglu and Gindy, 2000; Mansouri *et al.*, 2000). Van Hop and Tabucanon (2000) solved the grouping problem of electronic component families as a multiple attribute problem with the difficulty of the criteria conflict when one component belongs to a group. Tang and Denardo (1988), Rajkumar and Narendran (1998) proposed the greedy heuristic approaches respectively; the former determined the job sequence in a flexible

manufacturing systems, and the latter dealt with the similarities between boards and set up on magazine. Furthermore, some local search heuristics are used to solve the problem, as tabu search, GA and simulated annealing (Maimon and Braha, 1998; Gronalt *et al.*, 1997; Van Hop and N. Nagarur, 2004). Van Hop and N. Nagarur (2004) proposed the composite genetic algorithm (CGA) to solve the integrated scheduling problem of PCB industry with non-identical parallel machines, and focused on determines job grouping and sequence without considering the job splitting problem. However, solving these three problems separately may result in an inferior solution in minimizing the total makespan due to the interrelation between these problems. The scheduling problem with parallel machines assignment and order sequencing issues is a well-known problem, the hybrid flow shop scheduling (HFS) problem in flexible manufacturing systems environment. Hybrid flow shop scheduling (HFS) was first proposed by Salvador(Salvador, 1972), defined as *n* number of jobs processed in *m* number of production stages, comprises two general issues that namely parallel machines scheduling and flow shop scheduling. The standard HFS production system (be shown in Figure 1) includes the following common features(Rubén Ruiz and Vázquez-Rodríguez, 2010): (1) production stage m contains at least two stages; (2) for all processes in stage k, the number of parallel machines for at least one process must be greater than or equal to two; (3) all the jobs are processed in the same sequence; (4) each job can only be processed on one machine within the same time period. Based on the aforementioned features of HFS environment, the scheduling problem of SMT line in DRAM module may be classified as a HFS problem.

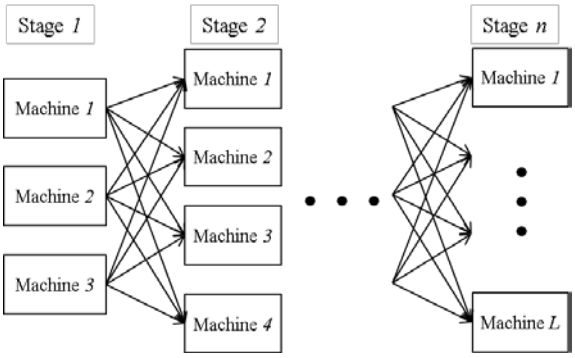

Figure 1. HFS standard structure

Chen and Lee (Chen and Lee, 1999) highlighted that past studies of HFS were commonly based on allocating one job on one machine type. Under the HFS production environment, the number of machine resources for each job can be determined initially. The difficulty is which machine resource should be allocated, and still the production sequence has to be decided. In recent years, scholars have dedicated significant effort to solving the HFS production scheduling problem. Though this problem can be solved using optimization methods, the study by Garey and Johnson proved that minimizing the makespan scheduling under HFS is an NP-complete problem (Garey and Johnson, 1979). The number of machines can be dynamically adjusted to complete the job within the shortest possible time. Since, the production quantity of each job needs to be splitted into small sublots to complete in a short time. Then various similar sublots can be processed simultaneously on a number of machines within the same stage, thereby reducing the production time.

Previously, little research was spent on solving the problem of multiprocessing in job splitting. Recently, Quadt and Kuhn (2007) did propose the use of nested GA, which divides a job into several lots of the same size, enabling a number of products to be distributed to several sets of identical parallel machines for simultaneous processing. However, this method omitted two crucial factors, namely the setting up time and job delivery time, which are key factors affecting the scheduling of DRAM

module industry. Therefore, this study employs metaheuristic to obtain the approximate optimal solution under the production characteristics of DRAM module industry.

This study aims to minimize the total makespan of DRAM module production by simultaneously determining the job production sequence, number of sublots, and the machines that these sublots should be assigned. More specifically, the characteristics of parallel processing, dedicated machines, sequence-independent setup time, and sequence-dependent setup time will be considered in planning the DRAM production schedule. This study will simultaneously generate the optimal job sequence, number of sublots and parallel-machine scheduling in each stage for achieving the minimization of the total makespan. Due the computational complexity of the model, a hybrid approach based on the variable neighborhood search and particle swarm optimization (Hybrid-PSO) was proposed to design to obtain the near-optimal scheduling configuration. Our preliminary computational study shows that the developed VNPSO not only provides good quality solutions within a reasonable amount of time but also outperforms the classic branch and bound method.

The rest of the paper is organized as follows: Section 2 reviews related works in the literature regarding variable neighborhood search and particle swarm optimization; Section 3 defines the multi-stage and parallel-machine scheduling problem in DRAM module industry and formulates a mixed integer programming model; due to the computational complexity, Section 4 employs a Hybrid-PSO algorithm to obtain the near-optimal solution; Section 5 addresses the good performance of the Hybrid-PSO algorithm through the computational study and empirical data from real industries and Section 6 finally presents the concluding

remarks.

2. Literature review (background methods)

2.1 Variable Neighborhood Search (VNS)

Variable neighborhood search (VNS) is an algorithm developed by Mladenovi'c and Hansen(Mladenovi´c N and P, 1997), which involves the constant interchange between different neighborhood search methods to avoid the regional local solution space when finding an optimal solution. In an example of minimizing the makespan during job production sequence decisions, VNS uses shaking approach to generate the *x* variable. VNS was followed with a localized search, beginning at the first search area (QN_1 (x)) and constantly spreading. VNS demonstrated its ability to avoid the regional local makespan solution. Chen and Chen (Chen and Chen, 2009) proposed a hybrid variable neighbourhood descent (VND) combines with TS to solve the non-equivalent parallel machine and sequence-dependent setup time. Behnamian *et al.* (Behnamian *et al.*, 2009b)proposed the use of VNS with ant colony optimization (ACO) and a TS hybrid algorithm to obtain the parallel machine under a sequence-dependent setup time consideration. Behnamian et at. (Behnamian *et al.*, 2009a) proposed the use of VNS with an SA hybrid algorithm to solve the practical process-type production scheduling problems. Jarboui *et al.* (Jarboui *et al.*, 2009) proposed the use of the estimation of distribution algorithm (EDA) through VNS, with minimizing the total process time as the objective, to solve process-based production scheduling problems. The quality of the solutions derived from their results was no worse than those derived from ACO, GA, and particle swarm optimization (PSO). This study also uses EDA as one of its solution combinations. Rahimi-Vahed *et al.* (Rahimi-Vahed *et al.*, 2009) used the VNS to search for the Pareto optimal solution in process-based production scheduling problems. Therefore,

the algorithm proposed in this study was combined with VNS during the solution process used to search for the optimal job sequence to solve the problem under HFS.

2.2 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart (Kennedy and Eberhart, 1995) in 1995. The theory was based on the population dynamics-based optimization method, and the concept originated from research on the group behavior of animals (birds). In a community-based group, the behavior of an individual is not only affected by the individual's past experience and knowledge, but also by the impact of society overall.

During the PSO search process, the memory of each particle can record the optimal solution location of its search. Each particle has its own optimal solution position. The velocity of each particle is based on the current flight speed of the particle itself, while the difference in optimal solution between the current particles and the current individual body is based on the best experience of the individual body. Additionally, the difference in the optimal solution between current particles and the current group is due to the three factors of the best group experience learning to adjust their own speed to determine the next step in particle velocity and direction. Then, the updated velocity is used to adjust the current position to update the search distance and direction of each particle.

Zhang and Sun (Zhang and Sun, 2009)proposed a new crossover method called a reversal crossover operation, which differs from the GA-based crossover operator. Their crossover operator uses two crossover rules to solve process-type production scheduling problems. Kashan and Karimi (Kashan and Karimi, 2009) proposed an operator with a new speed and position that uses a calculus operator to solve the scheduling of parallel machines. Anghinolfi and Paolucci (Anghinolfi and Paolucci,

2009) redefined the rules for computing speed, location, and operators used in single machine scheduling problems under independent setup time considerations.

In this study of parallel-machine configuration problems, the solution algorithms belong to numerical PSO. In previous numerical optimization studies, Kou *et al.*(Kou *et al.*, 2009) added the concept of synergy and collaboration to the PSO algorithm process to solve nonlinear optimization problems. Sun *et al.* (Sun *et al.*, 2011a) proposed a modified version of PSO against the PSO algorithm to strengthen its local search capabilities. The modified algorithm was primarily used to solve nonlinear optimization problems. Chuang *et al.* (Chuang *et al.*, 2011) combined ChaoticPSO(Coelho and Mariani, 2009) and CatfishPSO (Chuang *et al.*, 2008a) to solve numerical optimization problems; the subsequent solutions generated relatively good results. Thus, the algorithm is applied in this study.

In previous studies, PSO was found in different production environments for a wide range of scheduling applications, and used extensively in numerous application fields because: (1) the search is distributed; (2) only a small number of parameters require adjustment; and (3) convergence speed to the optimal solution can be increased. Additionally, the results or problem-solving speed are relatively satisfactory. Therefore, the algorithm proposed in this study is combined with the PSO algorithm to comprise the solution process used to search for the optimal parallel-machine configuration to solve the HFS problem.

3. Hybrid flow shop scheduling for memory module manufacturing

3.1 Problem Description

Figure 2 illustrates the manufacturing process of memory module. The manufacturing of memory module comprises five processes, and each process has its practical characteristics that influence the production schedule. Therefore, these production characteristics should be included in the production scheduling. The

characteristics are detailed below:

(a) Re-entrant flow process: Memory module are basically classified into two types: single-sided boards and double-sided boards. Manufacturing machines of both types are the same and the difference lies in the SMT stage. Double-sided board needs to go through the SMT stage twice which impacts the subsequent scheduling method in the job production process.

(b) Parallel machine processing: Identical parallel machines are used in the manufacturing process of memory module. When the job demand is high, a job must be allocated to more than one machine in order to increase the capacity and reduce the makespan required to complete the job.

(c) Sequence dependent setup time: Van Hop and N. Nagarur (2004) solved the scheduling problem with minimal total maskespan, it is the same as minimizing the maximum number of component switches. The board sequence for each machine referred to as sequence dependent setup time. In general, SMT stage is the bottleneck of memory module manufacturing process. Therefore, setting the optimal production sequence to reduce the number of setups and shortening the overall completion time is the key focus of diminish this restriction.

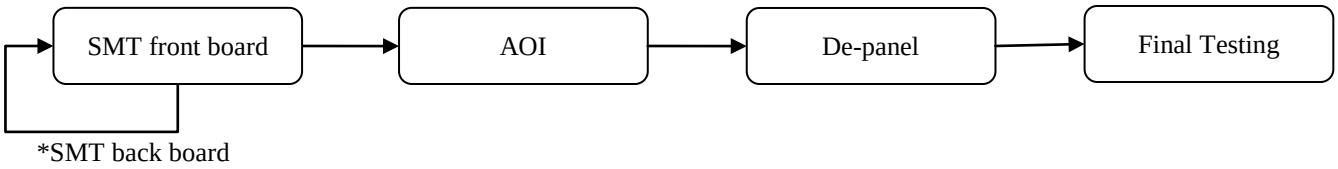

Figure 2. Manufacturing process of memory modules

3.2 Mathematical model

In this section, we formulate the mixed integer programming model for the addressed HFS scheduling problem in the memory module manufacturing, which is based on the standard form of the HFS problem. The assumptions and notations are

defined as follows.

3.2.1 Assumptions

1. All jobs and machines are available at time zero.
2. Machines at a given stage are identical.
3. A machine can process only one operation at a time.
4. Preemption is not allowed.
5. The capacity of buffers between stages is unlimited.
6. The demand of each work order is a multiple of the number of machines at each stage.
7. The problem data is deterministic and known in advance.

3.2.2 Notation definition

A. indexes:

i, u = work order index, i, u=1,2,…, I.

j = processing stage index, j=1,2,.., J.

k = machine index, k=1,2,…, M_j, M_j is the number of machines at stage j.

l = the index of the lot splitting, l=1,2,…, L_j, L_j is the number of ways of lot splitting at stage j.

e= product type index, e=1, 2,..,E

B. Parameters:

d_i = the processing quantity demanded for work order i.

$p_{i,j}$ = the unit processing time for work order i at stage j.

$r_{j,l}$ = the production ratio factor at stage j when applying the lot-splitting way l.

s_i^{SI} = the sequence independent setup time for work order i.

bi j= the required number of the product type for work order *i* at stage j.

$f_{i'}$ = the component quantity of front and back board for order *i'*

$s^{SD}_{e,e'}$ = the product component quantity different between front and back board

di'= the required number of stage 1 for work order *i'*

pi',1= the work time of stage 1 for work order *i'*

$s^{SI}_{i',1}$ = the sequence dependent setup time of stage 1 for work order *i*

bi',1 : the product type of stage 1 for work order *i'*

oj: the batch of stage j

M = a large enough number.

C. Decision variables:

C_{max} = the makespan.

$C_{i,j,k}$ = the completion time of work order *i* in machine *k* at stage *j*.

$C_{i',1,k}$ = the completion time of work order *i'* in machine *k* at stage *1*.

$X_{i,j,k}$ = the production ratio of work order *i* in machine *k* at stage *j*.

$X_{i',1,k}$ = the production ratio of work order *i'* in machine *k* at stage *1*.

$Y_{i,j,k}$ = binary variable, $Y_{i,j,k}$=1, if work order *i* is processed in machine *k* at stage *j*; $Y_{i,j,k}$=0, otherwise.

$Y_{i',1,k}$ = binary variable, $Y_{i',1,k}$=1, if work order *i'* is processed in machine *k* at stage *1*; $Y_{i',1,k}$=0, otherwise.

$W_{i,j,k,l}$ = binary variable, $W_{i,j,k,l}$=1, if work order *i* is processed in machine *k* at stage *j* with applying the lot-splitting way *l*; $W_{i,j,k,l}$=0, otherwise.

$W_{i',1,k,l}$ = binary variable, $W_{i',1,k,l}$=1, if work order *i'* is processed in machine *k* at stage *1* with applying the lot-splitting way *l*; $W_{i',1,k,l}$=0, otherwise.

$S_{i,u,j,k}$ = binary variable, $S_{i,u,j,k}$=1, if work order *i* is processed in machine *k* at stage *j*

before work order u; $S_{i,u,j,k}$=0, otherwise.

$S_{i',u,1,k}$ = binary variable, $S_{i',u,1,k}$=1, if work order i' is processed in machine k at stage *1* before work order u; $S_{i',u,1,k}$=0, otherwise.

3.2.3 Mixed integer linear programming model

A mixed integer linear programming model is formulated for the addressed HFS scheduling problem in the memory module manufacturing. The objective function in Eq.(1) is to minimize the makespan of the schedule which is equal to the completion time of the last sublot processed in the system.

Objective function:

$$\text{Minimize } Z = C_{max} \tag{1}$$

$$C_{max} \geq C_{i,J,k} \quad \forall i,k \tag{2}$$

$$C_{i',j-1,k} \geq d_{i'} \times X_{i',j,k} \times p_{i',j} + C_{i',j\text{-}1,k} \quad \forall i',\ j,\ k;\ j = 1 \tag{3-1}$$

$$C_{i,j-1,k} + \left[d_i \times X_{i,j,k} \div q_j \right] \times p_{i,j} \leq C_{i,j,k} \ \forall i,\ j,\ k;\ j=2,\ 3,\ 4 \tag{3-2}$$

$$X_{i',j,k} = \sum_{l=1}^{L_j} r_{j,l} \times W_{i',j,k,l} \quad \forall i',\ j,\ k;\ j = 1 \tag{4-1}$$

$$X_{i,j,k} = \sum_{l=1}^{L_j} r_{j,l} \times W_{i,j,k,l} \quad \forall i,\ j,\ k;\ j = 2,\ 3,\ 4 \tag{4-2}$$

$$\sum_{l=1}^{L_j} W_{i',j,k,l} \leq 1 \quad \forall i',\ j,\ k;\ j = 1 \tag{5-1}$$

$$\sum_{l=1}^{L_j} W_{i,j,k,l} \leq 1 \quad \forall i,\ j,\ k;\ j = 2,\ 3,\ 4 \tag{5-2}$$

$$\sum_{k=1}^{M_j} X_{i',j,k} = \begin{cases} 1, & \text{if } b_{i',j} = 1 \\ 0, & \text{if } b_{i',j} = 0 \end{cases} \quad \forall i',\ j;\ j=1 \tag{6-1}$$

$$\sum_{k=1}^{M_j} X_{i,j,k} = \begin{cases} 1, & \text{if } b_{i,j} = 1 \\ 0, & \text{if } b_{i,j} = 0 \end{cases} \quad \forall i,\ j;\ j=2,\ 3,\ 4 \tag{6-2}$$

$$\begin{aligned} C_{u',j,k} \geq C_{i',j,k} + d_{u'} \times X_{u',j,k} \times p_{u',j} + \left(s_{u',j}^{SI} + s_{e,e'}^{SD}\right) \times Y_{u',j,k} \times b_{u',j} \\ - M(2 - Y_{i',j,k} - Y_{u',j,k}) - M(1 - S_{i',u',j,k}) \quad \forall i', u', j, k;\ i' \neq u';\ j = 1 \end{aligned} \tag{7}$$

$$\begin{aligned} C_{i',j,k} \geq C_{u',j,k} + d_{i'} \times X_{i',j,k} \times p_{i',j} + \left(s_{u',j}^{SI} + s_{e,e'}^{SD}\right) \times Y_{i',j,k} \times b_{i',j} \\ - M(2 - Y_{i',j,k} - Y_{u',j,k}) - M \times S_{i',u',j,k} \quad \forall i', u', j, k;\ i' < u';\ u' = i + I\ ;\ j = 1 \end{aligned} \tag{8}$$

$$\begin{aligned} C_{u',1,n} \geq C_{i',1,k} + d_{u'} \times X_{u',1,n} \times p_{u',1} + \left(s_{u',1}^{SI} + s_{e,e'}^{SD}\right) \times Y_{u',1,n} \times b_{u',1} \\ \forall i', u', k, n;\ i' = 1, 2, ..., I; u' = i' + I \end{aligned} \tag{9}$$

$$C_{i,1,k} = C_{u',1,k} \quad \forall i, k; u' = i + I \tag{10}$$

$$\begin{aligned} C_{u,j,k} \geq C_{i,j,k} + \left[d_u \times X_{u,j,k} \div q_j \right] \times p_{u,j} + s_{u,j}^{SI} \times Y_{u,j,k} \times b_{u,j} \\ - M(2 - Y_{i,j,k} - Y_{u,j,k}) - M(1 - S_{i,u,j,k}) \quad \forall i, u, j, k;\ i \neq u;\ j = 2,\ 3,\ 4 \end{aligned} \tag{11}$$

$$\begin{aligned} C_{i,j,k} \geq C_{u,j,k} + \left[d_i \times X_{i,j,k} \div q_j \right] \times p_{i,j} + s_{i,j}^{SI} \times Y_{i,j,k} \times b_{u,j} \\ - M(2 - Y_{i,j,k} - Y_{u,j,k}) - M \times S_{i,u,j,k} \quad \forall i, u, j, k;\ i < u;\ j = 2,\ 3,\ 4 \end{aligned} \tag{12}$$

$$X_{i',j,k} \leq M \times Y_{i',1,k} \quad \forall i', k \tag{16}$$

$$X_{i,j,k} \leq M \times Y_{i,j,k} \quad \forall i,\ j, k;\ j = 2,\ 3,\ 4 \tag{17}$$

$$Y_{i,j,k} \in \{0,1\} \ ,\ Y_{i',1,k} \in \{0,1\} \quad \forall i,\ i',\ j,\ k \tag{18}$$

$$W_{i,j,k,l} \in \{0,1\} \ , W_{i',1,k,l} \in \{0,1\} \quad \forall i,\ i',\ j,\ k,\ l \tag{19}$$

$$S_{i,u,j,k} \in \{0,1\} \ , S_{i',u',1,k} \in \{0,1\} \quad \forall i,\ i',\ u,\ u',\ j,\ k \tag{20}$$

$$X_{i,j,k} \geq 0 \ , \ X_{i',1,k} \geq 0 \quad \forall i, i', j, k \tag{21}$$

$$C_{i,j,k} \geq 0 \text{ (integers)} \ , \ C_{i',1,k} \geq 0 \text{ (integers)} \quad \forall i, i', j, k \tag{22}$$

Eq.(2) state that the makespan of the schedule. Eq.(3-1) and Eq.(3-2) state that the completion time for work order *i* in machine *k* at stage *j* is greater or equal to the completion time on the preceding stage *j-1* plus the processing time on the current

stage *j*. Eq.(4-1) and Eq.(4-2) determine the production ratio ($X_{i',j,k}$ and $X_{i,j,k}$). Because the number of processing machines at every stage is given in advance, all possible ways of lot-splitting at any stage are also known. Eq.(5-1) and Eq.(5-2) ensures that the production ratio ($X_{i',j,k}$ and $X_{i,j,k}$) only equals one of all possible production ratio factors at any stage or zero. Eq.(6-1) and Eq.(6-2) state that the sum of the production ratio ($X_{i',j,k}$ and $X_{i,j,k}$) of the order *i* in each machine *k* at stage *j* equals 1. Eq.(7) and Eq.(8) is enforced to guarantee that work order *u'* will begin after completing the work order *i'* of stage 1. Eq.(9) ensures that same order back board never be process before front board. Eq.(10) state that order *u'* equal to the completion time on stage 1. Eq.(14) and Eq.(15) are similar to Eq.(9) and Eq.(10) except for adding the sequencing dependent setup time between two work orders in the stage 2, 3 and 4. In Eq.(16) and Eq.(17) is shown whether the work order *i* and *i'* is processed in machine *k* at stage *j*. $Y_{i,j,k}$ equals 1 if work order *i* is processed in machine *k* at stage *j* ($X_{i,j,k} > 0$); otherwise ($X_{i,j,k} = 0$), $Y_{i,j,k}$ equals 0. Eq.(18)–Eq.(22) are the basic restrictions on the decision variables.

4. An integrated VNS and PSO algorithm

In this section, a two-phase integrated variable neighborhood search (VNS) and particle swarm optimization (PSO) algorithm, named HSPSO algorithm, is developed. In the first phase, VNS is used to decide the processing sequence of orders and in the second phase, PSO is used to determine the machine allocation for all manufacturing orders at each stage.

The algorithmic procedures of the proposed HPSO algorithm in this paper are demonstrated in Figure 1. In the first phase, the VNS algorithm is adopted to decide the new production sequence of orders. Next, the new processing sequence of orders

and the current optimal machine allocation for orders recorded by VNS, are inputted into the PSO algorithm in the second phase. Based on the input processing sequence of orders, PSO searches for an optimal machine allocation which minimizes the makespan. Afterwards, the optimal value of makespan and the machine allocating result are feedbacked to VNS, consequently, VNS generates a new processing sequence of orders again, and a solution of minimal makespan is found out once again through PSO. The process is kept going till the termination criterion is met. Finally, the optimal processing sequence of orders, machine allocations for each order and lot size will be obtained.

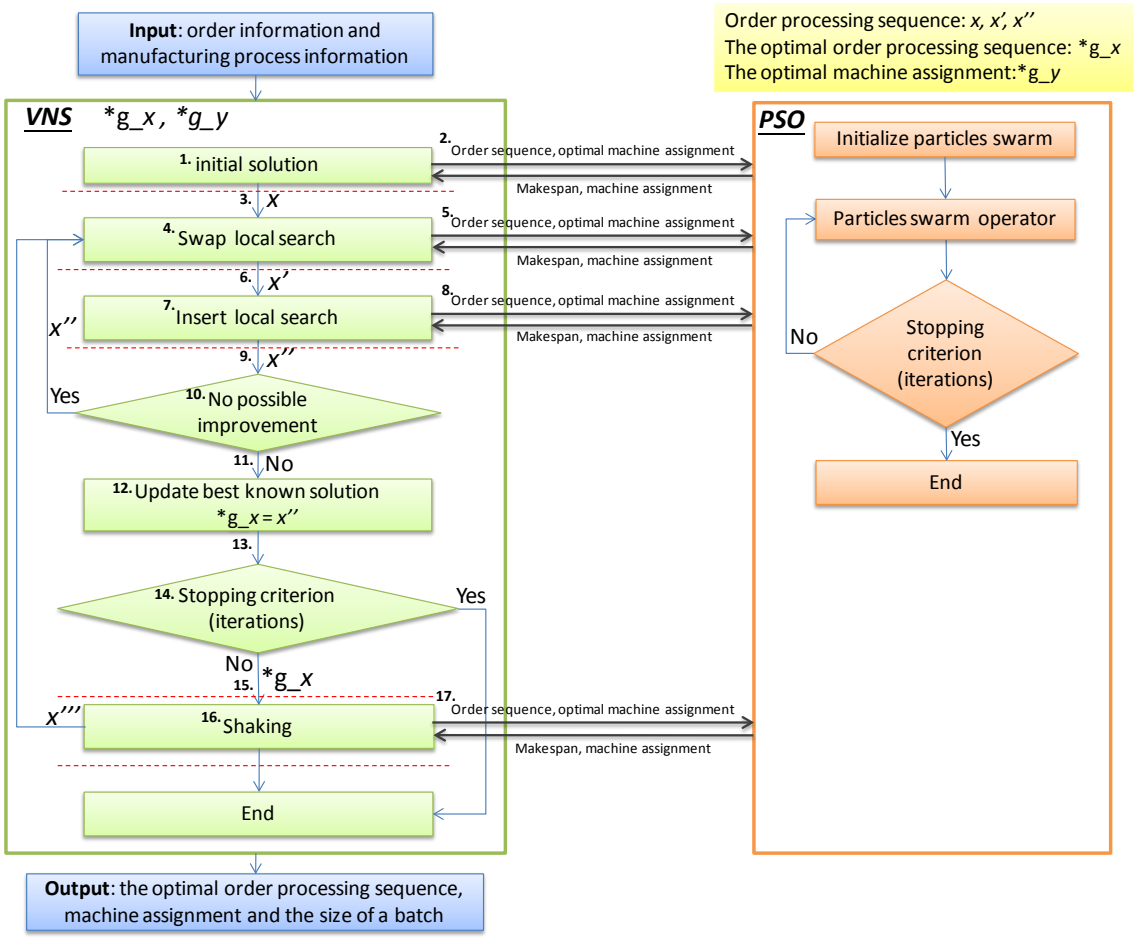

Figure 3 The algorithmic procedures of the proposed VNPSO algorithm

The first step of VNS is to confirm order information and manufacturing process information. Second step is to generate an initial order processing sequence. The SPT

(shortest processing time) rule is adopted as the generating rule. Step 3 is to Generate a neighborhood solution through “Swap local search”. A neighborhood solution is gained through “swap local search” mechanism. Step 4 is to generate a neighborhood solution through “Insert local search”. Insert the designated order into the priority sequence position through the “Insert local search” mechanism and a neighborhood solution is accordingly gained. Step 5 is to decide whether the criterion to terminate the local search is met. When the current processing sequence of orders is the same as the one got through “Swap local search” and “Insert local search”, then move to the next step, otherwise return to step 3. Step 6 is to update the best solution. If the makespan of the processing sequence of orders got through local search is less than the current best solution, update the best value. Step 7 is to decide whether the terminating criterion of the HPSO algorithm is met. The last step is the “Shaking” mechanism. Through the “shaking” mechanism, the current best processing sequence of orders is disturbed in order to gain different region solutions and avoid falling into local optimum. The way to deal with it is to randomly generate two swap points; they are respectively i and j, with which the orders in the position i and j for the current optimal processing sequence of orders are exchanged. That is to say, a new order processing sequence is generated. And then return to step 3.

After inputting the processing sequence of orders and the optimal machine allocation recorded by the current VNS, PSO algorithm is employed to search and decide the number of allocated machines for all manufacturing orders in each stage, and the fitness is makespan. The first step is to generate an initial population. The second step is to initialize the value of Cr. Step 3 is to Update the inertia weight. Step 4 is to Calculate the fitness value. Based on the known processing sequence of orders and machine allocation, this step precedes the forward capacity allocation to

calculate the fitness values (makespan) of all the particles. Step 5 is to Update particle best (pBset). The pBest is the best position of each particle in its own searching process. During the iterations, the particle's fitness evaluation is compared with pBest. If the current value is better than pBest, then set pBest value equal to the current value. Step 6 is to update global best (gBest). Compare fitness evaluation with the population's overall previous best, gBest. If the current value is better than gBest, then update the current particle's value to gBest. Step 7 is to update Cr, velocity and position of the particle. Step 8 is to determine whether the same optimal solution of the population which iterates n times exists If yes, execute Step 9 and proceed the boundary search. If not, skip to Step 10. Step 9 is boundary search. The boundary search aims to prevent that the current solution falls into the local optimum and enable it to avoid being a regional solution, and in turn to find the global optimum. The boundary search is to generate new particles for each dimension by random. The amount is 10% of the population, with which displace the worst 10% of the original population. Step 10 is to decide wither the designated times of iteration are reached. The termination criterion of the PSO algorithm proposed in this paper is that when the number of iteration exceeds the designated maximum iteration times, terminate the algorithm. If it is not reached, return to Step 3.

5. Computational study

The purposes for computational studies in this paper is to test and compare performances of the developed HPSO algorithm with the traditional approaches, one is branch-and-bound (B&B) algorithm which is frequently used to solve the MILP scheduling model and the other is traditional PSO method.

Six samples of different sizes shown in Table 1 are collected and modified from the memory module manufacturing industry for this computational study. The MILP

model of hybrid flow shop (HFS) scheduling is formulated by using the IBM ILOG CPLEX optimization software package and solved by traditional B&B algorithms. The proposed HPSO algorithm is implemented by Visual C++ programming language. A personal computer with an Intel® Core(TM) i7-2600K 3.40GHz 3.39GHz and 3.42 GB RAM is used to execute and test the algorithms. In order to determine the suitable control parameters of the HPSO algorithm, we conducted the full factorial design of experiments to identify the optimal settings for the control parameters of the proposed HPSO algorithm. The best settings of PSO control parameters are found in Table 2 and they will be used in the following computational study.

Table 1. Ten testing samples with different sizes

Data Size	*Problem Number*	*Problem size (product number)*	*No. of continuous variables*	*No. of binary variables*	*No. of integer variables*	*No. of total variables*	*No. of constraints*
Small	1	3	45	252	43	340	1196
	2	5	75	525	91	691	2418
Large	3	10	150	1575	181	1905	9675
	4	20	300	5250	361	5911	30722
	5	30	450	11025	541	12016	69226
	6	50	750	29025	901	30676	176729

Table 2. The best settings of HPSO control parameters

The HPSO parameters	Value
Population size	200
C_1	2
C_2	2

Two criteria are used to measure and evaluate the effectiveness of the developed HPSO algorithm. First is a solution gap between the optimal solution of the MILP scheduling model and the best solution found from the HPSO algorithm. The optimal solution of the MILP model is solved by the classic B&B algorithm. But, for large

size problems, the MILP scheduling model cannot find optimal solutions because the CPLEX optimizer runs out of memory. At this time, we use the best CPLEX solution found before the B&B procedure was terminated to estimate and represent the optimal solution. Therefore, the solution gap of large size problems can also be calculated. In addition, due to stochastic nature of the HPSO algorithm, we calculate minimal, average and maximal solution gaps of each problem by ten replications using different random seeds. A solution gap is defined as follows.

$$Solution\ Gap\ (\%) = \frac{S - B}{B} \times 100\% \tag{23}$$

where B = the optimal solution obtained from the B&B algorithm using CPLEX optimizer, if CPLEX runs out of memory, this value is the best CPLEX solution found before the B&B procedure was terminated; S = the best solution of the HPSO algorithm.

The second criterion is the CPU computational time which measures the computation speeds of the two algorithms being compared. But the MILP model in large size problems cannot find the optimal solution because CPLEX optimizer runs out of memory. We kept the duration of time that the system is out of memory and used it to represent the estimated CPU time. Thus, the actual CPU time is greater than the estimated CPU time. Similarly, for the HPSO algorithm, minimal, average and maximal CPU times of each problem are calculated from ten replications.

5.1 Performance comparison in objective value and solution gap

We compared the objective value and solution gap (%) between the developed HPSO algorithm and the B&B algorithm. The minimal, average, maximal solution gaps and the objective value of all solved problems are shown in Table 3. The row with symbol * means that its optimal solution computed by the MILP model is an estimated value since CPLEX optimizer runs out of memory. From this table, we can

observe that the value of average solution gaps between these two algorithms equals zero in problem # 1. This shows that the proposed HPSO algorithm can find the optimal solution.

Moreover, the value of average solution gaps between these two algorithms is "negative" from problem # 2 and problem # 3. It means that the solution provided by the HPSO algorithm is better than the one generated by the terminated B&B algorithm. In these samples, the B&B algorithm runs out of the memory and has little distance with the optimal solution. The HPSO algorithm, on the other hand, can still find the near-optimal solution and even outperform the terminated B&B algorithm.

In the large samples (from problem #3 to problem #6), we further observed that the B&B algorithm not only spends more time but cannot obtain any feasible solutions. The developed HPSO algorithm still generates better solutions for the large samples in the reasonable time. Consequently, from above analysis, our results claim that the proposed HPSO algorithm not only provides the near-optimal solutions irrespective of the size of the sample data; it also generates the better solutions for large scale samples in which B&B algorithm cannot found any feasible solutions.

Table 3. The objective value and solution gap (%) between B&B and VNPSO

Data Size	*Problem Number*	*Problem size (product number)*	*CPLEX Optimizer (B&B)*	*The VNPSO Algorithm*		
			Solution	*Min. Solution (Solution Gap %)*	*Average Solution (Solution Gap %)*	*Max. Solution (Solution Gap %)*
Small	1	3	50,695	50,695 (0)	50,695 (0)	50,695 (0)
	2	5	97,590*	97,530 (-0.06)	98,391 (0.82)	100,243 (2.72)
Large	3	10	194,820*	192,320 (-1.28)	194,305 (-0.26)	195,500 (0.35)
	4	20	-	344,648	346,686	351,023
	5	30	-	505,053	510,232	513,57
	6	50	-	829,377	838,185	847,263

* Branch and bound method in CPLEX runs out of memory

5.2 Performance comparison in computational speeds

The minimal, average and maximal CPU time of all solved problems are shown in Figure 4. From Figure 4, we observed that the computational speeds of the HPSO algorithm outperform the traditional B&B algorithm and the B&B algorithm grows exponentially with the problem sizes. The HPSO algorithm only needs fewer CPU time and memory to generate the near-optimal solution compared with the terminated B&B algorithm and have best performances in large size problems. From the above analyses of the solution quality and computational time, it can be deduced that the HPSO algorithm can provide better quality solutions within a reasonable amount of time as data size increases. Thus, this heuristic is more suitable for solving realistically large size problems of hybrid flow shop scheduling problems.

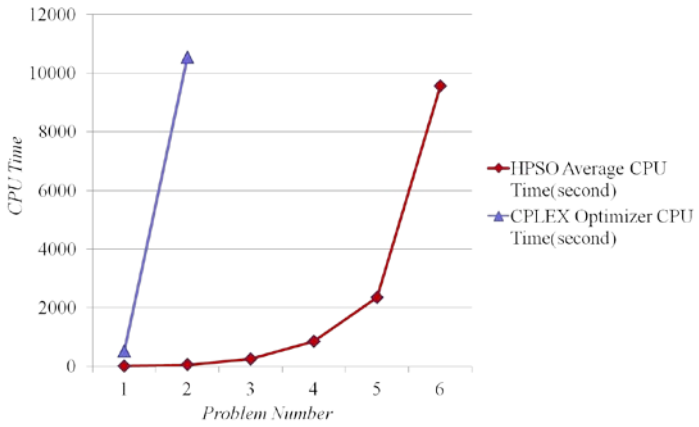

Figure 4 The comparison graph of CPU time between B&B and HPSO algorithm

6. Conclusion

This paper proposes a hybrid-PSO approach for solving production scheduling problem with multi-stages and parallel-machines in the DRAM module industry, similar to the traditional hybrid flow shop scheduling (HFS). The HFS problem simultaneously determines order production sequence, multiprocessor task scheduling and optimal machine configuration through dynamically allocating all

jobs to multiple machines under the minimization of the maximum makespan. A hybrid-PSO approach combined VNS algorithm with PSO algorithm, here VNS is applied for determining the order production sequence and PSO solves the allocating problem that allocating each order to multiple machines. A proposed hybrid-PSO considers of many practical characteristics including parallel machine system, sequence-independent setup time, and sequence-dependent setup time. The computational study shows that the proposed hybrid-PSO could be more suitable and efficient for solving large size problems than the conventional B&B algorithm both in solution quality and computational speed. For the future research, other heuristic algorithms may be employed to efficiently attack large-scale instances and compare with the proposed algorithm.

Acknowledgements

The authors gratefully acknowledge the National Science Council, Taiwan, R.O.C., for support under contract NSC 98-2221-E-029-018-MY3.

References

Anghinolfi, D., Paolucci, M., 2009. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. European Journal of Operational Research 193, 73–85.

Baykasoglu, A., Gindy, N.N.Z., 2000. MOCACEF 1.0: Multiple objective capability based approach to form part-machine groups for cellular manufacturing applications.

Behnamian, J., Fatemi Ghomi, S.M.T., Zandieh, M., 2009a. A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid

flowshop using a hybrid metaheuristic. Expert Systems with Applications 36, 11057–11069.

Behnamian, J., Zandieh, M., Fatemi Ghomi, S.M.T., 2009b. Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. Expert Systems with Applications 36, 9637-9644.

Chen, J., Lee, C.-Y., 1999. General Multiprocessor Task Scheduling. Naval Research Logistics 64, 57-74.

Chen, C.-L., Chen, C.-L., 2009. Hybrid metaheuristic for unrelated parallel machine scheduling with sequence-dependent setup times. International Journal of Advanced Manufacturing Technology 43, 161-169.

Chuang, L.-Y., Tsai, S.-W., Yang, C.-H., 2008a. Catfish Particle Swarm Optimization, Swarm Intelligence Symposium, 2008. SIS 2008. IEEE St. Louis, MO pp. 1-5.

Chuang, L.-Y., Tsai, S.-W., Yang, C.-H., 2011. Chaotic catfish particle swarm optimization for solving global numerical optimization problems. Applied Mathematics and Computation 217, 6900-6916.

Coelho, L.d.S., Mariani, V.C., 2009. A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch. Chaos, Solitons & Fractals 39, 510–518.

Garey, M., Johnson, D.S., 1979. Computers and intractability: a guide to the theory of NP-completness. W. F. Freeman.

International Journal of Production Research 38, 1133-1162.

Gronalt, M., Grunow, M., Gunther, H.O., Zeller, R., 1997. A heuristic for components switching on SMT placement machines. International journal of Production Economics 53, 181-190.

Islam K.M.S, Sarker, B.R., 2000. A similarity coefficient measure and machine-parts

grouping in cellular manufacturing systems. International Journal of Production Research 38, 699-720.

Jarboui, B., Eddaly, M., Siarry, P., 2009. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Computers and Operations Research 36, 2638–2646.

Kashan, A.H., Karimi, B., 2009. A discrete particle swarm optimization algorithm for scheduling parallel machines. Computers & Industrial Engineering 56, 216-223.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, IEEE International Conference on Neural Networks, pp. 1942–1948.

Kou, X., Liu, S., Zhang, J., Zheng, W., 2009. Co-evolutionary particle swarm optimization to solve constrained optimization problems. Computers & Mathematics with Applications 57, 1776-1784.

Kusiak A., 1990. Intelligent Manufacturing Systems. Prentice Hall.

Maimon, O.Z., Braha, D., 1998. A genetic algorithm approach to scheduling PCBs on a single machine. International Journal of Production Research 36, 761-784.

Mansouri, S.A., Husseini, S.M.M., Newman, S.T., 2000. A review of the modern approaches to multi-criteria cell design. International Journal of Production Research 38, 1201-1218.

Quadt, D., Kuhn, H., 2007. Batch scheduling of jobs with identical process times on flexible flow lines. International Journal of Production Economics 105, 385-401.

Mladenovi´c N, P, H., 1997. Variable neighborhood search. Computers and Operations Research 24, 1097-1100.

Rahimi-Vahed, A., Dangchi, M., Rafiei, H., Salimi, E., 2009. A novel hybrid multi-objective shuffled frogleaping algorithm for a bi-criteria permutation flow shop scheduling problem. International Journal of Advanced Manufacturing Technology 41,

1227–1239.

Rajkumar, K., Narendran, T.T., 1998. A heuristic for sequencing PCB assembly to minimize set-up times. Production Planning and Control 9, 465-476.

Rubén Ruiz, Vázquez-Rodríguez, J.A., 2010. The hybrid flow shop scheduling problem European Journal of Operational Research 205, 1-18.

Salvador, M.S., 1972. A solution to a special class of flow shop scheduling problems, in: Elmaghraby, S.E. (Ed.), Symposium on the theory of scheduling and its applications. Case Western Reserve University, pp. 83-91.

Sun, C.-l., Zeng, J.-c., Pan, J.-s., 2011a. An improved vector particle swarm optimization for constrained optimization problems. Information Sciences 181, 1153-1163.

Tang, C.S., Denardo, E.V., 1988. Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. Operations Research 36, 767-777.

Van Hop, N., Tabucanon, M.T., 2000. Fuzzy multi-attribute decision-making for grouping of electronic components in process planning. Production Planning and Control 11, 677-688.

Van Hop, N., Nagarur, N.N., 2004. The scheduling problem of PCBs for multiple non-identical parallel machines. European journal of Operational Research 158, 577-594.

Zhang, C., Sun, J., 2009. An alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. Expert Systems with Applications 36, 5162-5167.