Expert Systems with Applications 36 (2009) 10962–10968

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An efficient approach to cross-fab route planning for wafer manufacturing

Muh-Cherng Wu *, Chang-Fu Shih, Chen-Fu Chen

Department of Industrial Engineering and Management, National Chiao Tung University, Hsin-Chu 300, Taiwan

ARTICLE INFO

Keywords:
Dual-fab
Cross-fab route
Route planning
Capacity-sharing
Genetic algorithm
Linear programming

ABSTRACT

This paper proposes an efficient approach to solve a cross-fab route planning problem for semiconductor wafer manufacturing. A semiconductor company usually adopts a *dual-fab* strategy. Two fab sites are built neighbor to each other to facilitate capacity-sharing. A product thus may be produced by a *cross-fab route*; that is, some operations of a product are manufactured in one fab and the other operations in the other fab. This leads to a cross-fab routing planning problem, which involves two decisions—determining the cut-off point of the cross-fab route and the route ratio for each product—in order to maximize the throughput subject a cycle time constraint. A prior study has proposed a method to solve the cross-fab route planning problem; yet it is computationally extensive in solving large scale cases. To alleviate this deficiency, we proposed three enhanced methods. Experiment results show that the best enhanced method could significantly reduce the computational efforts from about 13 h to 0.5 h, while obtaining a satisfactory solution.

1. Introduction

Recently, some semiconductor companies tend to adopt a *dual-fab* strategy. The strategy advocates that building two fabs at a time, which are next door to each other. One reason why this strategy arises is due to that semiconductor equipment, compared with fab space, is much more expensive and shorter in lead time of acquisition. In practice, more than 80% expenditure of an advanced wafer fab is spent on equipment. The acquisition lead time for equipment ranges from 3 to 9 months, while that for fab space takes about 1–2 years. To quickly respond to volatile market demand, some companies tend to build a large scale space for two fabs in advance and gradually purchase equipment based on future demand over time.

Not only good in fast capacity expansion, the dual-fab strategy also provides a *capacity-sharing* mechanism due to close proximity of the two fabs. Consider a *single-fab* production policy which requests each wafer job be manufactured in *only* one fab. In a dual-fab configuration, such a policy usually leads to underutilization of equipment because idle equipment capacity in one fab cannot be used by the other. To utilize the idle capacity, we may have to adopt a *cross-fab* production policy. That is, the manufacturing of a wafer job could be partly done in one fab and partly in the other.

However, under the cross-fab production policy, we would be confronted with a *route planning* problem—how to appropriately assign the operations of a wafer job to each of the two fabs. Prior studies on such a route planning problem are relatively few. With each job route being cut into several segments, Toba, Izumi, Hatada, and Chikushima (2005) studied the route planning problem in a *real-time* manner. That is, whenever a segment is completed, a decision—which fab to manufacture the next segment—must be immediately made. Wu and Chang (2007) examined a route planning problem in a *weekly* horizon. Assuming the two fabs plan capacity exchange weekly, they attempted to find an optimal capacity-trading portfolio in order to maximize the total throughput of the two fabs.

Aside from the track of short-term route planning, Wu, Erkoc, and Karabuk (2005) addressed the problem from a *relatively long-term* perspective. Given a product mix to produce, say in a *quarter*, they attempted to determine how to cut the route of each product into two segments; and determine the production ratio of each segment that should be assigned to each fab. Their objective function is to maximize the total throughput of the two fabs subject that a target cycle time must be met.

Numeric experiments indicated that the method proposed by Wu, Chen, and Shih (2008) could effectively increase equipment utilization and total throughput for a dual-fab scenario. However, their method may become computationally extensive in dealing with large scale cases.

In order to *efficiently* solve the route planning problem, this paper presents an enhanced approach based on Wu et al. (2008). Numeric experiments indicated that solutions obtained by the enhanced approach are almost as good as that obtained by Wu et al. (2008) yet requires much less computational efforts.

The remainder of this paper is organized as follows. Section 2 reviews relevant literature. Section 3 explains the route planning problem. Section 4 outlines the LP–GA solution framework pro-

* Corresponding author. Tel.: +886 3 5731913; fax: +886 3 5720610.
E-mail address: mcwu@mail.nctu.edu.tw (M.-C. Wu).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.02.008

posed by Wu et al. (2008). Section 5 presents the linear program (LP) and our enhancements to reduce computational time. Section 6 presents the genetic algorithm (GA) and our enhancements to reduce computational time. Numerical experiments are in Section 7 and concluding remarks are in the last section.

2. Relevant literature

In a company with multiple manufacturing sites, planners would face a capacity allocation decision—how to allocate a given demand to each manufacturing site. Literature on the capacity allocation problem could be grouped in two categories: product-level and operation-level.

In the product-level category, most literature assumed a single-site production policy—each product should be completely manufactured with a single-site. A literature survey has been published by Wu et al. (2005), and some recent studies can be referred to Chiang, Guo, Chen, Cheng, and Chen (2007), Lee, Chung, Lee, and Kang (2006) and ManMohan (2005). Most of these prior studies used the linear programming (LP) technique to solve the capacity allocation problems.

In the operation-level category, most literature assumed a cross-site production policy manufacturing operations for a product could be distributed among different sites. The need of studying the cross-site-route planning problems thus arises. Such route planning problems were mostly addressed in the context of group technology (GT). Example literature includes Dimopoulos (2006), Kim, Beak, and Jun (2005), Mahdavi, Rezaeian, Shanker, and Amiri (2006), Nsakanda, Diaby, and Price (2006), Spiliopoulos and Sofianopoulou (2007), Vin, Lit, and Delchambre (2005).

In GT, each site is a manufacturing cell and multiple cells from a factory. A GT cell is designed for manufacturing a particular group of products, and by nature is functionally limited. A cell thus may need to outsourcing decision of each cell.

By contrast, in the route planning problem we address, each of the two fabs is assumed to be functionally comprehensive. Each product can be completely manufactured in either one of the two fabs. The purpose of cross-fab route planning is to maximize the aggregate throughput through optimum capacity-sharing.

3. Problem statement

The *dual-fab route planning* problem is explained in more detail, where the two fabs are called *Fab_1* and *Fab_2*. We first present the assumptions, and proceed to the decision variables, objective function and constraints.

Assumption 1: Each fab is functionally comprehensive. Both fabs are so comprehensively equipped that each fab can individually complete the production of each product—not requiring support of the other fab.

Assumption 2: The transportation path between any two workstations/buffers is unique, rather than multiple. In practice, there exist multiple paths in transporting a wafer job from a workstation/buffer to another. However, to reduce the problem complexity, we assume that a fixed path is predefined for such a transport.

Assumption 3: Each product has only four possible routes. The process route of each product is cut into two segments, where a route's break point is called a *cut-off point*. A product has four possible manufacturing routes: $1 \to 2$, $2 \to 2$, $1 \to 2$, and $2 \to 1$, where notation $i \to j$ denotes the first segment is manufactured at *Fab_i* while the second is manufactured at *Fab_j*.

Define $\overline{r_i} = [a_i, b_i, c_i, d_i]$ as the percentage of the four possible routes of product i. Each element of $\overline{r_i}$ in sequence represents the route percentage of $1 \to 1$, $2 \to 2$, $1 \to 2$ and $2 \to 1$. Define π_i as the cut-off point for the route of product i, which is the identification code (an integer) of the operation for separating a route into two segments. The range of π_i is $1 \leqslant \pi_i \leqslant o_i - 1$ where o_i is the total number of operation of product i. We set $\pi_i = 0$ while we determine not to manufacture product i by using any cross-fab routes.

Consider a dual-fab company that has n products to manufacture, represent a solution of the route planning problem by (Π, R), where $\Pi = [\pi_1, \ldots, \pi_n]$ and $R = [\overline{r_1}, \ldots, \overline{r_n}]$. The objective is to find an optimal solution (Π^*, R^*) in order to maximize the total throughput of the two fabs, subject to the constraint of meeting a target cycle time.

4. Solution framework

To solve the dual-fab route planning problem, we adopted the solution framework proposed by Wu et al. (2008), and developed several enhancements to their solution method in order to reduce computational efforts. As shown in Fig. 1, the solution framework involves two modules.

In Module 1, each path is assumed to be equipped with *infinite transportation capacity*; and the transportation time between any two workstations/buffers is thus zero. The problems so simplified are solved by an iterative use of a linear program (LP) model. This module is intended to find an optimum solution (Π^*_L, R^*_L), in terms of minimizing the total number of inter-fab transportations. In the prior study (Wu et al., 2008), this module is very computationally extensive because the number of LP iterations is quite huge for large scale cases. We proposed three heuristic methods to enhance the prior study by significantly reducing the number of LP iterations.

Let (Π^*_L, R^*_L) represent the solution obtained in Module 1. In Module 2—by taking Π^*_L as given parameters, we deal only with decision variable R by considering each path as a tool with *limited transportation capacity*. The transportation time for a path depends upon the traffic flow intensity. The higher the traffic intensity, the longer is the cycle time. The performance of a particular (Π^*_L, R) could be evaluated by applying a queueing network model (Connors, Feigin, & Yao, 1996). In the prior study (Wu et al., 2008), they developed a GA to find a near-optimal solution from the space $\{(\Pi^*_L, R)\}$, which is also computationally extensive while dealing with large scale cases. We enhanced the prior study by reducing the size of the GA chromosomes. In the enhanced GA, many elements in R are considered to be constant and only a few need to searched.

The essences of these two modules are compared below. Module 1 essentially deals with a *static capacity allocation* problem which does not consider job flow time. In contrast, Module 2 deals with a *time-phased capacity allocation* problem, in which job flow time is addressed and computed by a queueing network model.

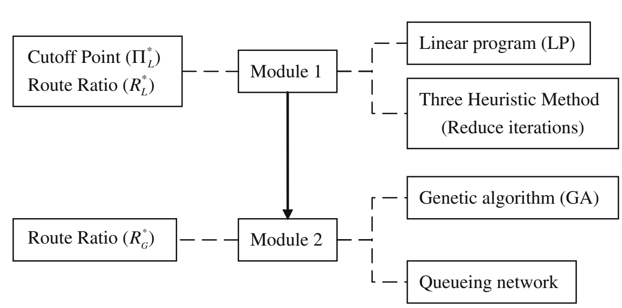

Fig. 1. Solution framework.

5. Module 1 – LP model and enhancements

Obtaining the solution for Module 1 is through an *iterative* use of an LP model. We first describe the LP model, and then present the architecture of the iterative process. Finally, we describe the three methods designed to reduce the number of LP iterations.

5.1. LP model

Indices

i index of product
g index of workstation in *Fab_1*
h index of workstation in *Fab_2*

Parameters

n total number of products
π_i cut-off point for defining the cross-fab routes of product i
Π $\Pi = [\pi_i]$, $1 \leqslant i \leqslant n$, the cut-off points of all products
V estimated total throughput of the two fabs, input by user
z_i percentage of product i in the given product mix, $\sum_{i=1}^{n} z_i = 1$, $0 \leqslant z_i \leqslant 1$
C_g available machine hours of workstation g in *Fab_1*
C_h available machine hours of workstation h in *Fab_2*
m_1 total number of workstations in *Fab_1*
m_2 total number of workstations in *Fab_2*
W_{ig}^a total processing time per lot required on workstation g in *Fab_1*, while product i is manufactured by route $1 \to 1$
W_{ig}^c total processing time per lot required on workstation g in *Fab_1*, while product i is manufactured by route $1 \to 2$
W_{ig}^d total processing time per lot required on workstation g in *Fab_1*, while product i is manufactured by route $2 \to 1$
W_{ih}^b total processing time per lot required on workstation h in *Fab_2*, while product i is manufactured by route $2 \to 2$
W_{ih}^c total processing time per lot required on workstation h in *Fab_2*, while product i is manufactured by route $1 \to 2$
W_{ih}^d total processing time per lot required on workstation h in *Fab_2*, while product i is manufactured by route $2 \to 1$

Decision variables

R $R = [\overline{r_1}, \ldots, \overline{r_n}]$, where $\overline{r_i} = [a_i, b_i, c_i, d_i]$
a_i percentage of using route $1 \to 1$ in producing product i
b_i percentage of using route $2 \to 2$ in producing product i
c_i percentage of using route $1 \to 2$ in producing product i
d_i percentage of using route $2 \to 1$ in producing product i

The LP model is to compute an optimum R for a given pair of (V, Π), in terms of minimizing the number of cross-fab transportation. Define the objective function by $Z(V, \Pi)$. The LP model is formulated below:

$$\text{Min} \quad Z(V,\Pi) = \sum_{i=1}^{n} V \cdot z_i \cdot (c_i + d_i)$$

$$\text{s.t.} \quad a_i + b_i + c_i + d_i = 1, \quad 1 \leqslant i \leqslant n, \tag{1}$$

$$\sum_{i=1}^{n} V \cdot z_i \cdot (a_i \cdot W_{ig}^a + d_i \cdot W_{ig}^d + c_i \cdot W_{ig}^c) \leqslant C_g, \quad 1 \leqslant g \leqslant m_1, \tag{2}$$

$$\sum_{i=1}^{n} V \cdot z_i \cdot (b_i \cdot W_{ih}^b + d_i \cdot W_{ih}^d + c_i \cdot W_{ih}^c) \leqslant C_h, \quad 1 \leqslant h \leqslant m_2. \tag{3}$$

The objective function is to minimize the number of cross-fab production lots. The rationale for defining this objective is that cross-fab production requires longer transportation time than within-fab production. Subject to a target cycle time, an attempt to minimize cross-fab production lots tends to increase total throughput. Constraint (1) describes the dependent relationship among the route ratios. Constraints (2) and (3) ensure that the capacity used in each workstation, in *Fab_1* and *Fab_2*, should be lower than its available supply. Notice that V is the estimated throughput; the LP may yield no solution while V is too large.

In the above LP model, each of the n products is eligible for cross-fab production. To reduce computational complexity, we propose to divide the products into two sets: Q_c and Q_s. Products in Q_c are eligible for *cross-fab* production, and those in Q_s are only allowed for*single-fab* production. To deal with such a general scenario (Q_c, Q_s), the above LP should be modified by including the following constraints:

$$c_k = 0 \quad \text{and} \quad d_k = 0 \quad \text{for each product } k \text{ in } Q_s. \tag{4}$$

A procedure *LP_Module*(V, Π, Q_c, Q_s) is defined below to facilitate explaining the iterative procedures for calling the modified LP.

Procedure LP_Module (V, Π, Q_c, Q_s)

Step 1: Compute LP(V, Π, Q_c, Q_s)
Step 2: If (LP has no solution) then Pass_Check = "Fail", **Return**
If (LP has solution) then Pass_Check = "Pass",
Return $Z(V, \Pi)$, $R(V, \Pi)$, Pass_Check

In Step 1 of the above procedure, LP(V, Π, Q_c, Q_s) denotes the modified LP. In Step 2, Pass_Check is a flag in which "Fail" denotes the value of V is too large. Moreover, $Z(V, \Pi)$ denotes the obtained route ratio and $Z(V, \Pi)$ denotes the obtained value in objective function.

5.2. Iterative process of LP

To solve the route planning problem, we need to iteratively run *LP_Module*(V, Π, Q_c, Q_s). The architecture of the iterative process is shown in Fig. 2. The architecture involves four procedures, which are organized in a hierarchical manner. The bottom level of the hierarchy is the *LP_Module*(V, Π, Q_c, Q_s). Details of the other three procedures are presented in Appendices 1–3.

Of the three top level procedures, *Route_Planning* is intended to ask users input (Q_c, Q_s) and (L, U) which is the range of V. Given a scenario (L, U, Q_c, Q_s), *Route_Planning_for_Given_Throughput* is

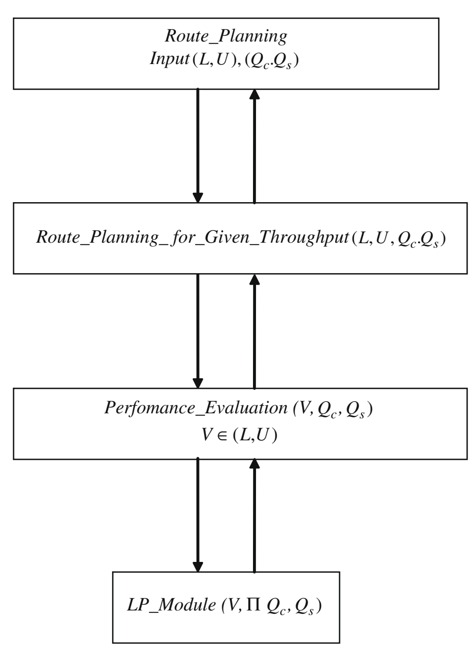

Fig. 2. Logic flow of the LP iterative procedures.

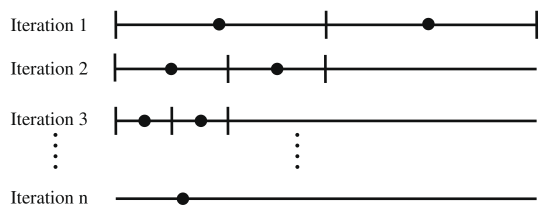

Fig. 3. Binary-search method.

intended to find an optimal $V \in (L, U)$, where the algorithm for identifying V is based on a binary-search method (Fig. 3) *Performance_Evaluation* is intended to find (Π^*, R^*) for a given scenario (V, Q_c, Q_s), based on a binary-search algorithm over multiple intervals and each interval is a product route.

Assume set Q_c has n_c products; that is, there are n_c product routes to search for their optimal cut-off points. The computational complexity of the iterative process is $O(2^{n_c \cdot k_1})$, where k_1 is a constant which denotes the maximum number of search required to carry out on each product route. It might be very much computationally extensive while $n_c = n$ (i.e., all products are eligible for cross-fab production). One way to efficiently solve the route planning problem is to find an appropriate Q_c, which has small value of n_c and can yield a good quality solution.

5.3. Reduction of iteration number

To find such an appropriate Q_c, we developed a procedure *Product_Sorting* to categorize all products into three groups. Taking each group as a particular selection of Q_c, we would have three different versions of Q_c. The procedure is presented below.

Procedure *Product_Sorting*

Step 1: Identify the bottleneck workstation (say, B) of the two fabs.
Step 2: Compute the workload of each product on B.
Step 3: Sort all the n products according to their workload on B.
Step 4: Categorize products into three groups, based on the sorted results.

With three different versions of Q_c, we could have three solution methods in Module 1. The method, using the product group with the highest bottleneck workload, is called ***LP₁***. The one with middle-level bottleneck workload is called ***LP₂***, and the remaining one is called ***LP₃***. The method proposed by Wu et al. (2008) is called ***LP₀***.

The rationale for taking bottleneck workload as the criterion for grouping products is two-fold. First, the utilization of bottleneck workstation dominates the two fabs' throughput. Thus, in cross-fab route planning decisions, the capacity allocation of bottleneck workstation would be most critical. Second, we attempt to justify which product group is most critical in the cross-fab route planning—the heavily load group, the middle-level load group, or the lightly loaded group.

6. Module 2 – GA

Define the solution of Module 1 as (Π_L^*, R_L^*), which is obtained under the assumption of *infinite transportation capacity*. In Module 2, with (Π_L^*, R_L^*) being available, we developed a GA in order to find a better solution (Π_G^*, R_G^*) under the assumption of *finite transportation capacity*.

The GA is an enhanced version of the one proposed by Wu et al. (2008). Like Wu et al. (2008), we first set $\Pi_G^* = \Pi_L^*$ and attempt to find R_G^*. But in the search of R_G^*, we make an enhancement by setting $c_k = d_k = 0$ for each product k in Q_s (i.e., the single-fab production policy presumed in Module 1 is preserved).

The enhancement could simplify the representation of a solution. Consider a chromosome (a possible solution) represented by a vector $R = [\overline{r_1}, \ldots, \overline{r_n}]$, where $\overline{r_i} = (a_i, b_i, c_i, d_i)$. We call $\overline{r_i}$ a *gene-segment*, and each element in $\overline{r_i}$ a *gene*. Since $a_i + b_i + c_i + d_i = 1$, we have three *free* genes for each product in Q_c and one *free* gene for each product in Q_s. Here, a *free* gene is one whose value is changeable in the search process, while a gene whose value is not changeable is called a *static* gene. With this enhancement, a chromosome has only $3n_c + (n - n_c)$ free genes, rather than $3n$ ones as in Wu et al. (2008). The GA proposed by Wu et al. (2008) is called ***GA₀*** and our enhanced version is called ***GA₁***.

The performance (also called fitness) of each chromosome is computed by a queueing network model (Wu et al., 2008), which is adapted from the one developed by Connors et al. (1996). For a given chromosome (i.e., a route plan), the queueing network can be used to compute the aggregate throughput of the two fabs subject to meeting a target cycle time.

The GA is an iterative algorithm which can be briefly described as follows:

Procedure GA

Step 1: Initialization
- $t = 0$, Status = 'Not-terminate'
- Randomly generate N_p chromosomes to form a population P_0

Step 2: Genetic Evolution
While (Status = 'Not-Terminate') do
- Use a *cross-over* operator to create N_c new chromosomes
- Use a *mutation* operator to create N_m new chromosomes
- Form a pool by taking the union of P_t and the set of newly created chromosomes
- $t = t + 1$, and select the best N_p chromosomes from the pool to form P_t
- Check if *termination condition* is met; if yes, set Status = "Terminate"

Endwhile
Step 3: Set the best chromosome in P_t as R_G^*. Output R_G^*.

The cross-over operation is to create two new chromosomes (say, R_3 and R_4) from two existing ones (say, R_1 and R_2). Let each *gene-segment i* in R_1 and R_2 be respectively represented by $\overline{r_{i1}}$ and $\overline{r_{i2}}$. We proposed a one-point cross-over operation (Binh & Lan, 2007) on gene-segments $\overline{r_{i1}}$ and $\overline{r_{i2}}$ to create two new ones $\overline{r_{i3}}$ and $\overline{r_{i4}}$, which in turn could yield two new chromosomes: $R_3 = [\overline{r_{i3}}]$, $R_4 = [\overline{r_{i4}}]$, $1 \leqslant i \leqslant n$.

The one-point cross-over operation on a gene-segment is briefly introduced. For two gene-segments (i.e., $\overline{r_{i1}}$ and $\overline{r_{i2}}$), we randomly choose a *free* gene, swap their gene values, and modify another gene values in order to ensure meeting the constraint $a_i + b_i + c_i + d_i = 1$. Consider an example, where the 2nd gene (a free one) is chosen as the cross-over point for mixing $\overline{r_{i1}} = (a_{i1}, b_{i1}, c_{i1}, d_{i1})$ and $\overline{r_{i2}} = (a_{i2}, b_{i2}, c_{i2}, d_{i2})$. By the swap and modification operations, we would obtain $\overline{r_{i3}} = (a_{i1}, b_{i2}, c_{i1}, 1 - a_{i1} - b_{i2} - c_{i1})$ and $\overline{r_{i4}} = (a_{i2}, b_{i1}, c_{i2}, 1 - a_{i2} - b_{i1} - c_{i2})$.

In the mutation operation, a new chromosome (say, R_2) is created by an existing one (say, R_1). The mutation algorithm creates R_2 by modifying a particular gene-segment in R_1. The modified gene-segment is randomly chosen. While being selected, two of its free genes are randomly chosen and their gene values are swapped. For example, if gene-segment i^* is chosen for modification; and the 2nd and 4th genes are chosen to swap for $\overline{r_{i^*1}} = (a_{i1}, b_{i1}, c_{i1}, d_{i1})$, then $\overline{r_{i^*2}} = (a_{i1}, d_{i1}, c_{i1}, b_{i1})$, which in turn yield a new chromosome $R_2 = [\overline{r_{11}}, \overline{r_{i^*2}}, \ldots, \overline{r_{n1}}]$ from $R_1 = [\overline{r_{11}}, \overline{r_{i^*1}}, \ldots, \overline{r_{n1}}]$. Notice that only products in Q_c are eligible for applying the mutation operation.

Two termination conditions are defined for the GA. First, the best solution in P_t has not been changed for over a certain period (say, T_b iterations). Second, population P_t has evolved over a certain number of iterations; that is, t has reached its predefined upper bound (T_u).

7. Experiments

Numeric experiments are carried out to compare the performance of our three proposed methods against the one proposed by Wu et al. (2008). The one proposed by Wu et al. (2008) is called $\boldsymbol{LP_0-GA_0}$. The three we proposed are respectively called $\boldsymbol{LP_1-GA_1}$, $\boldsymbol{LP_2-GA_1}$ and $\boldsymbol{LP_3-GA_1}$. A personal computer equipped with Pentium (R) Dual CPU 3.4 GHz and 1 GB RAM is used in the experiments.

In the experiments, the data for machines, product routes and operation times are adapted from a data set provided by a semiconductor company. Each of the two fabs involves 60 workstations. Fab_1 involves 292 machines and Fab_2 involves 352 machines. The MTBF (mean time between failure) and MTTR (mean time to repair) of each machine is available, exponentially distributed.

Three scenarios are considered in the experiments. Scenario 1 involves three products (Table 1); Scenario 2 involves six products (Table 2); Scenario 3 involves nine products (Table 3). In the genetic algorithms, we set $T_b = 10{,}000$, $T_u = 500$, $P_0 = 1000$, $P_{cr} = 0.9$ and $P_m = 0.1$. The target cycle time is $CT_0 = 40{,}000$ min or 27.7 days.

Table 4 compares the four methods in terms of the two fabs' aggregate throughput. Of the three proposed methods, $\boldsymbol{LP_1-GA_2}$ appears to be the best one, in particular in Scenario 3—only 2.48% less than $\boldsymbol{LP_0-GA_0}$ in throughput. However, the computation time required by $\boldsymbol{LP_2-GA_1}$ is greatly reduced. From Table 5, in dealing with Scenario 3, $\boldsymbol{LP_0-GA_0}$ requires 46,578 s (about 13 h), while $\boldsymbol{LP_2-GA_1}$ requires only 2112 s (about 35 min). In practice, taking half a day in computation is generally not acceptable to practitioners. Therefore, $\boldsymbol{LP_2-GA_1}$ appears to be a useful decision aid in solving cross-fab route planning problems.

Table 6 shows the two components of computation times required in Scenario 3. Table 6 indicates that the reduction in computation time is substantially due to the enhancement in LP. In the LP module, $\boldsymbol{LP_0-GA_0}$ takes 42,900 s (about 12 h) while $\boldsymbol{LP_2-GA_1}$ requires only 110 s (about 2 min).

The reasons why $\boldsymbol{LP_2-GA_1}$ outperforms the other two proposed methods, in terms of solution quality, are analyzed below. In $\boldsymbol{LP_1-GA_1}$, products in Q_c are high-level in terms of bottleneck workload. This implies that these products are higher in product mix ratios. This leads to a higher eligible range for each route ratio in Q_c. In turn, the GA solution space of route ratios would become much larger. Under the same GA terminating conditions, the solution obtained by $\boldsymbol{LP_1-GA_1}$ may not be as good as that obtained by $\boldsymbol{LP_2-GA_1}$.

By contrast, in $\boldsymbol{LP_3-GA_1}$, products in Q_c are low-level in terms of bottleneck workload; that is, products are generally lower in product mix ratios. This leads to a lower eligible range for each route ratio in Q_c. In turn, the space for improving the solution quality is also reduced. Therefore, $\boldsymbol{LP_2-GA_1}$ would outperform $\boldsymbol{LP_3-GA_1}$.

8. Conclusion

This paper presents an efficient approach to solve cross-fab route planning problems for semiconductor wafer manufacturing. In the problem, each product has four possible production routes,

Table 1
Scenario 1 which involves three products in the route planning.

Product	P1	P2	P3
Number of operations	338	338	338
Product mix	0.5	0.3	0.2

Table 2
Scenario 2 which involves six products in the route planning.

Product	P1	P2	P3	P4	P5	P6
Number of operations	338	338	338	300	300	300
Product mix	0.25	0.25	0.15	0.15	0.1	0.1

Table 3
Scenario 3 which involves nine products in the route planning.

Product	P1	P2	P3	P4	P5	P6	P7	P8	P9
Number of operations	338	338	338	300	300	300	250	250	250
Product mix	0.17	0.17	0.16	0.1	0.1	0.1	0.07	0.07	0.06

Table 4
Throughput comparison for various solution methods.

Scenario	Scenario 1		Scenario 2		Scenario 3	
	TH (Lot)	Gap (%)	TH (Lot)	Gap (%)	TH (Lot)	Gap (%)
LP_0-GA_0	652	0	725	0	846	0
LP_1-GA_1	650	0.31	724	0.14	795	6.03
LP_2-GA_1	651	0.15	723	0.28	825	2.48
LP_3-GA_1	650	0.31	697	3.86	790	6.62

Table 5
Computation time comparison for various solution methods.

Scenario	Scenario 1		Scenario 2		Scenario 3	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
LP_0-GA_0	892	0	3111	0	46,587	0
LP_1-GA_1	437	48.99	1497	48.12	2197	4.72
LP_2-GA_1	532	59.64	1478	47.51	2112	4.53
LP_3-GA_1	539	60.43	1590	51.11	2940	6.31

Table 6
Computation time analysis for LP_0-GA_0 and LP_2-GA_1.

Algorithm	LP time (s)	Gap (%)	GA + queueing (s)	Gap (%)	Total time (s)	Gap (%)
Scenario 3						
LP_0-GA_0	42,900	0	3687	0	46,587	0
LP_2-GA_1	110	0.26	2002	54.3	2112	4.53

which are defined by a cut-off point. We need to determine the cut-off point and the route ratio for each product in order to maximize the throughput subject a cycle time constraint.

A prior study has proposed a method (called $\boldsymbol{LP_0-GA_0}$) to solve the problem, yet it is computationally extensive in dealing with large scale cases. In this paper, we enhanced the prior method and proposed three efficient methods (called $\boldsymbol{LP_1-GA_1}$, $\boldsymbol{LP_2-GA_1}$ and $\boldsymbol{LP_3-GA_1}$). Numerical experiments indicate that the three enhanced methods can significantly reduce the required computation time. Of the three enhanced methods, $\boldsymbol{LP_2-GA_1}$ outperforms the other two in terms of solution quality, in dealing with large scale cases.

Some extensions of this research are being considered. The first extension is the route planning for a multiple-fab production system—for example, three or more fabs share the capacity in production. The second extension is the route planning for a scenario with higher flexibility in production routes—for example, each product could have two or more cut-off points and in turn have more than four routes.

Appendix 1

Procedure *Route_Planning*

Step 1:
  Input $(\boldsymbol{L},\boldsymbol{U})$
  Input $(\boldsymbol{Q_c},\ \boldsymbol{Q_s})$
Step 2:
  Call ***Route_Planning_for_Given_Throughput*** $(\boldsymbol{L},\boldsymbol{U},\boldsymbol{Q}_c,\boldsymbol{Q}_s)$
Step 3:
  Output Z^*, Π_L^*, R_L^*

Appendix 2

Procedure *Route_Planning_for_Given_Throughput* $(\boldsymbol{L},\boldsymbol{U},\boldsymbol{Q}_c,\boldsymbol{Q}_s)$

Initialization /* set initial range of throughput*/
  $i=1$, /* i is iteration number*/
  $L_i=L$, $U_i=U$
  $I_i=[L_i,U_i]$
While $\left\{\left(i=1 \text{ or } \frac{V_2-V_1}{V_1}\geqslant\varepsilon\right)\right\}$ /* ε is a small value, e.g., 0.2%*/
  Step 1: Determine the two test points for the throughput interval I_i
    $V_1=\lfloor(U_i+L_i)/4\rfloor$
    $V_2=\lfloor 3(U_i+L_i)/4\rfloor$
  Step 2: Evaluate and record the performance of the two test points
    Call *Performance_Evaluation* $(\boldsymbol{V}_1,\boldsymbol{Q}_c,\boldsymbol{Q}_s)$
      $P_1=Pass\_Check(V_1)$ /* Check if V_1 is too large*/
      $\Pi_1=Optimal\_Cutoff(V_1)$
      $R_1=Optimal\_Route\_Ratio(V_1,\Pi_1)$
      $Z_1=Optimal\_Objective\_Value(V_1,\Pi_1)$
    Call *Performance_Evaluation* $(\boldsymbol{V}_2,\boldsymbol{Q}_c,\boldsymbol{Q}_s)$
      $P_2=Pass\_Check(V_2)$ /* Check if V_2 is too large*/
      $\Pi_2=Optimal\_Cutoff(V_2)$
      $R_2=Optimal\_Route\_Ratio(V_2,\Pi_2)$
      $Z_2=Optimal\_Objective\_Value(V_2,\Pi_2)$
  Step 3: Update the throughput interval for search
    If (P_2 = "Pass") then $L_{i+1}=\lfloor(U_i+L_i)/2\rfloor$, $U_{i+1}=U_i$, $k=2$
    If (P_1 = "Pass") and (P_2 = "Fail") then $L_{i+1}=L_i$, $U_{i+1}=\lfloor(U_i+L_i)/2\rfloor$, $k=1$
    If (P_1 = "Fail") and (P_2 = "Fail") then $L_{i+1}=L_i$, $U_{i+1}=\lfloor(U_i+L_i)/4\rfloor$, $k=0$
    $i=i+1$
Endwhile
  If $k=0$, Stop /* User warning: the input value of L is too large*/
  Else $Z^*=Z_k$, $\Pi_L^*=\Pi_k$, $R_L^*=R_k$
Return Z^*, Π_L^*, R_L^*

Appendix 3

Procedure *Performance_Evaluation* $(\boldsymbol{V},\boldsymbol{Q}_c,\boldsymbol{Q}_s)$

Assumption: $\boldsymbol{Q}_c$ has n products, and the number of operations for product k is O_k
Initialization
  $j=1$, /* iteration number*/
  For each product k, set its initial interval for search.
    $L_{jk}=0$, $U_{jk}=O_k$, $1\leqslant k\leqslant n$
    $I_{jk}=[L_{jk},U_{jk}]$, $1\leqslant k\leqslant n$
  Identify the longest route /* for terminating the following While loop*/
    $h=\operatorname{Arg\,Max}_{1\leqslant k\leqslant n}O_k$
While $\{j=1 \text{ or } (m_{2h}-m_{1h})\leqslant 1\}$
  Step 1: Determine the two cut-off points for each segment I_{jk}
    $m_{1k}=\lfloor(U_{jk}+L_{jk})/4\rfloor,\quad 1\leqslant k\leqslant n$
    $m_{2k}=\lfloor 3(U_{jk}+L_{jk})/4\rfloor,\quad 1\leqslant k\leqslant n$
  Step 2: Generate all possible combinations of cut-off points
    $S_j=\{\Pi|\Pi=(\pi_1,\ldots,\pi_n),\ \text{where } \pi_k=m_{1k} \text{ or } \pi_k=m_{2k}\}$
  Step 3: Identify the best combination of cut-off points from S_j
    Set $H_1=\phi$, $H_2=\phi$
    For each $\Pi\in S_j$
      Call LP_Module (V,Π,Q_c,Q_s)
      If (Pass_Check = "Pass"), put $Z(V,\Pi)$ in H_1 and $R(V,\Pi)$ in H_2
    Endfor
  Step 4: Check if there exist a solution in S_j
    If $(H_1\neq\phi)$, then $\Pi^*=\operatorname{ArgMin}_{\Pi\in H_1}Z(V,\Pi)$ and $R^*=R(V,\Pi^*)$
    If $(H_1=\phi)$, then Pass_Check = "Fail", **Return**
  Step 5: Update the interval for each product k
    If $(\pi_k^*=m_{1k})$ then $L_{j+1,k}=L_{j,k}$, $U_{j+1,k}=\lfloor(U_{jk}+L_{jk})/2\rfloor$, $1\leqslant k\leqslant n$
    If $(\pi_k^*=m_{2k})$ then $L_{j+1,k}=\lfloor(U_{jk}+L_{jk})/2\rfloor$, $U_{j+1,k}=U_{j,k}$, $1\leqslant k\leqslant n$
    $j=j+1$
Endwhile
  Optimal_Cutoff $(V)=\Pi^*$
  Optimal_Route_Ratio $(V)=R^*$
  Optimal_Objective_value $(V)=Z(V,\Pi^*)$
  Pass_Check (V) = Pass_Check
Return

References

Binh, Q. D., & Lan, P. N. (2007). Application of a genetic algorithm to the fuel reload optimization for a research reactor. *Applied Mathematics and Computation, 187*, 977–988.

Chiang, D., Guo, R. S., Chen, A., Cheng, M. T., & Chen, C. B. (2007). Optimal supply chain configurations in semiconductor manufacturing. *International Journal of Production Research, 45*(3), 631–651.

Connors, D. P., Feigin, G. E., & Yao, D. D. (1996). A queueing network model for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 9*(3), 412–427.

Dimopoulos, C. (2006). Multi-objective optimization of manufacturing cell design. *International Journal of Production Research, 44*(22), 4855–4875.

Kim, C. O., Beak, J. G., & Jun, J. (2005). A machine cell formation algorithm for simultaneously minimizing machine workload imbalances and inter-cell part movements. *International Journal of Advanced Manufacture Technology, 26*, 268–275.

Lee, Y. H., Chung, S., Lee, B., & Kang, K. H. (2006). Supply chain model for the semiconductor industry in consideration of manufacturing characteristics. *Production Planning and Control, 17*(5), 518–533.

Mahdavi, I., Rezaeian, J., Shanker, K., & Amiri, Z. R. (2006). A set partitioning based heuristic procedure for incremental cell formation with routing flexibility. *International Journal of Production Research, 44*(24), 5343–5361.

ManMohan, S. S. (2005). Managing demand risk in tactical supply chain planning for a global consumer electronics company. *Production and Operations Management, 14*(1), 69–79.

Nsakanda, A. L., Diaby, M., & Price, W. L. (2006). Hybrid genetic approach for solving large-scale capacitated cell formation problems with multiple routings. *European Journal of Operational Research, 171*, 1051–1070.

Spiliopoulos, K., & Sofianopoulou, S. (2007). Manufacturing cell design with alternative routings in generalized group technology: Reducing the complexity of the solution space. *International Journal of Production Research, 45*(6), 1355–1367.

Toba, H., Izumi, H., Hatada, H., & Chikushima, T. (2005). Dynamic load balancing among multiple fabrication lines through estimation of minimum inter-operation time. *IEEE Transactions on Semiconductor Manufacturing, 18*(1), 202–213.

Vin, E., Lit, P. D., & Delchambre, A. (2005). A multiple-objective grouping genetic algorithm for the cell formation problem with alternative routings. *Journal of Intelligent Manufacturing, 16*, 189–209.

Wu, M. C., & Chang, W. J. (2007). A short-term capacity trading method for semiconductor fabs with partnership. *Expert Systems with Application, 33*(2), 476–483.

Wu, M. C., Chen, C. F., & Shih, C. F. (2008). Route planning for two wafer fabs with capacity-sharing mechanisms. *International Journal of Production Research*. doi:10.1080/00207540802172029.

Wu, S. D., Erkoc, M., & Karabuk, S. (2005). Managing capacity in the high-tech industry: A review of literature. *The Engineering Economist, 50*, 125–158.