Optimizing Movement Sequences for Step-and-Scan Lithography Equipment

Yuji SHINANO**, Nobuo INUI***, Youzou FUKAGAWA† and Noburu TAKAKURA†

** Zuse Institute Berlin, JST CREST
Takustrasse 7, D-14195, Berlin-Dahlem, Germany
E-mail: shinano@zib.de

*** October Sky Co., Ltd.
Zelkova Bldg 1-25-12, Fuchucho, Fuchu-shi, Tokyo 183-0055, Japan

† CANON INC., Optical Products Operations, Semiconductor Production Equipment Technical Promotion Dept. 2
20-2 Kiyohara-Kogyodanchi, Utsunomiya-shi, Tochigi 321-3292, Japan

Abstract
The purpose of this work is to improve the throughput of step-and-scan lithography equipment to shorten the production time of a wafer. For this purpose, we propose a method for solving the MSOP (Movement Sequence Optimization Problem), which is the problem of computing the fastest schedule for visiting all shots on a wafer. It is well-known that the MSOP on step-and-repeat lithography equipment can be modeled as a traveling salesman problem. In contrast to step-and-repeat lithography equipment, a schedule for step-and-scan lithography equipment must also indicate the scanning direction of each shot, in addition to the sequence of the shots. For this reason, the traveling salesman problem formulation for step-and-repeat lithography equipment cannot be applied to solve the MSOP on step-and-scan lithography equipment directly. We overcame this difficulty by introducing auxiliary vertices to model the scanning directions in the traveling salesman problem formulation. By this method, we were able to compute exact optimal sequences considering the scanning directions of shots for several MSOP instances. Our numerical experiments demonstrated that our proposed method was capable of computing exact optimal solutions for real-world MSOP instances having up to 232 shots on a wafer. These optimal solutions gave a 0.25% to 4.66% improvement in productivity over solutions computed by previously known methods.

Key words: Optical Instrument, Precision Instrument, Scheduling, Semiconductor Lithography, Traveling Salesman Problem, Optimization

1. Introduction

The productivity of semiconductor fabrication of step-and-scan lithography equipment can be increased by shortening the production time of a wafer. In this paper, this is achieved by optimizing the movement sequences of the reticle and wafer stages of the equipment. Fig. 1 shows an overview of the equipment and illustrates a situation where a pattern on the reticle is being projected onto the wafer. In Fig. 1, the wafer stage can move in any direction on a horizontal flat surface, but the reticle stage can only move in front-and-back directions. The pattern is projected when the reticle stage and the wafer are in a synchronized scan state. In such a moment, a rectangle region on the wafer is exposed by the pattern on the reticle; we call such a rectangle region a shot. In this paper, we will use upward scanning to refer to an exposure of a shot where the movement of the reticle stage is to the front and downward scanning when referring to an exposure during a movement of the reticle stage to the back.
A process for one wafer is as follows. At first, the wafer stage moves to the substrate attachment position to set up the wafer. Next, alignment marks, which are marked on the wafer in the previous process, are measured by the alignment microscope to align the wafer. Then, one by one, all shots are exposed. Finally, the wafer stage moves to the substrate detachment position to remove the wafer.

The positions and the number of shots are predefined for each wafer. In order to improve the productivity of semiconductor fabrication of the equipment, the time to expose all shots on the wafer should be minimized. Physical movement speeds for all parts of the equipment are almost critical and are hard to accelerate. Therefore, we propose a method to generate a sequence of movements for all positions of the substrate attachment, all alignment marks, all shots, and the substrate detachment that minimizes the process time for one wafer. We call the positions wafer setting up position, alignment measuring positions, shot exposure positions and wafer removing position, respectively. For each movement to a shot, we must decide whether the scanning of the shot is upward or downward, because the moving time to the shot depends on the scanning direction. In this paper, we define the Movement Sequence Optimization Problem (MSOP) to be the problem of determining the fastest sequence of the movements from the substrate attachment position to the substrate detachment position taking into account the scanning directions of all shots.

The step-and-repeat lithography equipment projects a pattern on the reticle onto the wafer without the reticle stage movement. For this equipment we do not have to take into account the scanning direction of shots and the optimization problem can be formulated as a Traveling Salesman Problem (TSP) (see Section 3) directly as proposed by Miyaji[4]. For step-and-scan lithography equipment, a genetic algorithm was proposed by Yoshida[5]. However, the genetic algorithm does not provide a bound on the solution quality and is not guaranteed to produce optimal solutions. In this paper, we transform instances of the MSOP to equivalent instances of the TSP and then solve the resulting TSPs using the powerful TSP solver Concorde[1]. The TSP has been studied very deeply[3] and successfully applied to many problems in industry (for an example in manufacturing processes, see Fujimura et al.[2]). Our computational experiments demonstrate that this method enables us to find optimal solutions to real-world MSOP instances in an amount of time that is reasonable for use in practice.

The rest of this paper is organized as follows: In Section 2, we give a formal definition of the MSOP. In Sections 3 and 4, we give a short introduction to the TSP and transform the MSOP to the TSP. In Section 5, we propose a method to solve the MSOP and show computational results. In Section 6, we conclude the paper.

2. Movement Sequence Optimization Problem (MSOP)

In order to specify an instance of the MSOP, four kinds of positions must be given: a
wafer setting up position, the alignment measuring positions, the shot exposure positions, and a wafer removing position, as illustrated in Fig. 2. A digraph is created with vertices corresponding to these positions: we have a beginning vertex \( b \), alignment vertices, shot vertices, and a terminal vertex \( t \), respectively, as illustrated in Fig. 3. For the MSOP specification, arcs are created between a pair of vertices if a movement is possible between their corresponding positions. A cost is also assigned to each arc equal to the movement time from the tail position to the head position. A movement sequence is represented by a sequence of these vertices connected by arcs. Between each pair of shot vertices, there are four parallel arcs in one and four parallel arcs in the opposite directions. The selected arcs must be consistent with respect to the scanning directions of each shot vertex.

A permissible movement sequence starts from the beginning vertex \( b \). Then, it visits alignment vertices. The alignment vertices are classified into \( G \) groups (\( G \geq 1 \)) according to measuring accuracy. Here, we denote them \( A_g, g \in \{1, \ldots, G\} \) and the accuracy increases from \( A_1 \) to \( A_G \). Each group has a given number of vertices. The sequence must visit vertices from the lower accuracy measuring groups before visiting any vertices from a higher accuracy measuring group. Only after visiting all vertices in \( A_g \) is the movement sequence allowed to visit a vertex in \( A_{g+1} \). Therefore, the beginning vertex \( b \) can be connected only to vertices in \( A_1 \) and the vertices in \( A_g \) are connected to vertices in \( A_g \) and to vertices in \( A_{g+1} \). The movement sequence can visit a shot vertex only after it has visited all vertices in \( A_G \).

Let \( S \) be the set of shot vertices. For each shot vertex, a scanning direction has to be decided. In Fig. 3, each shot vertex is drawn by a pentagon which indicates its scanning direction. After visiting all shot vertices, the movement sequence terminates at the terminal vertex \( t \).

The following costs depending on movement times are assigned to each arc.

- \( w_{bj} \): from the beginning vertex \( b \) to vertex \( j \in A_1 \)
- \( w_{ij} \): from vertex \( i \in A_g \) to vertex \( j \in A_{g+1}, g = 1, \ldots, G - 1 \)
- \( w_{ij} \): from vertex \( i \in A_g \) to vertex \( j \in A_g \setminus \{i\}, g = 1, \ldots, G \)
- \( w_{ij}^{ui}, w_{ij}^{ud}, w_{ij}^{di}, w_{ij}^{du} \): from vertex \( i \in S \) with upward/downward scanning to vertex \( j \in S \setminus \{i\} \) with upward/downward scanning

Here, subscripts indicate vertices and superscripts indicate scanning directions \( u(“up”) \) and \( d(“down”) \).
or $d$ ("down").

Note that a movement between the alignment vertices can be done only by wafer stage movement, so $w_{ij} = w_j$, for a vertex $i \in A_g$ and a vertex $j \in A_g \setminus \{i\}$. However, a movement to or from a shot vertex can be done by both wafer stage movement and reticle stage movement, so the cost of vertex $i$ to a vertex $j$ is not always the same as that of vertex $j$ to vertex $i$, where either $i$ or $j$ is a shot vertex.

The MSOP is the problem of finding the minimum cost path from the beginning vertex $b$ to the terminal vertex $t$ visiting all vertices in $A_1, \ldots, A_G$ (respecting the constraint that all vertices in $A_g$ must be visited before visiting any vertex in $A_{g+1}$), after that, visiting all vertices in $S$ and determining their scanning directions, and finally moving to the terminal vertex $t$. To every possible movement, that is, for each arc, we associate a variable, which takes value 1 if the arc is in the path and 0, otherwise. Then, the MSOP can be formulated as the following Integer Linear Program (ILP):

\[
\text{minimize} \quad \sum_{j \in b} w_{bj} x_{bj} + \sum_{g=1}^{G-1} \sum_{i \in A_g, j \in A_{g+1}} w_{ij} x_{ij} + \sum_{g=1}^{G} \sum_{i \in A_g} w_{ij} x_{ij} + \sum_{i \in A_g} w_{ij} x_{ij} \\
\quad + \sum_{i \in S} (u_i x_{ii}^u + u_i^d x_{ii}^d) \\
\text{subject to} \quad \sum_{j \in b} x_{bij} = 1, \quad (2) \\
\quad \sum_{j \in A_g \setminus \{i\}} x_{ij} = 1, \quad i \in A_g, \quad g = 1, \ldots, G - 1, \quad (3) \\
\quad \sum_{j \in A_g} x_{ij} + \sum_{j \in S \setminus \{i\}} x_{ij} = 1, \quad \forall i \in A_1, \quad (4) \\
\quad \sum_{j \in A_g} x_{ij} = 1, \quad i \in A_g, \quad g = 2, \ldots, G, \quad (5) \\
\quad \sum_{j \in A_g} x_{ij} + \sum_{j \in S \setminus \{i\}} x_{ij} = 1, \quad \forall j \in S, \quad (6) \\
\quad \sum_{j \in A_g} x_{ij} = 1, \quad \forall j \in A_g, \quad g = 1, \quad (7) \\
\quad \sum_{j \in A_g} x_{ij} + \sum_{j \in S \setminus \{i\}} x_{ij} = 1, \quad \forall j \in S, \quad (8) \\
\quad \sum_{j \in S} x_{ij} = 1, \quad (9)
\]

Fig. 3 An Example of an Optimal Solution for an MSOP
Equations (2)–(5) say that, for each vertex except \( t \), exactly one movement has to be chosen to a vertex connected by outgoing arc. Equations (6)–(9) say that, for each vertex except \( h \), exactly one movement has to be chosen from a vertex connected by incoming arc. Equations (10) say that exactly one movement has to be chosen from one vertex in \( A_\ell \) to a vertex in \( A_{\ell+1} \). Equation (11) says that exactly one movement has to be chosen from one vertex in \( A_G \) to a vertex in \( S \). Equations (12) forbid cycles which contain the vertices in \( A_g \) for all \( g \) and equations (13) forbid cycles which contain the vertices in \( S \). Equations (14) to (19) impose to keep consistency of scanning direction of each shot.

3. Traveling salesman problem (TSP)

The Traveling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem and has been studied very thoroughly\(^{(3)}\). Given a list of cities and their pairwise distances, the goal of the TSP is to find a shortest possible tour that visits each city exactly once. The TSP can be classified as STSP (Symmetric TSP) or ATSP (Asymmetric TSP) depending on whether the pairwise distances between cities are symmetric (the distance from a city \( i \) to a city \( j \) is equal to that from city \( j \) to city \( i \)) or asymmetric (the distance from a city \( i \) to a city \( j \) is not necessarily equal to that from city \( j \) to city \( i \)).

Let \( G_{ATS P} = (V, A) \) be a digraph where \( V \) is a set of \( n \) vertices and \( A \) is a set of arcs. Let \( C = (c_{ij})_{i,j \in A} \) be a distance matrix associated with \( A \) and define variables \( x_{ij} \in \{0, 1\} \) for all \( (i, j) \in A \). Each variable takes 1 if the arc is in the tour and 0, otherwise. The following ILP
formulation of ATSP is well-known:

\[
\text{minimize } \sum_{i,j \in A} c_{ij}x_{ij} \\
\text{subject to } \sum_{j \in A} x_{ij} = 1, \quad \forall j \in V, \\
\sum_{i \in A} x_{ij} = 1, \quad \forall i \in V, \\
\sum_{i,j \in A} x_{ij} \leq |K| - 1, \quad \forall K \subseteq V, K \neq \emptyset, K \neq V, \\
x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A.
\]

For STSP, \(c_{ij} = c_{ji}\), thus, we define a set of edges \(E\) and then define variables \(x_{ij}\) only for \((i, j) \in E, i < j\). Let \(G_{STSP} = (V, E)\) be an undirected graph. For a vertex \(i\), we define the set of edges \(\delta(i) = \{u, v \in E | u = i \text{ or } v = i\}\). For a vertex set \(K\), we denote \(E(K)\) as a set of edges with both ends in \(K\). The following ILP formulation of STSP is also well-known:

\[
\text{minimize } \sum_{(i, j) \in E} c_{ij}x_{ij} \\
\text{subject to } \sum_{j \in \delta(i)} x_{ij} = 2, \quad \forall i \in V, \\
\sum_{i \in E(K)} x_{ij} \leq |K| - 1, \quad \forall K \subseteq V, K \neq \emptyset, K \neq V, \\
x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in E.
\]

From the modeling point of view, ATSP is more general than STSP, because an STSP can be modeled as an ATSP by making each edge of the STSP into two parallel arcs with opposite directions having the same cost as the original undirected edge. Actually, as you can see the ILP formulation of MSOP, modeling MSOP as ATSP is quite natural. However, from the solution algorithm point of view, more practical efforts have been focused on solving STSP instances, and more mature software is available for this variation of the TSP. As a result, we have very effective solution procedures and computer codes for the STSP, some of which are designed to solve the TSP to optimality (codes for exact solutions) and some designed to find approximate solutions. There are several ways to transform an ATSP to an equivalent STSP (see p.195 of the book)\(^{(3)}\). Because of this, it is often advantageous to solve an ATSP by transforming it to an equivalent STSP and to use the best STSP solver in practice to solve hard TSP instances. In Section 4, we give several MSOP-specific transformations that transform MSOP directly to STSP.

Many of the exact TSP solvers use branch-and-cut algorithms based on Linear Programming (LP). MSOP can be formulated as the ILP and can be solved using an LP-based branch-and-cut algorithm designed for the MSOP-specific implementation. However, in general, in order to make a very sophisticated implementation, we would have to implement many of the same algorithms as a state-of-the-art TSP solver. Therefore, in this paper, the MSOP instances are transformed to equivalent STSP instances, which can then be solved directly by a state-of-the-art TSP solver.

4. MSOP modeled as STSP

In this section, we explain how the MSOP is modeled as an STSP. Here, the STSP model is explained by referencing an example in Fig. 4. The example has alignment groups \(A_1 = \{1, 2\}, A_2 = \{3, 4\}\), and a shot vertex set \(S = \{1, 2\}\). In the example, the symbol \(\uparrow\) represents a shot vertex and the symbol \(\bigcirc\) represents a vertex of the STSP. In order to model the MSOP as an STSP, each shot is represented by several vertices which model the decision about the scanning direction of that shot. The vertex \(b\) is the beginning vertex, the vertex \(i\) is the terminal vertex, and the vertices \(a_i, i \in A_g, g \in \{1, \ldots, G\}\) are the alignment vertices. The vertex \(d\) is an auxiliary dummy vertex. In order to represent the scanning direction of a shot, each shot vertex \(i \in S\) has vertices \(S_I^i, S_O^i\) and \(SO^i, SIO^i\) that represent \textit{upward scanning and}
downward scanning entering points and upward scanning and downward scanning exiting points of the shot, respectively. An edge \([u, v]\) is given between vertices \(u\) and \(v\), when a movement between them is allowed. For each \(u \in S\) to each \(v \in S, u \neq v\), four edges connect a vertex \(u \in \{SO_i, SO_j\}\) to a vertex \(v \in \{SI_i, SI_j\}\) with movement costs between \(i\) and \(j\), taking into account their scanning directions. Each shot vertex \(i \in S\) also has dummy vertices \(SD_i, SD_j, SD_k, SD_l\) which are connected to \(SI_i, SI_j, SO_i, SO_j\) by edges with appropriate costs to ensure that a valid path is made from \(SI_i\) to \(SO_i\) on the STSP solution tour. In this paper, we give three STSP modes: the eight-vertices-in-a-shot STSP model, the six-vertices-in-a-shot STSP model, and the four-vertices-in-a-shot STSP model.

4.1. The beginning vertex and the terminal vertex

The beginning vertex \(b\) and the terminal vertex \(t\) of the MSOP have a one-to-one correspondence to that of the STSP models. MSOP solution should take only one edge connected to the beginning and the terminal vertices on an STSP solution tour. This means that the two vertices have to be connected by a path with only dummy edges and dummy vertices on the STSP solution tour. In order to always construct such an STSP solution tour, a dummy vertex \(d\) and dummy edges \([b, d], [d, t]\) with their costs 0 are introduced.

4.2. Alignment vertices

The alignment vertices of the MSOP have a one-to-one correspondence to that of the STSP models. An alignment vertex \(i\) is represented by \(a_i\) as in Fig. 4(a). An edge between the beginning vertex \(b\) and vertex \(i \in A_1\) should be taken on the solution tour of the STSP models, because the tours always have an edge \([b, d]\). Then, the cost of the edge between the beginning vertex \(b\) and vertex \(i \in A_1\) is given by \(w_{bi}\) directly. On a feasible solution of MSOP, only after all vertices in \(A_1\) are visited, it is allowed to visit a vertex in \(A_2\). Therefore, the solution tour of the STSP model must contain a path that is composed entirely of vertices of \(A_1\) and a path which is composed entirely of vertices of \(A_2\), and so forth. On a feasible
solution of the MSOP, only after all alignment vertices are visited can a shot vertex be visited. To ensure each tour has this property, an additional cost $a$ is added to the cost of each edge between any vertex $i \in A_g$ and any vertex $j \in A_{g+1} (g = 1, \ldots, G - 1)$ and to the cost of any edge between a vertex $i \in A_G$ and a vertex $j \in S$. By choosing $a$ large enough, we ensure that any optimal tour will avoid choosing these edges as much as possible, enforcing the desired precedence constraint regarding the vertices in $A_g$ and the shot vertices. We give a way to decide the value of $a$ in Subsection 4.4.

4.3. Shot vertices

In the eight-vertices-in-a-shot STSP model, a shot has eight vertices of STSP which are connected as in Fig. 4(a). $SD_1^1, SD_1^2, SD_2^1, SD_2^2$ restrict a path composed of vertices within a shot. Having these vertices connected as in Fig. 4(a), the possible path within a shot on a solution tour of STSP is one of two cases: $\{S I_j^1, S D_1^1, S O_1^1, S D_1^2, S D_2^1, S I_j^1, S D_2^2, S O_1^1\}$ or $\{S I_j^1, S D_1^1, S O_1^1, S D_1^2, S D_2^1, S I_j^1, S D_2^2, S O_1^1\}$. The path of the first case indicates downward scanning of the shot and that of the second case indicates upward one. The computational difficulty of STSP increases in proportion to the number of circular permutations of vertices in general. Since most of the vertices of the MSOP are the shot vertices, it is desirable to reduce the number of vertices used to model each shot. Figs. 4(b),(c) show the six-vertices-in-a-shot STSP model and the four-vertices-in-a-shot STSP model respectively. In the six-vertices-in-a-shot STSP model, the scanning direction of the shot is indicated by taking either an edge $\{S I_j^1, S O_1^1\}$ or an edge $\{S I_j^1, S O_1^2\}$ to the optimal solution tour edges. In order to enforce the requirement that each shot is only entered once, a penalty cost of $a$ is added on the edges from vertices within the shot to vertices outside of the shot. In the four-vertices-in-a-shot STSP model, the scanning direction of a shot is indicated by taking either an edge $\{S I_j^1, S O_1^1\}$ or an edge $\{S I_j^1, S O_1^2\}$ to the tour edges. As in the previous model, a penalty cost of $a$ is added to all edges from vertices within a shot to vertices outside of the shot; additionally a penalty cost $a$ is also added to edges $\{S I_j^1, S O_1^1\}$ and $\{S I_j^1, S O_1^2\}$ to ensure that at most one is taken. It is possible to make three vertices in a shot or two vertices in a shot STSP models to represent scanning directions of a shot. However, in these models, $S I_j^1$ and $S O_1^1$, and, $S I_j^1$ and $S O_1^2$ are not distinguished. Therefore, the application of the models are restricted to the cases that the moving times between shot $i$ and shot $j (i \neq j)$ are always $u_{ij}^{dd} = u_{ji}^{ad}$, $u_{ij}^{dd} = u_{ji}^{ad}$, and $u_{ij}^{dd} = u_{ji}^{ad}$.

4.4. How to decide the penalty cost $a$

In order to ensure the validity of the six-vertices-in-a-shot STSP model and the four-vertices-in-a-shot STSP model to solve the MSOP, the additional cost $a$ must be chosen appropriately. Adding the cost $a$, a solution for the MSOP converted from an optimal solution tour of the STSP will always be a feasible solution of MSOP. From MSOP point of view, the optimal solution tour of the STSP requires the following property: exactly one tour edge connects between alignment groups, a shot has only two tour edges which are connected to the outside vertices of the shot, and for four-vertices-in-a-shot STSP model, the tour edges have only five edges (three edges with cost $a$) which are connected to vertices within the shot. All the optimal solution tours of an STSP have the following property: the tour has at least one edge between the alignment groups and a shot has at least two edges which are connected to the outside vertices of the shot, and for the four-vertices-in-a-shot STSP model, the tour edges have at least five edges (at least three edges with cost $a$) which are connected to vertices within a shot. This observation leads to a strategy to choose the cost $a$ so that the optimal tour for the STSP has the minimum number of edges with cost $a$. The minimum number of edges with cost $a$ in the STSP tour is a fixed value which depends on the number of alignment groups and the number of shots. Let $M$ be the minimum number of edges with added cost $a$ in a tour, then $M = G$ for the eight-vertices-in-a-shot model, $M = G + |S| - 1$ for the six-vertices-in-a-shot model and $M = G + 2|S| - 1$ for the four-vertices-in-a-shot model. Therefore, depending on
the STSP model, let the shortest moving time of the MSOP be $Y^*$, the costs of these tours will become:

- Eight-vertices-in-a-shot model: $Y^* + aG$
- Six-vertices-in-a-shot model: $Y^* + a(G + |S| - 1)$
- Four-vertices-in-a-shot model: $Y^* + a(G + 2|S| - 1)$

Assume that a solution tour of the STSP has $N$ edges with cost $a$. If $N = M$ holds, the solution tour of the STSP can be transformed to a feasible solution of the MSOP. Let the optimal value of the STSP models with $a = 0$ be $\overline{Y}$. Then, the following proposition holds.

**Proposition 1** A solution transformed from a feasible solution of MSOP is a feasible solution of the STSP models.

**Proof** All the STSP models have edges between all feasible combinations among the beginning vertex, the terminal vertex, alignment points, and shots of the MSOP. This proposition holds (Q.E.D).

**Proposition 2** If $a > Y^* - \overline{Y}$, then an optimal solution of any of the three STSP models gives an optimal solution of the MSOP.

**Proof** Let $Y + aN$ be the optimal value of an STSP model. For any $N \geq M + 1$, a solution of the STSP models with its objective function value of $Y^* + aM$ such that $Y^* + aM < Y + aN$ can be transformed to a feasible solution of MSOP. Since the lower bound of $Y$ is $\overline{Y}$ and the minimum value of $N$ is $M + 1$, an optimal solution of the STSP models gives a feasible solution of the MSOP when the cost $a$ holds $Y^* + aM < \overline{Y} + a(M + 1)$. Proposition 1 shows that the feasible solution set of the MSOP is included in the feasible solution set of any of the three STSP models. Therefore, the optimal solution of any of them gives an optimal solution of the MSOP (Q.E.D).

This proposition shows that an optimal solution of the MSOP can be found to solve the STSP models by giving a sufficient large cost $a$. However, it is impossible to obtain $Y^*$ in advance and $\overline{Y}$ is also hard to be determined because finding the value requires solving an STSP. Therefore, in order to obtain the cost $a$, we used a fast heuristic algorithm to solve the MSOP. Let $Y^*$ be a total moving time obtained by solving the heuristic algorithm. In this paper, we set the cost $a = Y^*$ because $Y^* > Y^* - \overline{Y}$ holds. In order to obtain $Y^*$, we use a simple nearest neighbor search algorithm. Imposing constraints to make a feasible solution of MSOP, from the beginning vertex $b$, it visits all alignment vertices and all shots and the terminal vertex $t$ exactly once. When the algorithm runs, visited vertices are marked and the next visiting vertex from the current visiting vertex always chooses the shortest movement time one among unmarked vertices.

5. Computational experiments

Computational experiments were conducted on a PC with an Intel Core 2 quad, 2.8GHz CPU and 4GB of memory. The STSP solvers used were Concorde to solve it to optimality together with the Linkern heuristic. Our computational experiments had two goals: The first one was to compare the computation time of the different STSP models. The second one was to evaluate the necessity of finding an exact optimal solution to the STSP. For this purpose, we used two kinds of STSP solvers. We prepared four instances which have a relatively large number of shots and are selected from real-world data. (For confidentiality reasons, the number of instances is small, but they accurately represent problems that are of interest in practice.) Table 1 shows the specification of the original instances including their symmetry of movement time (symmetric indicated by “sym” or asymmetric indicated by “asym”). Tables 2 and 3 show whether the movement times between shot $i$ and shot $j$ are the same or not depending on combinations of the movement direction between shots and the scanning directions of the shots. The symbol “=” means that it is always the same and the letters “UK”
Table 1 Data Specification (number of alignments and shots, symmetry of movement time)

| Instance No. | \(b\) | \(|A_1|\) | \(|A_2|\) | \(|S|\) | \(t\) | Movement time |
|--------------|------|-------|-------|------|-----|-------------|
| I1           | 1    | 2     | 1     | 12   | 232 | sym         |
| I2           | 1    | 2     | 12    | 232  | 1   | asym        |
| I3           | 1    | 2     | 12    | 308  | 1   | sym         |
| I4           | 1    | 2     | 12    | 308  | 1   | asym        |

Table 2 Symmetry between Shots for I1 and I3

<table>
<thead>
<tr>
<th>shot (f) → shot (j)</th>
<th>(u \rightarrow u)</th>
<th>(u \rightarrow d)</th>
<th>(d \rightarrow u)</th>
<th>(d \rightarrow d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u \rightarrow u)</td>
<td>UK</td>
<td>UK</td>
<td>UK</td>
<td>UK</td>
</tr>
<tr>
<td>(u \rightarrow d)</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>(d \rightarrow u)</td>
<td>UK</td>
<td>=</td>
<td>=</td>
<td>UK</td>
</tr>
<tr>
<td>(d \rightarrow d)</td>
<td>=</td>
<td>UK</td>
<td>=</td>
<td>UK</td>
</tr>
</tbody>
</table>

Table 3 Asymmetry between Shots for I2 and I4

<table>
<thead>
<tr>
<th>shot (f) → shot (j)</th>
<th>(u \rightarrow u)</th>
<th>(u \rightarrow d)</th>
<th>(d \rightarrow u)</th>
<th>(d \rightarrow d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u \rightarrow u)</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>(u \rightarrow d)</td>
<td>UK</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>(d \rightarrow u)</td>
<td>UK</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>(d \rightarrow d)</td>
<td>=</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

means it is not always the same. As we described in Subsection 4.3, it is not necessary to distinguish between \(S T^I_r\) and \(SO^I_r\) and between \(S T^O_r\) and \(SO^O_r\) in the symmetric movement time type. Therefore, the MSOP with symmetric movement time type can be modeled as an STSP with fewer vertices and is considered a relatively easy problem to solve.

For each MSOP instance, the three STSP models’ instances were generated and they were solved by Concorde and Linnerr. Table 4 shows the results for Concorde. In a box for each model’s column of Table 4, the top row number is computing time with the number of tasks (the number of nodes explored in the branch-and-bound algorithm) shown in parentheses, the middle row number is the optimal value of the MSOP (the minimum process time for one wafer), and the bottom row number is improvement ratio between the initial solution found by the heuristic algorithm to obtain the cost \(a\) described in Subsection 4.4 and the optimal value. When the STSP is solved to optimality, all the solutions generated from the three models’ instances give the same optimal value. For the instance I4, none of the STSPs generated from the three models could be solved in 24 hours. Therefore, we show only the number of tasks and the MSOP process time computed from the best incumbent solution at the time limit. The results for I1 show that the eight-vertices-in-a-shot model can be solved fast, even if the STSP solution space is much larger than that of the others. Solution time to optimality is not always fast in the smaller STSP. However, in general, the results show that a smaller number of vertices in a shot model has advantages. It is reasonable to find a better feasible solution by searching in a smaller STSP solution space than a larger one, if the solution time is the same for the two instances. Table 4 shows that the improvement ratio between 0.25% and 4.66% with respect to the initial solutions is obtained in these instances. The heuristic algorithm for obtaining the initial solution is basically the same method that has been used so far by humans. (We can see that it is difficult for humans to generate the optimal solution by hand due to the complexity of the instances as illustrated in Fig. 3.) The optimal values show that we could improve throughput per wafer by up to 4.66%. This improvement ratio is significant in mass production.

Table 5 shows the results obtained using Linnerr and uses the same format as Table 4. Table 5 shows that the solution time is extremely fast in general, although the improvement ratio from the initial solution is only between 0.03% and 2.95%. It is natural that instances with fewer vertices solve more quickly. However, the quality of the solutions does not clearly depend on the model used. It is better to solve instances generated by several models and to choose the best result. How to obtain the optimal solutions for large scale problems is a
Table 4 Optimal Solution of MSOP

<table>
<thead>
<tr>
<th>model</th>
<th>init.sol. (sec.)</th>
<th>comp.time(task) and sol. (sec.), ratio(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 vertices</td>
<td>6 vertices</td>
</tr>
<tr>
<td>I1 sol. ratio</td>
<td>66.963</td>
<td>691(174)</td>
</tr>
<tr>
<td>I2 sol. ratio</td>
<td>62.682</td>
<td>5924(994)</td>
</tr>
<tr>
<td>I3 sol. ratio</td>
<td>72.444</td>
<td>6864(819)</td>
</tr>
<tr>
<td>I4 sol. ratio</td>
<td>67.325</td>
<td>2506(539)</td>
</tr>
</tbody>
</table>

Table 5 Sub-Optimal Solution of MSOP by Linkern

<table>
<thead>
<tr>
<th>model</th>
<th>init.sol. (sec.)</th>
<th>comp.time and sol. (sec.), ratio(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 vertices</td>
<td>6 vertices</td>
</tr>
<tr>
<td>I1 sol. ratio</td>
<td>66.963</td>
<td>0.749</td>
</tr>
<tr>
<td>I2 sol. ratio</td>
<td>62.682</td>
<td>0.728</td>
</tr>
<tr>
<td>I3 sol. ratio</td>
<td>72.444</td>
<td>1.117</td>
</tr>
<tr>
<td>I4 sol. ratio</td>
<td>67.325</td>
<td>1.109</td>
</tr>
</tbody>
</table>

question for future research.

6. Conclusion

We have defined the MSOP and proposed a procedure to solve it by transforming it to an equivalent TSP instance. Using a powerful TSP solver, we demonstrated that exact optimal solutions to real-world MSOP instances could be computed in a reasonable amount of time and that high quality approximate solutions could be computed very quickly. Computational experiments showed that the wafer processing speed could be improved by 0.25% to 4.66% compared to previous methods.

Acknowledgement

We are deeply grateful to Prof. William J. Cook of the Georgia Institute of Technology, who allowed us to use Concorde for this research. We thank Daniel Steffy and Ambros Gleixner for their willingness to improve the readability of the paper. We would like to thank the two anonymous referees for providing us with constructive comments and suggestions.

References