A PARTITIONING COLUMN APPROACH FOR SOLVING LED SORTER MANIPULATOR PATH PLANNING PROBLEMS

Sheng-I Chen* and Yen-Che Tseng

Department of Industrial Engineering and Management
National Yang Ming Chiao Tung University
Hsinchu, 30010, Taiwan

Department of Industrial Engineering and Management
National Chiao Tung University
Hsinchu, 30010, Taiwan

(Communicated by Oleg Prokopyev)

ABSTRACT. This study considers the path planning problem of picking light-emitting diodes on a silicon wafer. The objective is to find the shortest walk for the sorter manipulator covering all nodes in a fully connected graph. We propose a partitioning column approach to reduce the original graph’s size, where adjacent nodes at the same column are seen as a required edge, and the connection of vertices at different required edges is viewed as a non-required edge. The path planning problem turns to find the shortest closed walk to traverse required edges and is modeled as a rural postman problem with a solvable problem size. We formulate a mixed-integer program to obtain the exact solution for the transformed graph. We compare the proposed method with a TSP solver, Concorde. The result shows that our approach significantly reduces the problem size and obtains a near-optimal solution. For large problem instances, the proposed method can obtain a feasible solution in time, but not for the benchmarking solver.

1. Introduction. This study developed an algorithm using the idea of grouping nodes in columns to transform original problems into arc routing problems. Our approach significantly reduces problem sizes and obtains positive results in both runtimes and solution qualities as benchmarking with a TSP solver, Concorde. The applications can be in the light-emitting diode (LED) die picking and other industries concerned with manipulator path planning problems.

An effective path for manipulators can reduce the processing time for LED sorters. We consider the operation of transferring LEDs at the same quality level from the wafer to the container (known as blue tapes in the semiconductor manufacturing industry). The decision is to determine the picking sequence to minimize manipulator traveling distance. We propose a method to reduce the problem size to obtain a quality solution in time.

Prior studies have investigated the path planning problem in various fields. Sheng et al. focus on the path planning for eye-in-hand robot manipulators, where the

2020 Mathematics Subject Classification. 05A18, 05C45, 90B06, 90C11.

Key words and phrases. Light-emitting diode sorter; Manipulator path planning; Mixed-integer programming; Rural postman problem; Traveling salesman problem.

The study is supported by MOST, Taiwan grant 109-2221-E-009-065-

* Corresponding author: Sheng-I Chen.
problem is modeled as a clustered traveling salesman problem (CTSP), and approximation algorithms are developed to obtain a suboptimal solution in time [24]. Kala et al. apply evolutionary algorithms for solving the robot path planning problem in a static environment [17]. A comprehensive report of CTSP variants and algorithms can be found in the book by Lawler et al. [18]. To the best of our knowledge, there is no literature that explicitly investigated the manipulator path planning problem for LED sorters. A close study applies the simulation model to assess mechanical component positions to evaluate LED sorter performances [25].

The LED picking problem can be defined via notations in graph theory. Each diode refers to a vertex, and a feasible move of manipulator between any pair of diodes corresponds to an arc. The objective is to find a simple tour that minimizes the total cost or distance, and such a problem setting is a special case of a TSP. Since the general TSP is intractable, it is unlikely to find the optimal tour for a normal size problem instance. To tackle such a computational challenge, we develop a transformation approach to consolidate LED diodes as components. Then, the path planning decision turns to find a minimal cost tour that traverses each component at least once in the transformed graph. Despite the transformed problem still in the NP-complete class, its problem size is significantly reduced and provides a promising direction to obtain a quality solution in time.

The remaining sections of this paper are organized as follows. The path planning problem for diode picking is discussed in section 2. The proposed partitioning column method is described in section 3. The mixed-integer programming (MIP) model of the transformed problem is presented in section 4. Computational results are presented in section 5. The last section contains discussion and conclusions.

2. The path planning problem of led sorters. The sorter processes LEDs of a specific grade (or quality level) at a time. After completion, the sorter starts over again to process LEDs at the next level. In order to minimize the completion time, the manipulator moves and accesses LEDs for each grade level in the shortest path manner, and that this problem is essentially a TSP among the required diodes.

We consider one manipulator installed on a sorter. Only the manipulator’s movement is taken into account for the objective function. This is because the container for holding sorted LEDs (known as blue tapes) moves along with the manipulator, and thus no additional time spent on placing LEDs on containers. Also, we ignore the time to move the manipulator to the position where the wafer locates and returns the manipulator since both operations’ duration is fixed and does not affect path planning.

Figure 1 illustrates the spatial distribution of different level LEDs on a wafer. There are six subfigures, and each of them refers to a set of diodes with the same quality level, where (Figure 1a) has the fewest diodes, and (Figure 1f) has the most diodes. Each point on the figure is a LED according to the actual we obtained.

Next, we utilize the graph representation to define the path planning problem. Each LED diode is a vertex, and the connection between a pair of diodes is an edge. The collection of diodes is denoted as a vertex set $V$. It is possible to move the manipulator from one vertex to another, and thus, the edge set $E$ contains all pairs of vertices (that is $(i, j) \in E$ and $i, j \in V$) in the complete graph $G = (V, E)$. The traveling time of any pair of diodes $(c_{ij})$ is symmetric and can be either Euclidean or Manhattan distance. Let $x_{ij}$ be the binary decision variable to the edge that connects vertices $i$ and $j$. The edge is selected in the solution when the decision
Figure 1. The problem instances of LED distribution

value is equal to one, and unselected when the value is zero. The objective is to determine a minimum distance walk passing through each vertex in the graph (minimize $\sum_{(i,j) \in E} c_{ij}x_{ij}$).

The complexity of the path planning problem can be analyzed by comparing it with TSP. The TSP is to seek the shortest tour through each vertex exactly once (that is $\sum_{i \in V} x_{ij} = 1, \forall j \in V$ and $\sum_{j \in V} x_{ji} = 1, \forall i \in V$), whereas the path planning problem does not restrict to the one-time visiting. Relaxing the one-time visiting is known as the graphic TSP, which is only solvable when the graph is in a serial-parallel structure [7, 14]. By lack of such a specific structure, the path planning problem is distinct from these solvable cases and remains in the NP-hardness class.

A number of methods have been developed for solving large-scale TSPs. One of the classical methods may refer to the Lin-Kernighan heuristic that starts from an initial tour and then iteratively exchanging selected edges with unselected ones to improve the objective value [19]. Also, a fast but straightforward procedure to find feasible solutions may refer to the nearest-neighbor heuristic [22]. This approach starts with a selected node and then find an unconnected node closest to selected nodes until all nodes are connected. Applications and computational analysis of the nearest-neighbor method may refer to [2, 23]. For solving metric TSP, a well-known approximation algorithm may refer to the Christofides’ algorithm, which has been proven to obtain a feasible solution at most 1.5 times the optimum [3].

In the aspect of exact algorithms, Crowder and Padberg implement a cutting-plane approach for sub-tour elimination during the branch-and-bound procedure
Other scholars tackle symmetric TSP based on the branch-and-cut algorithm [21]. Additionally, Applegate et al. implement a method based on cutting plans for solving problem instances with more than 1,000,000 nodes [1]. Later, Cook integrates both heuristics and exact algorithms and develops a tool (known as Concorde) to solve problem instances as large as 85,900 nodes [5]. In this study, we report computational performances between our method with Concorde. The result shows that although our problem instances are relatively small, considering fractions for distances has led to difficulties for Concorde to find optimal solutions in time.

3. Transformation procedure. This section explains the procedure to transform the path planning problem to the edge routing one. The purpose is to obtain a compact problem with reductions on both vertices and edges. Before the transformation, a preprocessing step is required to separate LEDs at different quality levels. Each level can be seen as a problem instance and solved separately. Additionally, the preprocessing step will prepare parameter values, and their definitions are described as the following table.

| Notations  | Description                                                                 |
|------------|-----------------------------------------------------------------------------|
| $M_u$      | The $y$-position of the uppermost diode                                      |
| $M_l$      | The $y$-position of the lowermost diode                                      |
| $r$        | The maximum mismatch ratio                                                   |
| $n_x$      | The number of LEDs at the $x$-th column                                     |
| $\theta$  | The threshold of empty positions in the column ($\theta = \lceil n_x r \rceil$) |
| $\bar{b}$ | The lower bound of the upper component (initialized as $M_u$)                |
| $b$        | The upper bound of the lower component (initialized as $M_l$)                |
| $X$        | The collection of diode’s $y$-position                                       |
| $Y$        | The set of grades of diodes                                                 |
| $g$        | A function mapping $X$ into $Y$                                             |

3.1. Transforming diodes to columns. This subsection describes the procedure and reasons to transform the node routing problem to the arc routing one. Let $(x,y)$ be the coordinate of each LED relative to the lower and left corner. In the transformed graph, LEDs in the same column are transformed into an edge. Let $M$ be the set of columns or edges. Each of them contains LEDs located at the same horizontal position, where the number of diodes at the column ($n_x$) is determined using the wafer mapping data, as well as the vertices of each column are labeled accordingly.

The advantage to transform diodes to columns is that each edge is merely the shortest distance to traverse all LEDs on it, and thus, the overall runtime for the algorithm can be shortening by avoiding finding the shortest path for each component.

The quality level for each LEDs is uncertain and usually associated with the distance of the diode to the central point on the wafer. Since the distribution of same grade LEDs tends to in the ring-shaped, the manipulator should avoid passing through the central area. We consider both the manipulator’s efficiency and LED coverage. The path planning ensures that covered LEDs are more than the minimum
requirement while the manipulator passes through invalid points (diodes not of the same grade or bad diodes) as few as possible. We propose a grouping scheme in a column orientation. A column may further split into two segments if it contains too many invalid points. There are different ways to aggregate LEDs. For example, one may think of grouping LEDs in the same row. Either using row or column orientation for grouping vertices is empirical. Since the data we collected has fewer columns than rows, the advantage of grouping LEDs in a column orientation is to obtain a smaller transformed problem. The following figure illustrates the solution for the first problem instance (Fig 1a) obtained by our algorithm. Note that the path is optimal for the transformed problem, but not for the original one. There are two LEDs discarded. The manipulator would travel a much longer distance to recover the missing LEDs.

3.2. Partitioning columns. The objective of partitioning columns during the transformation procedure is to determine whether a column should keep as a piece or split into two segments. To obtain the minimal cost path, one may identify all segments in each column, and then solve for the optimal Euler tour to connect all of them. However, the transformed problem can be oversized, which results in difficulties in solving the corresponding rural postman problem (RPP). The advantage of our transformation is to obtain a manageable problem size for RPPs by identifying at most two segments in each column. Such a decomposition scheme works quite well when the distribution of LEDs is ring-shaped. Columns in the middle of the wafer are separated as upper and lower components, while columns at both sides remain unpartitioned.

The algorithm starts with setting parameter values $\bar{b}$ as the upper vertex’s $y$–position $M_u$, and $\bar{b}$ as the lower vertex’s $y$–position $M_l$. A parameter of the maximum mismatch ratio $(r)$ is predetermined as the threshold of the percentage of invalid diodes in a column. The notation $\theta$ refers to the threshold of mismatch diodes in a column, and its value is defined as multiplying $n_x$ by $r$ and then rounding
up to the nearest integer. When $\theta$ decreases to zero or the counter $i$ equals the number of diodes in the column, the algorithm terminates.

Line 2 of the algorithm is to check whether the selected column should be separated or not. If it contains excessive invalid points, the column is partitioned by labeling the lower boundary of the upper component ($\bar{b}$) and the upper boundary of the lower component ($b$). Otherwise, $\bar{b}$ and $b$ would be denoted as the upper boundary and lower boundary for the column respectively.

Next, lines 5 to 11 are to find the upper component lower bound along a downward direction ($d = True$). The index variable $i_1$ used to record the current position. When the current location has found the same grade diode, the lower bound of the upper component $\bar{b}$ will set as $i_1 - 1$. Otherwise, the search direction changes from downward to upward ($d = False$). A similar procedure applies for finding the upper boundary for the lower component (lines 13-21 in table 2).

**Table 2. The procedure to find component boundaries**

| Line | Description |
|------|-------------|
| 1.   | Set $\bar{b} = i_1 = M_u$, $b = i_2 = M_1$, $i = 0$, $\theta = \lceil n_x r \rceil$, and $d = True$; |
| 2.   | If $(M_u - M_1 + 1) - n_x > \theta$ |
| 3.   | While $i < |n_x|$ and $\theta > 0$ |
| 4.   | If $d = True$ then |
| 5.   | If $g(i_1) = g(i_1 - 1)$ then |
| 6.   | $\bar{b} := i_1 - 1$; |
| 7.   | Else |
| 8.   | $d = False$; |
| 9.   | $\theta - -$; |
| 10.  | End if |
| 11.  | $i_1 - -$; |
| 12.  | Else |
| 13.  | If $g(i_2) = g(i_2 + 1)$ then |
| 14.  | $\bar{b} := i_2 + 1$; |
| 15.  | Else |
| 16.  | $d = True$; |
| 17.  | $\theta - -$; |
| 18.  | End if |
| 19.  | $i_2 + +$; |
| 20.  | End if |
| 21.  | $i + +$; |
| 22.  | End while |
| 23.  | End if |

The above procedure is performed to discover discontinued points for each columns. There are $|M|$ columns on a wafer, and each column requires $n_x$ or fewer operations to find discontinued points on it. The procedure will execute the most operations
when setting $r$ to 0%. In contrast, the runtime will be shorter as setting a greater parameter value.

The following example illustrates how the partitioning procedure works. There are fifteen diodes in the column, where the rectangular diodes must be picked and the x-shaped diodes to be discarded. Assuming the maximum mismatch ratio sets as 0.3, and $n_x$ is equal to 10. Thus, the threshold of empty positions in the column ($\theta$) is equal to 3. Since $(M_u - M_l + 1) - n_x = 5$ is greater than the threshold, the column will be partitioned into the upper and lower components.

![Initialization and resulting graph](image)

**Figure 3.** The example for partitioning columns

The parameter of maximum mismatch diodes is used for balancing the solution quality and LED diode coverage. A greater value may decrease the solution quality due to containing invalid points in the partitioned column. In contrast, a smaller value may result in the central LEDs being discarded, and the worst case is missing $n_x - 2$ LEDs. In such a case, the manipulator only picks LEDs located at the extreme points of the column ($M_u$ and $M_l$) while ignores the others located at the center area. Figure 4 illustrates the worst case, where the column length is 14, the number of valid diodes ($n_x$) is 10, and the maximum mismatch ratio is set as 0.3. Our algorithm will stop as the cumulative invalid points reached the threshold of 4 ($\theta = \lceil n_x r \rceil = \lceil 10 \times 0.3 \rceil = 4$). In the first iteration, the lower bound of the upper component ($\tilde{b}$) is determined through lines 5-11 in table 2. Since the top two LED diodes in the column have different grades (i.e., $g(14) \neq g(13)$), $\tilde{b}$ remains unchanged ($\tilde{b} = M_u = 14$), and $\theta$ is decreased to 3 (line 9). In the second iteration, the algorithm changes to find the upper bound of the lower component ($\hat{b}$) through lines 13-21. Since the first and second diodes have different grades (i.e., $g(1) \neq g(2)$), $\hat{b}$ remains unchanged ($\hat{b} = n_x = 1$), and $\theta$ is decreased to 2. The third iteration is similar to the first iteration. Finally, the algorithm terminates at the fourth iteration as $\theta$ dropped to 0. The upper component’s lower bound is 14, and the lower component’s upper bound is 1. As a result, only the highest and lowest diodes in the column are selected, while the rest diodes located at central section are discarded.

4. **Mixed-integer programming model.** The MIP model presented in this section is based on the following notation. Let $C_1, C_2, \ldots, C_m$ be the components of the transformed graph $G_T$ obtained from the partitioning column procedure. A
Figure 4. The worst case of discarded diodes

component contains an edge to be visited in the feasible path at least once, known as the required edge. We denote the set of all required edges by $E_R$. Next, we define a non-required edge set $E_A$ by pairing vertices from two different components. The feasible path may not include every non-required edge. In fact, the non-required edge will be visited when the manipulator moves from one component to another. Let $V_R$ be the set of vertices on required edges. There is no non-required vertex in the transformed graph. The transformed graph is then denoted as $G_T = (V_T, E_T)$, where $E_T = E_A \cup E_R$ and $V_T = V_R$.

The RPP is to seek a minimum-cost closed walk that traverses the subset of edges. The first formulation provides by Christofides et al. [4]. Readers may refer to the comprehensive review of RPP in the published literature [11, 9, 10]. The mathematical programming model for RPP has improved by tightening the LP relaxation [6]. Still, such formulation contains an excessive number of the connectivity constraint, and it is impractical to apply directly for real-world problems. Fernández et al. provide a compact formulation with a tighten bound [13]. We apply their works to construct the mathematical programming model for solving the transformed problem.

A general procedure of the RPP algorithm requires additional steps of adding the shortest path of any pair of vertices at different components to the graph [11]. Since the graph we considered is completely connected and its edge distances satisfied the triangular inequality, the shortest path between each pair of vertices is simply the edge connected them. As a result, our algorithm can perform in faster by saving time on computing the all-pairs shortest path problem.

Also, it is necessary to ensure that all vertices are even degree to obtain a closed walk on the transformed graph. Let $V_o$ be the subset of required vertices with odd numbers of required edges (R-even vertices). After adding the connective edge in the graph, the R-odd degree vertex will turn to an even degree while the R-even degree vertex will become an odd degree. To ensure all vertices in even degree, one may duplicate the R-odd degree vertices only. However, duplicating R-even degree vertices would increase problem sizes and thus affect the computational performance. The advantage of our proposed method is that every component contained one required edge and both required vertices and R-odd degree. Therefore, there is no need to copy any required vertices due to the odd-degree property [16].

Next, we determine whether non-required edges should be added to the Euler tour or not to ensure all components are connected. The objective value can be seen as the additional cost of traversing invalid points for manipulator. Also, the overall traveling distance is equal to the required edge distance plus objective value.
Let \( \delta(j) \) be the set of non-required edges connected to vertex \( j \). The binary decision variable \( x_{ij} \) is to determine whether the edge \((i,j)\) is selected or not. The continuous variable \( y_{kl} \) refers to the flows sending from \( C_k \) to \( C_l \). It works along with constraint (3)-5 to ensure that all components are connected. The MIP model is formulated as the following.

\[
\text{Minimize} \quad \sum_{(i,j) \in E_A} c_{ij} x_{ij} \quad (1)
\]

\[
\text{Subject to}
\]
\[
\sum_{(i,j) \in \delta(i)} x_{ij} = 1, \quad \forall i \in V_T \quad (2)
\]
\[
\sum_{k=2,...,m} y_{1k} = m - 1 \quad (3)
\]
\[
\sum_{k \neq l} (y_{lk} - y_{kl}) = 1, \quad \forall t \in 2, ..., m \quad (4)
\]
\[
y_{kl} \leq (m - 1) \sum_{(i,j) \in E_A, i \in C_k \text{ and } j \in C_l} x_{ij} \quad (5)
\]
\[
y_{kl} \leq 0, \quad \forall k, l \in 1, ..., m, k \neq l \quad (6)
\]
\[
x_{ij} \in 0, 1, \quad \forall (i,j) \in E_A \quad (7)
\]

The objective function (1) is to minimize the total cost of selected edges. Constraint (2) is to restrict that the selected edge flowing out from each vertex is exactly equal to one. The constraints of (3) to (5) are to ensure that all components are connected, where (3) is used for sending out \( m - 1 \) flows from the depot, (4) is to ensure that each component can only receive one unit flow, and (5) is to ensure that any two components are connected if their flow variable is greater than zero. The MIP solution ensures that all components are connected, and then the resulting walk is closed. The decision variable refers to the edge to traverse in addition to required edges. Thus, the total distance for picking LEDs is equal to the objective value plus the required edge length. After adding the selected edges to the graph, the Eulerian cycle can be obtained easily using the end-pairing algorithm [12].

We further analyze the optimal solution of the MIP. Let \( \alpha^*(i) \) be the optimal solution to the incident edges for the vertex \( i \) in the transformed graph \( G_T \). According to the equation (2), \( \alpha^*(i) \) is equal to one for every vertex in \( V_T \). Furthermore, there are no R-even degree vertices and \( V_T = V_R \). This implies that the optimal solution connects every vertex in \( V_T \) exactly once. This result is similar to the finding in Lemma 5 by Garfinkel et al. [16].

### 5. Computational result

This section reports computational performances by comparing the proposed transformation method and the TSP formulation. We arbitrarily select six grades of LEDs from a wafer for the experiment. The small-size problem instances include Grade1 and Grade2, and each of them has less than a hundred diodes. The medium-size problem instances are Grade3 and Grade4, which contains 118 and 396 diodes for each of them. The Grade5 involves of 1849 diodes with approximately 3.4 million edges. The Grade6 has the most diodes among different grades in the dataset we have, where the size of the edge set exceeds 15 million (Table3).

We perform the parameter tuning to determine the maximum mismatch ratio for the proposed algorithm. The result is given in Figure 5, where each line represents
problem sizes and reduction percentage for a problem instance. We examine the algorithm performance using different parameter values, starting from 0.01 to 0.65. As a result, the number of coverage diodes increases when using a greater value of \( r \) in the algorithm. The transformed graph can cover more than 80% of LEDs in a wafer when \( r \) is equal to or greater than 30%. There is an insignificant difference in coverage diodes for any value of \( r \) greater than 30%. Therefore, we set \( r \) to 30% in the algorithm to obtain the transformed graph.

![Figure 5. The coverage percentages of diodes in transformed graphs using different maximum mismatch ratios](image)

Table 3 provides information about the original and transformed problem sizes for selected instances. The second and third columns are the numbers of vertices and edges of the original problems. The third and fourth columns refer to the numbers of vertices and edges in the transformed graphs. The last two columns are the reduction percentages of vertex and edges after the transformation. The numbers of vertices range from 30 to 226 for the transformed problems, and the numbers of edges are between 885 to 50,963. The transformation algorithm can reduce problem sizes from most problem instances, except for the Grade2 instance. Since there are no adjacent diodes in the Grade2 instance for grouping (Figure 1b), the number of components in the transformed graph is the same as the number of vertices in the original problem. The transformed graph size is greater than the original one due to each component consists of two required vertices. The partitioning column approach yields a higher reduction percentage for large-scale problems. For example, there are 96% vertex reduction 100% edge reduction in the Grade6 instances.

In the following section, we analyze the reduction percentage when using different maximum mismatch ratios. The vertex reduction percentage defines as \( 1 - |E_T|/|E| \). Tables 4 and 5 show the results of the vertex reduction percentage and edge reduction percentage, respectively. Gray-color shading represents the reduction percentage unchanged over different values of \( r \). As a result, the graph size decreases when using a greater parameter value for most problem instances. Only Grade2 instance receives a negative reduction percentage, which refers to the problem size is increased instead of decreased after the partitioning. In the Grade3 instance, reduction percentages are indifferent across all minimum mismatching limits. When we set \( r \) to 0.3 or any value greater than 0.3, both vertex and edge reduction percentages are unchanged.
Table 3. The size of the original and transformed problems (maximum mismatch ratio = 0.3).

| Instance | Original problem size | Transformed problem size | Reduction |
|----------|----------------------|--------------------------|-----------|
|          | $|V|$ | $|E|$ | $|V_r|$ | $|E_r|$ | $1 - |V_r|/|V|$ | $1 - |E_r|/|E|$ |
| Grade1   | 48     | 2,256       | 30   | 870   | 38% | 61% |
| Grade2   | 74     | 5,402       | 108  | 11,556 | -46% | -114% |
| Grade3   | 118    | 13,806      | 80   | 6,320  | 32% | 54% |
| Grade4   | 396    | 156,420     | 126  | 15,750 | 68% | 90% |
| Grade5   | 1,849  | 3,416,952   | 226  | 50,850 | 88% | 99% |
| Grade6   | 3,921  | 15,370,320  | 164  | 26,732 | 96% | 100% |

Table 4. Vertex reduction percentages in transformed graphs using different mismatch ratios.

| Instance | $r = 0.01$ | $r = 0.05$ | $r = 0.1$ | $r = 0.15$ | $r = 0.2$ | $r = 0.25$ | $r = 0.3$ | $r = 0.33$ | $r = 0.35$ | $r = 0.5$ | $r = 0.65$ |
|----------|-------------|-------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| Grade1   | 33%         | 33%         | 33%        | 38%        | 38%        | 38%        | 38%        | 38%        | 38%        | 38%        | 38%          |
| Grade2   | -46%        | -46%        | -46%       | -46%       | -46%       | -46%       | -46%       | -46%       | -46%       | -46%       | -46%         |
| Grade3   | 32%         | 32%         | 32%        | 32%        | 32%        | 32%        | 32%        | 32%        | 32%        | 32%        | 32%          |
| Grade4   | 67%         | 67%         | 67%        | 67%        | 67%        | 68%        | 68%        | 68%        | 68%        | 68%        | 69%          |
| Grade5   | 87%         | 87%         | 87%        | 87%        | 87%        | 88%        | 88%        | 88%        | 88%        | 88%        | 88%          |
| Grade6   | 95%         | 96%         | 96%        | 96%        | 96%        | 96%        | 96%        | 96%        | 96%        | 96%        | 96%          |

Table 5. Edge reduction percentages in transformed graphs using different maximum mismatch ratios.

| Instance | $r = 0.01$ | $r = 0.05$ | $r = 0.1$ | $r = 0.15$ | $r = 0.2$ | $r = 0.25$ | $r = 0.3$ | $r = 0.33$ | $r = 0.35$ | $r = 0.5$ | $r = 0.65$ |
|----------|-------------|-------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| Grade1   | 55%         | 55%         | 55%        | 61%        | 61%        | 61%        | 61%        | 61%        | 61%        | 61%        | 61%          |
| Grade2   | -114%       | -114%       | -114%      | -114%      | -114%      | -114%      | -114%      | -114%      | -114%      | -114%      | -114%       |
| Grade3   | 54%         | 54%         | 54%        | 54%        | 54%        | 54%        | 54%        | 54%        | 54%        | 54%        | 54%          |
| Grade4   | 89%         | 89%         | 89%        | 89%        | 90%        | 90%        | 90%        | 90%        | 90%        | 90%        | 90%          |
| Grade5   | 98%         | 98%         | 98%        | 98%        | 98%        | 98%        | 98%        | 98%        | 99%        | 99%        | 99%          |
| Grade6   | 100%        | 100%        | 100%       | 100%       | 100%       | 100%       | 100%       | 100%       | 100%       | 100%       | 100%         |

Table 6 compares the computational performance of using the partitioning column approach versus modeling the original problems as a TSP based on the Miller-Tucker-Zemlin (MTZ) formulation [20]. We solve the MIP via the branch-and-cut algorithm on the commercial solver, IBM CPLEX version 12.6. The computational experiment was performed under the environment of Intel i5-6200U CPU with 12.0 GB memory. The runtime limit set to 7,200 seconds, and the tolerance for the MIP gap set to $10^6$. The result shows the optimal RPP solution obtained for every problem instance, in which the runtime is less than a second for solving the Grade1 problem instance, and 6,356 seconds for solving the largest problem instance (Grade6). Using the MTZ formulation can construct the first three smallest problem instances only, but not for others. For the first problem instance, the MIP gap is 42%, and the total cost is slightly higher than the one discovered by using our proposed method. The optimal solution for the Grade2 problem instance is obtained in 184 seconds. For the Grade3 problem instance, the approach of using MTZ formulation can only find a suboptimal solution of the MIP gap 61% within the time limit.
Table 6. Computational results of solving the original and transformed problems.

| Instance | Partitioning column | MTZ formulation |
|----------|---------------------|-----------------|
|          | Run time(s) | Total cost | MIPGap | Run time(s) | Total cost | MIPGap |
| Grade1   | < 1        | 227       | 0%     | 7,200      | 229       | 42%    |
| Grade2   | 14        | 386       | 0%     | 184        | 386       | 0%     |
| Grade3   | 8         | 393       | 0%     | 7,200      | 444       | 61%    |
| Grade4   | 78        | 660       | 0%     | –          | –         | –      |
| Grade5   | 260       | 2,078     | 0%     | –          | –         | –      |
| Grade6   | 6,356     | 4,205     | 0%     | –          | –         | –      |

This subsection investigates a TSP solver in order to compare it with the proposed algorithm. Concorde has implemented both exact and heuristic methods for solving large-scale problem instances [5]. In the benchmarking, we use the exact method to explore the solver’s capability. The solution obtained by the tool may differ from the optimum because of the fractional distance is rounded to the nearest integer. To investigate the rounding effect, we perform the testing of using the original data versus inflating the original coordinate by 10,000 times and then dividing the objective value by 10,000 to obtain an approximate solution with four decimal places accuracy. Table 7 displays computational results for problem instances using the Euclidean metric for computing distance. When rounding the fractional distance to the nearest integer, Concorde is capable of solving small problem instances (Grades 1 to 4) within 5 seconds, and the largest problem instance in 42 seconds. As we inflate the distance by 10,000 times, Concorde can find the optimal solution for the smallest four problem instances within 18 seconds, but not for others within the time limit of 7,200 seconds. We define the rounding error (%) by the total cost of four-decimal accuracy divided by the total cost of merely rounding to the nearest integer, and then minus by one. The rounding error (%) is defined as the objective value for the problem using four-decimal distances divided by the solution for the problem merely rounding to the nearest integer, and then minus by one. As one can see, there is no rounding error for the first three grades, but the fourth instance is 4%.

Table 7. Comparing the performance of Concorde in solving problem instances with integral and fractional edge distances.

| Instance | Concord (with rounding to the nearest integer) | Concorde (with four decimal place) |
|----------|-----------------------------------------------|-----------------------------------|
|          | Run time(s) | Total cost | Rounding Error % | Run time(s) | Total cost |
| Grade1   | < 1        | 227       | 0%               | 1           | 227       |
| Grade2   | < 1        | 386       | 0%               | 1           | 386       |
| Grade3   | 1          | 393       | 0%               | 1           | 394       |
| Grade4   | 5          | 582       | 4%               | 18          | 605       |
| Grade5   | –          | –         | –                | –           | –         |
| Grade6   | 42         | 3,926     | –                | –           | –         |
Table 8. The benchmark of Concorde and the proposed algorithm using Manhattan and Euclidean distances.

| Instance | Euclidean distance | Manhattan distance |
|----------|--------------------|-------------------|
|          | Concorde          | Proposed algorithm | Concorde | Proposed algorithm |
| Grade1   | 227               | 227               | 258      | 258               |
| Grade2   | 386               | 386               | 462      | 462               |
| Grade3   | 394               | 393               | 430      | 428               |
| Grade4   | 605               | 660               | 692      | 742               |
| Grade5   | –                 | 2,078             | –        | 2,154             |
| Grade6   | –                 | 4,205             | –        | 4,286             |

Table 8 displays the objective value obtained by Concorde and the proposed algorithm. We consider Euclidean and Manhattan metrics. Both of them are commonly used in manipulator path planning. We report the Concorde solutions with the accuracy of four decimal places. Obviously, using the Manhattan metric obtains a greater objective value than the Euclidean metric for each problem instance. The results using Euclidean for problem instances 1 to 3 are close, where both Concorde and our proposed algorithm obtain the same costs for instances 1 and 2. There is a tiny difference in problem instance 3 because the partitioning algorithm ignores diodes in the middle of the column when the number of invalid points exceeding the maximum mismatch threshold. In grade 4, the partition algorithm solution is larger than Concorde by 9%. The excess cost is due to invalid points in the required edges. In grades 5 and 6, Concorde cannot find the optimal solution, and we report the solutions from the proposed algorithm only. A similar observation holds when computing distance via Manhattan metric.

6. Discussion and conclusions. This study has proposed a solution approach dedicated to the manipulator path planning problem. We proposed a column partitioning scheme to aggregate vertices and then formulated a MIP to obtain the optimal perfect matching. Parameter tuning has been performed to identify the best setting for the algorithm.

The computational analysis shows that our algorithm obtained an optimal or close-to-optimal solution in a short time, comparing with the approach of modeling the path planning problem via MTZ can only solve for the smallest three problem instances. Additionally, we solved problem instances using a well-known TSP solver as a benchmark with our proposed methods. The result shows that, when solving medium size problem instances, our proposed method ties with the benchmarking solver in the aspects of run time and solution quality. For large problem instances (vertices more than a thousand), our method can obtain a quality solution regardless of the distance is in either Euclidean or Manhattan metric.

There are limitations to this study. The partitioning column approach dedicates for the ring-shaped diode distribution. Modifications may require for solving other distributions. Another limitation is that the size of the transformed problem can be larger than the original one if the diode distribution is sparse (such as the Grade2 problem instance). Finally, the transformed problem may be unsolvable for instances with excessive problem sizes. This study formulated RPP as a MIP and then utilized a commercial solver to solve the problem. However obtaining an
exact solution may be unnecessary in real-world practices. One may use heuristics for solving the RPP and then integrate with the partitioning column approach to tackle large-scale problems.

Acknowledgments. The authors would like to thank Yi-Nung Tsao for proofreading and reformatting the final version of the manuscript. The authors are grateful to the Ministry of Science and Technology, Taiwan (https://www.most.gov.tw/) grant 109-2221-E-009-065- for providing funding to support this study.

REFERENCES

[1] D. Applegate, R. Bixby, V. Chvátal and W. Cook, Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems, Mathematical Programming, 97 (2003), 91–153.
[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman and A. Y. Wu, An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM, 45 (1998), 891–923.
[3] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Technical Report, 388, Graduate School of Industrial Administration, CMU, 1976.
[4] N. Christofides, V. Campos, A. Corberan and E. Mota, An Algorithm for the Rural Postman Problem, Imperial College, London, England, Tech. Report, 1981.
[5] W. Cook, Concorde TSP solver, Available at: http://www.math.uwaterloo.ca/tsp/concorde/index.html. [Accessed on March 29, 2019], 2006.
[6] A. Corberan and J. M. Sanchis, A polyhedral approach to the rural postman problem, European Journal of Operational Research, 79 (1994), 95–114.
[7] G. Cornuéjols, J. Fonlupt and D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra, Mathematical Programming, 33 (1985), 1–27.
[8] H. Crowder and M. W. Padberg, Solving large-scale symmetric travelling salesman problems to optimality, Management Science, 26 (1980), 495–509.
[9] M. Drezel, On the generalized directed rural postman problem, Journal of the Operational Research Society, 65 (2014), 1143–1154.
[10] M. Dror, Arc Routing: Theory, Solutions, and Applications, Kluwer, Boston, MA, 2000.
[11] H. A. Eiselt, M. Gendreau and G. Laporte, Arc routing problems, part II: The rural postman problem, Operations Research, 43 (1995), 399–414.
[12] J. Edmond and E. L. Johnson, Matching, Euler tour and the chinese postman problem, Mathematical Programming, 5 (1973), 88–124.
[13] E. Fernández, O. Meza, R. Garfinkel and M. Ortega, On the undirected rural postman problem: Tight bounds based on a new formulation, Operations Research, 51 (2003), 281–291.
[14] J. Fonlupt and A. Nachev, Dynamic programming and the graphical traveling salesman problem, Journal of the ACM (JACM), 40 (1993), 1165–1187.
[15] R. W. Floyd, Algorithm 97: Shortest path, Communications of the ACM, 5 (1962), 345.
[16] R. S. Garfinkel and I. R. Webb, On crossings, the crossing postman problem, and the rural postman problem, Networks: An International Journal, 34 (1999), 173–180.
[17] R. Kala, A. Shukla and R. Tiwari, Robotic path planning using evolutionary momentum-based exploration, Journal of Experimental and Theoretical Artificial Intelligence, 23 (2011), 469–495.
[18] E. L. Lawler, J. K. Lenstra, A. R. Kan and D. B. Shmoys, The traveling salesman problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons, Ltd., Chichester, 1985.
[19] S. Lin and B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Operations Research, 21 (1973), 498–516.
[20] C. E. Miller, A. W. Tucker and R. A. Zemlin, Integer programming formulation of traveling salesman problems, Journal of the ACM, 7 (1960), 326–329.
[21] M. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM Review, 33 (1991), 60–100.
[22] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis, II, An analysis of several heuristics for the traveling salesman problem, SIAM Journal on Computing, 6 (1977), 563–581.
[23] J. Sankaranarayanan, H. Samet and A. Varshney, A fast all nearest neighbor algorithm for applications involving large point-clouds, Computers and Graphics, 31 (2007), 157–174.
[24] W. Sheng, N. Xi, M. Song and Y. Chen, Robot path planning for dimensional measurement in automotive manufacturing, *Journal of Manufacturing Science and Engineering*, 127 (2005), 420–428.

[25] T. Wu, B. Li, L. Wang and Y. Huang, Study on path-optimization by grade sorting dies, *In IEEE International Conference on Mechatronics and Automation*, (2010), 876–880.

Received October 2020; revised December 2020.

E-mail address: sichen@nctu.edu.tw
E-mail address: johnny553166@gmail.com