

Exact Algorithms for Weighted Rectangular Covering Problems

Ryoya Umeda[1(✉)], Wei Wu[2], Yannan Hu[1], and Hideki Hashimoto[3]

[1] Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-0825, Japan
1423504@ed.tus.ac.jp
[2] Shizuoka University, 3-5-1 Johoku Naka-ku, Hamamatsu, Shizuoka 432-8561, Japan
[3] Tokyo University of Marine Science and Technology, 2-1-6 Etchujima, Koto-ku, Tokyo 135-8533, Japan

Abstract. Given n points on a 2-dimensional plane and t types of rectangles of different sizes and weights, the weighted rectangular covering problem (WRCP) aims to cover all the points with rectangles so as to minimize the total weight of the used rectangles. The WRCP is known to be NP-hard and can be solved in $O(n^6 2^n)$ time. In this paper, we first propose a new algorithm that improves the running time to $O(\min\{n^5 2^n, tn^3 2^n\})$. We then consider a special case where each rectangle has zero width or height, in other words, we are given weighted line segments. For the weighted line-segment covering problem (WLSCP), we show that it is still NP-hard and can be solved in $O(tn2^n)$ time. We also consider the WLSCP with unit-length line segments and show the problem can be solved in polynomial time.

Keywords: rectangular covering problem · dynamic programming · minimum cost flow · NP-hardness

1 Introduction

This study focuses on the 2-dimensional *weighted rectangular covering problem* (WRCP), where we are given n points on a 2-dimensional plane and t types of rectangles, each having width, height, and weight. The WRCP asks to cover all the points by the given rectangles in an arrangement parallel to the x and y axes, and the objective is to minimize the total weight of the used rectangles. The WRCP is known to be NP-hard in the strong sense and can be solved by a dynamic programming algorithm in $O(n^6 2^n)$ time proposed by Porschen [5]. In this study, we propose a new method also based on dynamic programming, but with a better running time of $O(\min\{n^5 2^n, tn^3 2^n\})$.

We then consider a special case of the WRCP, the *weighted line-segment covering problem* (WLSCP), where points should be covered by given weighted line segments, which can be treated as rectangles having zero widths or heights. For the WLSCP, we first prove its NP hardness, and then design an algorithm with a time complexity of $O(tn2^n)$. For the special case where all the points are

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024, LNCS 14813, pp. 16–28, 2024.
https://doi.org/10.1007/978-3-031-64605-8_2

placed on integral coordinates and each line segment has a unit length (length of 1), we show this WLSCP is polynomially solvable.

2 Weighted Rectangular Covering Problem

In this section, we describe the weighted rectangular covering problem, and introduce an exact algorithm proposed by Porschen [5]. For this problem, we propose a new dynamic programming algorithm with a better running time.

2.1 Problem Definition

We are given a set of points $P = \{1, 2, \ldots, n\}$, each point i with coordinates (a_i, b_i), and a set of rectangle types $R = \{1, 2, \ldots, t\}$, each rectangle type r with width w_r, height h_r, and weight c_r. We assume that all input values are non-negative integers. The weighted rectangular covering problem (WRCP) aims to place rectangles with types in R on the plane to cover all the given points, such that the total weight of the used rectangles is minimized. In this paper, we assume that there are enough rectangles of each type, and each rectangle must be placed without rotation and each edge of the placed rectangle must be parallel to either the x-axis or the y-axis. For input values t and n, we assume that

$$t \leq n^4. \tag{1}$$

The following Lemma shows that Assumption (1) is natural.

Lemma 1. *Given an instance with $t > n^4$, there exists an optimal solution using at most n^4 types of rectangles.*

Proof. For any four points $i, j, k, l \in P$, if there exists a type r ($\in R$) such that a type-r rectangle can cover all these four points, we use r'_{ijkl} to denote the type with the smallest weight among the types that can cover the four points. Obviously, set $R' = \{r'_{ijkl} \mid i, j, k, l \in P\}$ has at most n^4 elements. For any rectangle whose type is not in R', let i', j', k' and l' be the leftmost, rightmost, uppermost, and lowermost points covered by the rectangle, and we can replace this rectangle with a rectangle of type $r'_{i'j'k'l'}$ without increasing the weight. Therefore, any feasible solution can be transformed to a solution only using rectangle types in R' without increasing the total weight. ■

In this study, we use $A = \{(x_1, y_1, r_1), (x_2, y_2, r_2), \ldots, (x_k, y_k, r_k)\}$ to represent a solution with k rectangles, where $r_j \in R$ is the type of rectangle j and $(x_j, y_j) \in \mathbb{R}^2$ indicates its bottom-left coordinates. Tuple (x_j, y_j, r_j) is called the placement of rectangle j. A point i with coordinates (a_i, b_i) is covered by a rectangle placement (x_j, y_j, r_j) if and only if

$$x_j \leq a_i \leq x_j + w_{r_j} \qquad \text{and} \qquad y_j \leq b_i \leq y_j + h_{r_j}$$

hold.

The rectangular covering has been investigated from 1980s. For the WRCP, Fowler et al. [2] demonstrated its NP-hardness, and Porschen [5] proposed an exact algorithm with time complexity of $O(n^6 2^n)$. We first in Sect. 2.2 introduce Porschen's algorithm, and then in Sect. 2.3 propose an improved method with a better running time.

2.2 Porschen's Algorithm

In this section, we introduce the exact algorithm based on dynamic programming (DP) by Porschen [5]. Let $f_j(S)$ be the minimum weight required to cover a set of points S ($\subseteq P$) using at most j rectangles, $W(A)$ be the total weight of solution A, and $C(S)$ be the set of feasible solutions to the subproblem in which the point set to be covered is S. Then, $f_j(S)$ can be formally defined as

$$f_j(S) = \min\{W(A) \mid A \in C(S), |A| = j\}.$$

For each subset S, $R(S)$ is defined as a subset of rectangular types, each can cover all the points in S, that is,

$$R(S) = \left\{ r \in R \mid \max_{i \in S} a_i - \min_{i \in S} a_i \leq w_r \text{ and } \max_{i \in S} b_i - \min_{i \in S} b_i \leq h_r \right\}.$$

The boundary value $f_1(S)$ for each S can be computed as

$$f_1(S) = \begin{cases} 0 & \text{if } S = \emptyset \\ \min\limits_{r \in R(S)} c_r & \text{otherwise.} \end{cases} \tag{2}$$

Algorithm 1. A method to generate $G(S)$.

```
1: G(S) = ∅.
2: for (p_l, p_r, p_t, p_b) ∈ S^4 do
3:     S' ← {p ∈ S | a_{p_l} ≤ a_p ≤ a_{p_r} and b_{p_b} ≤ b_p ≤ b_{p_t}}.
4:     G(S) ← G(S) ∪ {S'}.
5: end for
```

To perform dynamic programming, Porschen's method first generates $G(S)$ by Algorithm 1 using $O(|S|^5)$ time. Based on $f_1(S)$ and $G(S)$, $f_j(S)$ can be computed by the following recurrence formula:

$$f_j(S) = \min_{S' \in G(S)} \{f_1(S') + f_{j-1}(S \setminus S')\}. \tag{3}$$

The basic idea is that an optimal solution to the subproblem corresponding to $f_j(S)$ is able to be separated into one rectangle covering points in S' and the other $j-1$ rectangles covering points in $S \setminus S'$. Each $f_j(S)$ can be computed in

$O(n^5)$ time, because the size of $G(S)$ is bounded by $O(n^4)$, and the computation of $S \setminus S'$ takes linear time. Value $\min_{j=1}^{n} f_j(P)$ is the optimal value to the WRCP.

The framework of the algorithm proposed by Porschen is shown in Algorithm 2. Time to compute all values of $f_1(S)$ in Step 1 is bounded by $O(tn2^n)$, while all the other steps require $O(n^6 2^n)$ time in total. With Assumption (1), the overall running time of this algorithm is $O(n^6 2^n)$.

Algorithm 2. Porschen's algorithm.

1: Compute $f_1(S)$ for each $S \subseteq P$ according to (2).
2: Generate $G(S)$ using Algorithm 1.
3: **for** $j = 2, 3, \ldots, n$ **do**
4: Compute $f_j(S)$ for each $S \subseteq P$ based on the recurrence formula (3).
5: **end for**
6: **return** $\min_{j=1}^{n} f_j(P)$.

2.3 Proposed Method

In this section, we propose a new algorithm also based on DP but different from the method described in Sect. 2.2.

In Algorithm 1, we introduced a construction method to generate candidates $G(S)$ for solving the subproblem $f_j(S)$. In this section, we propose a new construction method in Algorithm 3 that reduces the number of elements in $G(S)$ to $O(t|S|^2)$ if $t < |S|^2$.

For every combination of $(p_{\mathrm{l}}, p_{\mathrm{b}}) \in S^2$ and $r \in R$, Algorithm 3 adds to $G(S)$ the sets of points that are covered by a rectangle r with its bottom-left coordinates at $(a_{p_{\mathrm{l}}}, b_{p_{\mathrm{b}}})$. By selecting the more efficient construction method (i.e., better one between Algorithm 1 and Algorithm 3), the size of $G(S)$ can be bounded by $O(\min\{|S|^4, t|S|^2\})$.

Algorithm 3. A new method to generate $G(S)$.

1: $G(S) = \emptyset$.
2: **for** $(p_{\mathrm{l}}, p_{\mathrm{b}}) \in S^2, r \in R$ **do**
3: $S' \leftarrow \{p \in S \mid a_{p_{\mathrm{l}}} \leq a_p \leq a_{p_{\mathrm{l}}} + w_r$ and $b_{p_{\mathrm{b}}} \leq b_p \leq b_{p_{\mathrm{b}}} + h_r\}$.
4: $G(S) \leftarrow G(S) \cup \{S'\}$.
5: **end for**

We design a new DP method, in which we consider a new type of subproblem. Let $f(S)$ be the minimum weight to cover the point set S ($\subseteq P$) without any constraint on the number of rectangles. Note that this definition simplifies the

definition of $f_j(S)$ in Porschen's algorithm by omitting the number of rectangles. Value of $f(S)$ can be computed recursively as follows:

$$f(S) = \min\left\{ f_1(S), \min_{S' \in G(S) \setminus \{\emptyset, S\}} f(S') + f(S \setminus S') \right\}. \tag{4}$$

Value of $f(P)$ implies the optimal value of the problem.

The pseudocode of the proposed approach is shown in Algorithm 4. The boundary conditions of the new approach are the same, that is, computing $f_1(S)$ according to (2). We enumerate S lexicographically to ensure that when solving a subproblem of $f(S)$, all the corresponding subproblems have been solved.

We analyse the time complexity of Algorithm 4. In Step 1, we obtain all values of $f_1(S)$ in $O(tn2^n)$ time. Step 2 takes $O(n \min\{|S|^4, t|S|^2\})$ time to generate $G(S)$. In the loop from Step 3 to Step 5, each $f(S)$ is obtained in $O(n \min\{|S|^4, t|S|^2\})$ time. Considering the enumeration of set S, the loop of Steps 3–5 is the bottleneck of this algorithm, which implies that the overall time complexity of our new exact algorithm is $O(\min\{n^5 2^n, tn^3 2^n\})$.

Algorithm 4. A new dynamic programming approach.

1: Compute $f_1(S)$ for each $S \subseteq P$ according to (2).
2: Generate $G(S)$ using the one with the better running time between Algorithm 1 and Algorithm 3.
3: **for** $i = 0, 1, \ldots, 2^n - 1$ **do**
4: Construct S by binarizing i.
5: Compute $f(S)$ based on (4).
6: **end for**

3 Weighted Line-Segment Covering Problem

In this section, we consider the weighted line-segment covering problem (WLSCP), which can be treated as a special case of the WRCP when every given rectangle type has width or height of zero (i.e. $w_r = 0$ or $h_r = 0$ holds for each $r \in R$).

For the WLSCP, we first prove that it is NP-hard in Sect. 3.1, and then propose an exact algorithm in Sect. 3.2.

3.1 Complexity Analysis

We extend the idea of Fowler [2] to prove the NP-hardness of the WLSCP.

Theorem 1. *The WLSCP is NP-hard in the strong sense.*

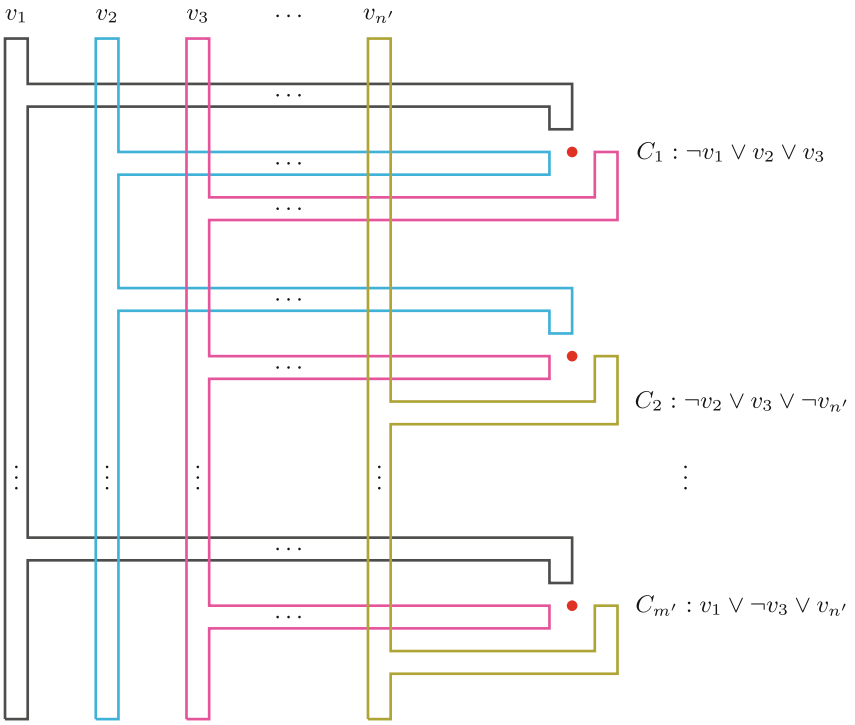


Fig. 1. A schematic overview of the constructions used in the proof of Theorem 1. (Color figure online)

Proof. We prove NP-hardness by a polynomial-time reduction from 3-SAT which is known to be NP-complete in the strong sense. We are given a 3-SAT formula Φ with n' variables $v_1, v_2, \ldots, v_{n'}$, and m' clauses $C_1, C_2, \ldots, C_{m'}$. A literal is either a variable (a positive literal) or the negation of a variable (a negative literal). A clause is a disjunction of literals (or a single literal). Each clause is a disjunction of at most three literals, where a literal is either a variable or the negation of a variable.

We transform the 3-SAT formula Φ to a WLSCP instance I_Φ with only two types of line segments:

$$t = 2; \quad w_1 = 0, h_1 = 3, c_1 = 1; \quad w_2 = 3, h_2 = 0, c_2 = 1.$$

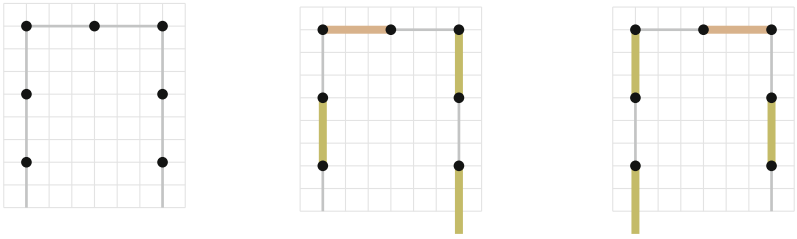
We first show how to set the points in the WLSCP instance. Figure 1 presents an overview of the construction. We call each line in the figure a *wire*, which is a linear arrangement of a set of points in the WLSCP such that any pair of consecutive points on the wire can be covered by a single line segment, while every type of line segment can only cover up to two points on the wire. Each variable in 3-SAT corresponds to a cycle (a colored cycle in Fig. 1) containing an even number of points. If the cycle of variable v_i consists of $2\gamma_i$ points,

then the minimum number of line segments required to cover the cycle is γ_i, and there are only two coverings which achieve this minimum, one representing $v_i = \mathsf{true}$, and the other representing $v_i = \mathsf{false}$. Figure 2 represents a portion of a wire, illustrating an example of the correspondence between two assignments and Boolean values. Note that no points located on the intersection of any two cycles (see Fig. 3). This property ensures that no line segment can cover three points in the intersection region.

We describe how to represent clauses and how to set up *clause points* (red points in Fig. 1 and Fig. 4), which are not belong to any cycle. For each clause in the formula there is a clause point located in a region, where the three variable cycles corresponding to the literals in the clause are brought into close proximity (see Fig. 4a). The cycles are arranged so that the clause point can be covered "for free" if at least one of the three wires is in the state corresponding to the literal in the clause (Fig. 4c). If none of the literals satisfies the clause, then an extra line segment is needed to cover the clause point (Fig. 4b). Because the distance between any two consecutive points in the cycle can be flexibly adjusted in the range of 2 to 3 (Fig. 5), all the above construction requirements can be easily satisfied.

Using such a transformation, if the formula Φ is satisfiable, then no additional line segments are needed to cover the clause points, which implies that the optimal value to the WLSCP is $\sum_{i=1}^{m'} \gamma_i$. Otherwise, if the formula Φ is unsatisfiable, the optimal value to the WLSCP is greater than $\sum_{i=1}^{m'} \gamma_i$.

By generating all the variable cycles in compact, γ_i can be bounded by $O(m'n')$, which implies that the size of instance I_Φ is bounded by polynomial with respect to m' and n', and the transformation can be done in polynomial time. ■



(a) A portion of a cycle corresponding to v_i.

(b) Assignment of $v_i = \mathsf{true}$.

(c) Assignment of $v_i = \mathsf{false}$.

Fig. 2. The correspondence between two assignments and Boolean values.

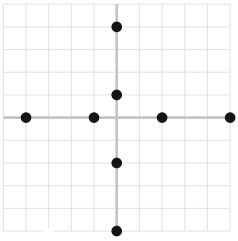


Fig. 3. No point lies at the intersection of any two cycles.

3.2 Exact Algorithm

In this section, we propose an exact algorithm for solving the WLSCP with time complexity of $O(tn2^n)$.

We first consider a special case of the WLSCP where $b_i = b_j$ holds for every pair $i, j \in P$, that is, all points are placed on a line parallel to the x-axis. We call this special case 1-dimensional WLSCP (1D-WLSCP), and we show that it can be solved in $O(n \log n + tn)$ as follows. We first sort the points by their x-coordinate values and assume that the points are numbered in non-decreasing order of x-coordinate values, that is,

$$a_1 \leq a_2 \leq \cdots \leq a_n. \tag{5}$$

Then, we show that the 1D-WLSCP can be solved by a straightforward DP approach. We define s_{ik} as

$$s_{ik} = \begin{cases} 0 & \text{if } a_1 \geq a_i - w_k \\ \max\{j \in P \mid a_j < a_i - w_k\} & \text{otherwise.} \end{cases}$$

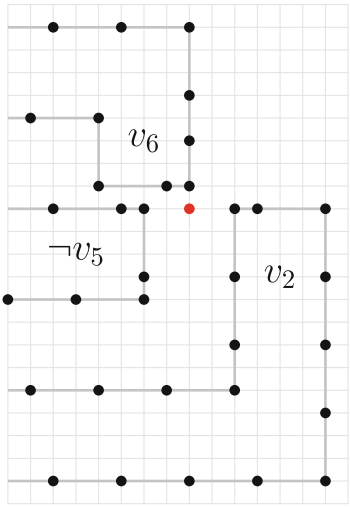
We first compute s_{ik} for all the pairs of (i, k). This computation can be performed in $O(tn)$ by using two-pointer technique [4]. Consider a subproblem of the 1D-WLSCP in which the point set are given as $\{1, 2, \ldots, i\}$, and we use $g(i)$ to denote the optimal value. By setting boundary condition

$$g(0) = 0,$$

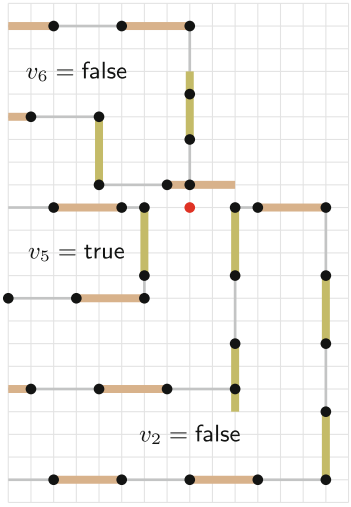
$g(i)$ for $i = 1, 2, \ldots, n$ can be computed by using the following recurrence formula:

$$g(i) = \min_{k \in R}\{g(s_{ik}) + c_k\}. \tag{6}$$

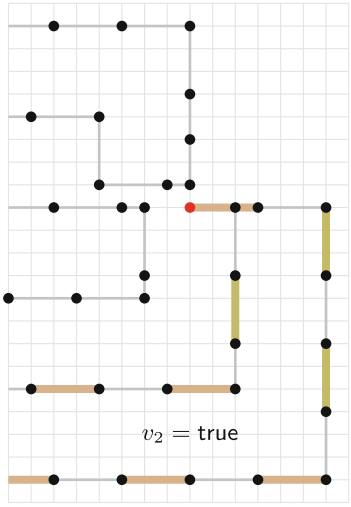
The optimal value to the 1D-WLSCP is $g(n)$, which can be computed in $O(tn)$ time. Considering the sorting process, the overall time complexity of this algorithm is $O(n \log n + tn)$. The correctness of the algorithm can be easily demonstrated through induction, as $g(i) \leq g(i + 1)$ holds for any $i \in \{0, 1, \ldots, n - 1\}$.



(a) The portion of the WLSCP instance that represents a clause, e.g., $v_2 \vee \neg v_5 \vee v_6$.



(b) The covering when $v_2 = \mathsf{false}$, $v_5 = \mathsf{true}$, and $v_6 = \mathsf{false}$.



(c) The covering when $v_2 = \mathsf{true}$.

Fig. 4. Relationship between covering and clause.

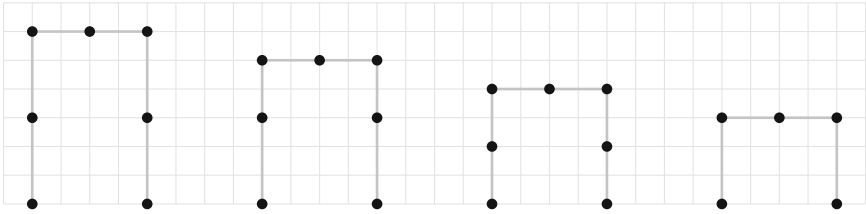


Fig. 5. Flexibility in cycle design.

Note that another variant of 1D-WLSCP with $a_i = a_j$ for every pair $i, j \in P$ can also be solved by a similar approach.

Based on the DP method for the 1D-WLSCP, we design the following divide-and-conquer method for the WLSCP. Let S be a subset of P, and we consider two subproblems $\mathbf{P}_{\mathrm{v}}(S)$ and $\mathbf{P}_{\mathrm{h}}(P \setminus S)$ of the WLSCP. In the subproblem $\mathbf{P}_{\mathrm{v}}(S)$ (resp. $\mathbf{P}_{\mathrm{h}}(P \setminus S)$), the point set is S (resp. $P \setminus S$) and the line-segment types are limited to those with $w_r = 0$ (resp. $h_r = 0$). Figure 6 shows an example with 24 points, where S is the subset containing all the black points in the middle subfigure. Both $\mathbf{P}_{\mathrm{v}}(S)$ and $\mathbf{P}_{\mathrm{h}}(P \setminus S)$ can be further divided into several 1D-WLSCPs, which can be solved by the proposed DP method. Thus, the optimal value $g_{\mathrm{v}}(S)$ of the $\mathbf{P}_{\mathrm{v}}(S)$ (resp. $g_{\mathrm{h}}(P \setminus S)$ of the $\mathbf{P}_{\mathrm{h}}(P \setminus S)$) can be obtained in $O(|S|t)$ (resp. $O(|P|t - |S|t)$) time, and the total time to solve both $\mathbf{P}_{\mathrm{v}}(S)$ and $\mathbf{P}_{\mathrm{h}}(P \setminus S)$ is bounded by $O(tn)$. Note that combining the optimal solutions to $\mathbf{P}_{\mathrm{v}}(S)$ and $\mathbf{P}_{\mathrm{h}}(P \setminus S)$ forms a feasible solution to the WLSCP. The proposed algorithm enumerates all the subsets of P and outputs the solution with the minimum weight as follows:

$$v_{\mathrm{alg}} = \min_{S \subseteq P} \{g_{\mathrm{v}}(S) + g_{\mathrm{h}}(P \setminus S)\}. \tag{7}$$

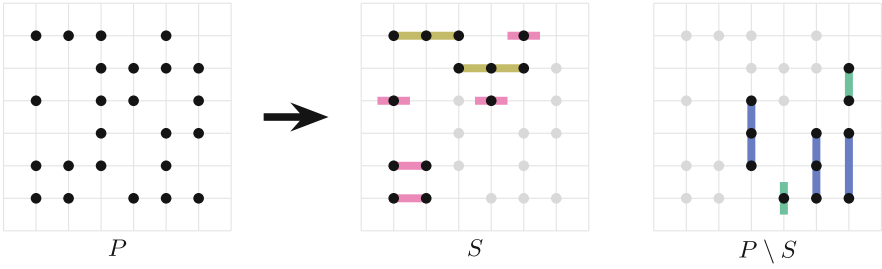


Fig. 6. An example of the partition of P into S and $P \setminus S$.

Let v^* be the optimal value to the WLSCP. To show the correctness of the proposed method, we first observe that the covering (corresponding to v_{alg}) obtained by the algorithm is feasible, and hence v_{alg} is a valid upper bound on

the optimal value v^*. Consider an arbitrary optimal covering A^* of the WLSCP, in which S^* is the point set covered by vertical line segments in A^*. Because

$$v_{\mathrm{alg}} \leq g_{\mathrm{v}}(S^*) + g_{\mathrm{h}}(P \setminus S^*) \leq v^*$$

holds, v_{alg} is also a valid lower bound on the optimal value v^*. Therefore, we have $v_{\mathrm{alg}} = v^*$, and the proposed algorithm is correct.

By lexicographically enumerating subset S, the WLSCP can be solved in $O(tn2^n)$ time, which is faster than directly applying the algorithms for the WRCP in Sects. 2.2 and 2.3.

4 Weighted Line-Segment Covering Problem with Unit-Length Line Segments

In Sect. 3.1, we proved that the WLSCP, in which the length of every line segment is 3, is NP-hard. In this section, we investigate the case when every line segment has a unit length, that is,

$$R = \{1, 2\}; \quad w_1 = 0, h_1 = 1; \quad w_2 = 1, h_2 = 0. \tag{8}$$

For the *weighted line-segment covering problem with unit-length line segments* (WLSCP-UL), we show that it is reducible to the *minimum cost flow problem* (MCFP).

In the MCFP [1], we are given a directed graph $G = (V, E)$ with a source vertex $v_{\mathrm{s}} \in V$ and a sink vertex $v_{\mathrm{t}} \in V$. Each edge $(i, j) \in E$ has an associated cost c'_{ij} that denotes the cost per unit flow on that edge. We assume that the flow cost varies linearly with the amount of flow. We also associate with each edge $(i, j) \in E$ a capacity u_{ij} that denotes the maximum amount that can flow on the edge and a lower bound l_{ij} that denotes the minimum amount that must flow on the edge. The MCFP aims to send an amount of flow q from source v_{s} to sink v_{t} with a minimum total cost.

Given a WLSCP-UL instance with n points ($P = \{1, 2, \ldots, n\}$), we construct a directed graph for the MCFP with $n + 4$ vertices:

$$V = P \cup \{v_{\mathrm{even}}, v_{\mathrm{odd}}, v_{\mathrm{s}}, v_{\mathrm{t}}\}.$$

We divide the vertices $1, 2, \ldots, n$ into the following two subsets:

$$V_{\mathrm{even}} = \{i \mid i \in P,\ a_i + b_i \text{ is even}\},$$
$$V_{\mathrm{odd}} = \{i \mid i \in P,\ a_i + b_i \text{ is odd}\}.$$

The edge set E are set as follows:

$$E = E_{\mathrm{s}} \cup E_{\mathrm{even}} \cup E_{\mathrm{odd}} \cup E_{\mathrm{match}} \cup E_{\mathrm{t}} \cup \{(v_{\mathrm{s}}, v_{\mathrm{even}}), (v_{\mathrm{even}}, v_{\mathrm{odd}}), (v_{\mathrm{odd}}, v_{\mathrm{t}})\},$$

where

$$
\begin{aligned}
&E_{\mathrm{s}} = \{(v_{\mathrm{s}}, j) \mid j \in V_{\mathrm{even}}\},\\
&E_{\mathrm{even}} = \{(v_{\mathrm{even}}, j) \mid j \in V_{\mathrm{odd}}\},\\
&E_{\mathrm{odd}} = \{(i, v_{\mathrm{odd}}) \mid i \in V_{\mathrm{even}}\},\\
&E_{\mathrm{match}} = \{(i, j) \mid i \in V_{\mathrm{even}}, j \in V_{\mathrm{odd}}, |a_i - a_j| + |b_i - b_j| = 1\},\\
&E_{\mathrm{t}} = \{(i, v_{\mathrm{t}}) \mid i \in V_{\mathrm{odd}}\}.
\end{aligned}
$$

Figure 7 shows an example of the graph construction.

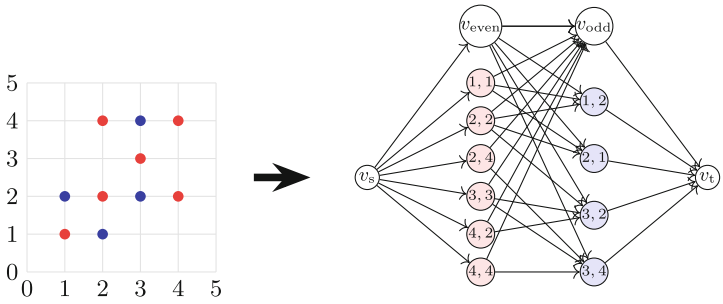


Fig. 7. An example of the reduction from WLSCP-UL to MCFP.

For each edge (i, j), c'_{ij}, u_{ij} and l_{ij} are set as

$$
c'_{ij} = \begin{cases} \min\{c_1, c_2\} & \text{if } (i,j) \in E_{\mathrm{even}} \cup E_{\mathrm{odd}} \\ c_1 & \text{if } (i,j) \in E_{\mathrm{match}} \text{ and } |a_i - a_j| = 0 \\ c_2 & \text{if } (i,j) \in E_{\mathrm{match}} \text{ and } |b_i - b_j| = 0 \\ 0 & \text{otherwise;} \end{cases}
$$

$$
u_{ij} = \begin{cases} n & \text{if } (i,j) \in \{(v_{\mathrm{s}}, v_{\mathrm{even}}), (v_{\mathrm{even}}, v_{\mathrm{odd}}), (v_{\mathrm{odd}}, v_{\mathrm{t}})\} \\ 1 & \text{otherwise;} \end{cases}
$$

$$
l_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E_{\mathrm{s}} \cup E_{\mathrm{t}} \\ 0 & \text{otherwise.} \end{cases}
$$

The flow amount q sent from v_{s} to v_{t} is set to $q = n$.

The rationale for this transformation is that the non-zero edge flows in E_{match} represent line segments covering two points, while the non-zero edge flows in $E_{\mathrm{even}} \cup E_{\mathrm{odd}}$ correspond to line segments used to cover only one point. The setting of l_{ij} ensures that all the nodes are covered. A flow from v_{s} to v_{t} corresponds to a covering of the WLSCP-UL with the same cost. Thus, using the above transformation, the optimal value of MCFP coincides with the optimal value of WLSCP-UL.

There are various algorithms [3] for the MCFP with polynomial time. Hence, the problem WLSCP-UL can also be solved in polynomial time.

Theorem 2. *The WLSCP-UL is solvable in polynomial time.*

For the WLSCP with one type of vertical line segment and one type of horizontal line segment, we showed that the problem is NP-hard if both line segments have length of 3 units, and polynomial-time solvable if both line segments have length of 1 unit. Whether the problem can be solved in polynomial time when involving line segments with length 2 is an intriguing question that remains unanswered in this study.

5 Conclusion

In this paper, we studied the weighted rectangular covering problem (WRCP), which requires to cover n points on a 2-dimensional plane by t types of weighted rectangles. For the WRCP, there exists an exact algorithm [5] with time complexity of $O(n^6 2^n)$. We proposed a new exact algorithm based on dynamic programming with a better running time of $O(\min\{n^5 2^n, tn^3 2^n\})$. We then considered a special case of the WRCP, the weighted line-segment covering problem (WLSCP), in which rectangles are given as horizontal or vertical line segments. For the WLSCP, we proved that it is NP-hard in the strong sense, even if there are only one horizontal type and one vertical type of line segments with length of 3 units. To solve the WLSCP, we proposed an exact algorithm with time complexity of $O(tn2^n)$. For the WLSCP with unit-length line segments, we showed that the problem can be solved in polynomial time by reducing it to the minimum cost flow problem.

Future research could be devoted to determining whether the WLSCP is NP-hard when line segments with length 2 units are involved. Designing approximate and heuristic algorithms for the WRCP and WLSCP would also be a good direction.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Hoboken (1993)
2. Fowler, R.J., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete. Inf. Process. Lett. **12**(3), 133–137 (1981)
3. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987)
4. Kravchenko, S.A., Sotskov, Y.N.: Optimal makespan schedule for three jobs on two machines. Math. Methods Oper. Res. **43**, 233–238 (1996)
5. Porschen, S.: On rectangular covering problems. Int. J. Comput. Geom. Appl. **19**(04), 325–340 (2009)