Escaping a Grid by Edge-Disjoint Paths*

Wun-Tat Chan† Francis Y.L. Chin† Hing-Fung Ting†

Abstract

We study the *edge-disjoint escape problem* in grids: Given a set of n sources in a two-dimensional grid, the problem is to connect all sources to the grid boundary using a set of n edge-disjoint paths. Different from the conventional approach that reduces the problem to network flow problem, we solve the problem by ensuring that no rectangles in the grid contain more sources than outlets, a necessary and sufficient condition for the existence of a solution. Based on this condition, we give a greedy algorithm which finds the paths in $O(n^2)$ time, which is faster than the previous approaches. This problem has applications in point-to-point delivery, VLSI reconfiguration and package routing.

1 Introduction

We study the *edge-disjoint escape problem* in grids, which is defined as follows. The problem input is a two-dimensional grid G and a set S of n sources in G. Note that there may be more than one sources in a vertex of G. The objective is to construct a set of n edge-disjoint paths (called a solution) such that there is a path connecting each source to the boundary of G. See Figure 1 for an example. We say that a problem is *escapable* if there is a solution to the problem. In this paper, we present an $O(n^2)$ time algorithm which determines whether a given problem is escapable. Based on this algorithm, we design another $O(n^2)$ time algorithm which finds a solution to the problem. Throughout our discussion, we assume that G has at most n rows and at most n columns, i.e., G has $O(n^2)$ vertices and edges. We can make this assumption without loss of generality by a simple preprocessing presented in [1].

The edge-disjoint escape problem is similar to the *escape problem*, which is to determine whether or not there are n *vertex-disjoint paths* connecting all sources to the boundary (see the book by Cormen et al. [3]). Finding disjoint paths in grids has applications in many real-world problems. The *point-to-point delivery problem* [7,8] is to determine a set of disjoint "shipping" paths matching the sources to destinations on the boundary. The VLSI *reconfiguration problem* [10–12] is to find a set of disjoint "compensation" paths which connects the faulty processors in a processor array to the healthy ones on the boundary. Another relevant problem is the *fanout routing problem* in pin/ball grid array packages [13, 14], which is to fan the array pins out to the package boundary for further connection.

A straightforward approach to solve the edge-disjoint escape problem reduces the problem to the maximum flow problem in a unit capacity network. The network can be constructed as follows. Given G and S, create a supersource that connects to every source in S and a supersink that connects to every boundary vertex of G, and then assign unit capacity to every edge. A maximum flow in the network corresponds to a solution to the edge-disjoint escape problem. Using the maximum flow algorithm of Goldberg and Rao [5], we can solve the edge-disjoint escape problem in $O(\sqrt{|E|}\min(|E|,|V|^{3/2}))$ time for a network $\mathcal{N}=(V,E)$, or equivalently $O(n^3)$ time because $|E|$ and $|V|$ are $O(n^2)$ in the reduction. The time complexity can be reduced to $O(|V|^{4/3}\log|V|)$, i.e., $O(n^{8/3}\log n)$ by applying the multiple sources and multiple sinks flow algorithm of Miller and Naor [9] for planar network, together with the fast shortest-path algorithm for planar graph by Henzinger et al. [6]. In view of the sparseness of the n sources in the $O(n^2)$ grid, an $O(n^{5/2})$ time algorithm [1] was proposed that compresses the $O(n^2)$ grid to a graph with $O(n^{3/2})$ edges for finding the edge-disjoint paths. Recently, the time complexity is further improved to $O(n^{9/4})$ by an algorithm [2] applying the layered network technique in the compressed grid.

To break through the $O(n^{9/4})$ barrier, we turn to a different approach. The approach bases on a simple observation that there are no solutions if there exists an *oversaturated region*—a region in the grid containing more sources than outlets. It turns out that the absence of oversaturated regions is also sufficient for the existence of a solution. To test for the absence of oversaturated regions, we need only confine ourselves to testing rectangles in the grid. However, testing if there are any *oversaturated rectangles* can be time-consuming

*The research is partially supported by a Hong Kong RGC grant 338/065/0022.

†Department of Computer Science and Information Systems, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail:{wtchan,chin,hfting}@csis.hku.hk

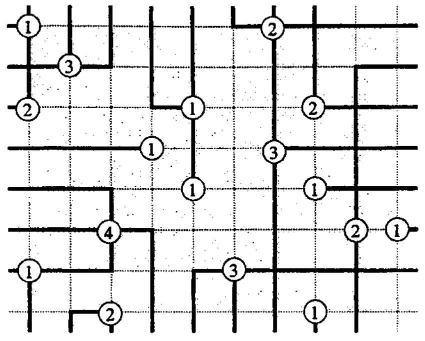
since there are $O(n^4)$ rectangles in the grid that may need to be tested. Reducing the number of rectangles to be tested is non-trivial. For example, a rectangle having a small number of sources relative to the area of the rectangle, i.e., sparse sources, could still have more sources than outlets, and thus cannot be exempted from testing. Moreover, although only the rectangles with sources on all four boundaries need to be considered, it still means $O(n^4)$ rectangles to be tested.

In next section, we give an $O(n^2)$ time algorithm to determine if a problem is escapable, by detecting if there are any oversaturated rectangles. The algorithm takes advantage of some properties in the oversaturated rectangles and applies the disjoint set union data structure to search the grid efficiently for the oversaturated rectangles. Section 3 focuses on the path-finding problem. We adopt a greedy approach to extend the edge-disjoint paths row by row. In each row, a test is carried out to ensure that given the paths built up thus far a solution still exists before moving on to the next row. If a solution no longer exists, we made a mistake in the extension of the paths in this row and should try again until we find the appropriate extension. The key to being able to solve the problem in $O(n^2)$ time is performing the test step in $O(n)$ time and limiting the number of extension attempts to $O(n)$ in total. Section 4 gives an efficient implementation of the approach and bounds the running time to $O(n^2)$.

2 Determining whether a problem is escapable

Denote an instance of the edge-disjoint escape problem by a pair (G, S) where G is an $H \times W$ grid and S is a set of n sources in G with $n \geqslant \max(H, W)$. Since there may be more than one sources in a vertex of G, S (and any set of sources) can be represented by a multi-set of vertices. The solution to (G, S) is a set of n edge-disjoint paths where each of which starts from a source in S and escapes G through a boundary vertex. The maximum number of paths that can escape through a boundary vertex v depends on the number of *outlets* connecting to v. An outlet of G is an edge connecting a (boundary) vertex in G to a vertex outside G (assuming G is a subgraph of the infinite grid). In general, as shown in Figure 1, at most two of the sources can escape G through each corner vertex[1] and one through other boundary vertices. In particular when $H = 1$ or $W = 1$, at most three of the sources can escape G through each corner vertex and two through other boundary vertices. We will discuss this special case in detail in Section 3.

[1]We can handle the case where at most one source can escape through each of the four corners by adding one additional source to each corner vertex.



ⓚ denotes the vertex containing k sources

Figure 1: An example showing 29 sources that escape from an 8×10 grid.

In this section we present an $O(n^2)$ time algorithm which determines whether (G, S) is escapable. First, we prove that the absence of oversaturated rectangles in G is necessary and sufficient for (G, S) to be escapable. Based on the distribution of the sources in G, we define H sequences and transform the search of oversaturated rectangles into finding n "maximum intervals" in these sequences. Finally, we give a novel technique for finding the maximum intervals.

2.1 Necessary and sufficient condition. For any two integers i and j with $1 \leqslant i \leqslant H$ and $1 \leqslant j \leqslant W$, let $[i, j]$ denote the vertex of G in row i (from the top) and column j (from the left). Given four integers t, b, ℓ and r with $1 \leqslant t \leqslant b \leqslant H$ and $1 \leqslant \ell \leqslant r \leqslant W$, a *rectangle* $[t, b; \ell, r]$ denotes the subgrid bounded by row t and b, and column ℓ and r, i.e., the subgraph induced by the set of vertices $\{[i, j] \mid t \leqslant i \leqslant b$ and $\ell \leqslant j \leqslant r\}$. For instance, $G = [1, H; 1, W]$. Let $\mathcal{O}(R)$ be the set of outlets of the rectangle R. We say that R is *oversaturated* (with respect to S) if the number of sources inside R is more than $|\mathcal{O}(R)|$, and R is *saturated* if the number of sources inside R equals $|\mathcal{O}(R)|$. Note that if there exists an oversaturated rectangle, then (G, S) is not escapable. The following lemma asserts that the reverse is also true.

LEMMA 2.1. *If (G, S) is not escapable, there is an oversaturated rectangle with respect to* S *in* G.

Proof. Let $G = (V, E)$. We create two vertices s and t, and define a flow network $\mathcal{N} = (V \cup \{s, t\}, E \cup E_s \cup E_t)$ where $E_s = \{(s, v) \mid v \in \mathsf{S}\}$ and $E_t = \{(u, t) \mid u \in V$ and (u, w) is an outlet of G for some vertices $w\}$. (Note that both E_s and E_t may be the multi-sets of edges.) Every edge in $\mathcal{N}$ has unit capacity.

As $|S| = n$, (G, S) is escapable if and only if the value of the maximum flow from s to t in $\mathcal{N}$ is n. Given (G, S) is not escapable, the value of the maximum flow, and hence the capacity of the minimum cut, in $\mathcal{N}$ is smaller than n. Consider the minimum cut MC of $\mathcal{N}$ with most edges in E_s. If we remove MC from $\mathcal{N}$, G is decomposed into a number of connected components. One of these components must still connect to s (through some edges in E_s) because $|E_s| = n > |MC|$. Let D be this component, $\mathcal{O}(D)$ be the set of outlets of D, and S_D be the set of sources in D. We must have $|\mathcal{O}(D)| < |S_D|$; otherwise, either $(MC - \mathcal{O}(D)) \cup \{(s,x) \mid x \in S_D\}$ is a cut with less edges than MC for the case $|\mathcal{O}(D)| > |S_D|$, or $(MC - \mathcal{O}(D)) \cup \{(s,x) \mid x \in S_D\}$ is another minimum cut containing more edges in E_s than MC for the case $|\mathcal{O}(D)| = |S_D|$. Let R be the smallest rectangle containing D, and S_R be the set of sources in R. Since $|\mathcal{O}(R)| \leqslant |\mathcal{O}(D)| < |S_D| \leqslant |S_R|$, R is oversaturated. □

As a result, determining whether (G, S) is escapable can be reduced to finding whether there are any oversaturated rectangles. In next two subsections we give an $O(n)$ time procedure to determine for a fixed integer t, whether there are any oversaturated rectangles whose top boundary is on row t. By repeating this procedure for all t with $1 \leqslant t \leqslant H$, we can determine whether (G, S) is escapable in $O(n^2)$ time. In our procedure we will ignore those rectangles $[t, b; \ell, r]$ which contain no sources in their bottom row b because even if they are oversaturated, we must have found some smaller oversaturated rectangles $[t, b'; \ell, r]$ with $b' < b$ and their bottom rows containing sources. This implies that we only need to find for each vertex $[b, h]$ containing sources whether there are any oversaturated rectangles $[t, b; \ell, r]$ whose bottom rows include $[b, h]$ (i.e., $\ell \leqslant h \leqslant r$).

2.2 Oversaturated rectangles and maximum intervals. Given two fixed integers t and b, let N_h be the number of sources in column h between row t and b inclusively (i.e., the number of sources in $[t, b; h, h]$). Note that $[t, b; \ell, r]$ has $N_\ell + N_{\ell+1} + \cdots + N_r$ sources and $2(b-t+1)+2(r-\ell+1)$ outlets. Thus, $[t, b; \ell, r]$ is oversaturated if and only if $(N_\ell-2)+(N_{\ell+1}-2)+\cdots+(N_r-2)$ is greater than $2(b-t+1)$, which is a constant with respect to t and b. Let $Q_b = (N_1-2, N_2-2, \ldots, N_W-2)$. Define the ***maximum interval of*** Q_b ***containing the h-th element*** to be the interval $(N_i-2, N_{i+1}-2, \ldots, N_j-2)$ where $i \leqslant h \leqslant j$ and $(N_i-2)+(N_{i+1}-2)+\cdots+(N_j-2)$ attains the maximum value.

FACT 2.1. *If $(N_i-2, N_{i+1}-2, \ldots, N_j-2)$ is an interval with its sum greater than $2(b-t+1)$, then $[t, b; i, j]$ is an oversaturated rectangle.*

The following lemma gives a condition for the existence of an oversaturated rectangle $[t, b; \ell, r]$ containing $[b, h]$.

LEMMA 2.2. *Given Q_b and $[b, h]$, there is an oversaturated rectangle with top boundary on row t and bottom boundary on row b and containing $[b, h]$ if and only if the sum of the maximum interval of Q_b containing the h-th element is greater than $2(b-t+1)$.*

In our procedure, we would not compute the sum of an interval directly. Instead, we would reduce it to finding the number of sources in a corresponding rectangle because after a preprocessing it can be computed very efficiently, i.e., in constant time. Let $\mathsf{s}(t, b; \ell, r)$ denote the number of sources in a rectangle $[t, b; \ell, r]$. Note that $\mathsf{s}(t, b; \ell, r) = \mathsf{s}(1, b; 1, r) - \mathsf{s}(1, t-1; 1, r) - \mathsf{s}(1, b; 1, \ell-1) + \mathsf{s}(1, t-1; 1, \ell-1)$. If we precompute all the values $\mathsf{s}(1, b; 1, r)$ for $1 \leqslant b \leqslant H$ and $1 \leqslant r \leqslant W$ and store them in a table, we can compute $\mathsf{s}(t, b; \ell, r)$ in constant time. By the definition of Q_b, the sum of an interval $(N_i-2, N_{i+1}-2, \ldots, N_j-2)$ equal to $\mathsf{s}(t, b; i, j) - 2(j-i+1)$, which can also be computed in constant time.

2.3 Finding maximum intervals. For notational simplicity, let $Q_b = (q_1, q_2, \ldots, q_W)$, i.e., $q_i = N_i - 2$. For any integer i with $1 \leqslant i \leqslant W$, let $\mathcal{R}(i)$ denote the smallest integer $r \geqslant i$ such that $q_i + q_{i+1} + \cdots + q_r$ attains the maximum value. Similarly, $\mathcal{L}(i)$ denotes the largest integer $\ell \leqslant i$ such that $q_\ell + q_{\ell+1} + \cdots + q_i$ attains the maximum value. See Figure 2 for an example. Given the values $\mathcal{R}(i)$ and $\mathcal{L}(i)$ for $1 \leqslant i \leqslant W$, we can identify for any integer h the maximum interval of Q_b containing q_h as follows:

LEMMA 2.3. *The maximum interval of Q_b containing q_h is $(q_{\mathcal{L}(h)}, \ldots, q_{\mathcal{R}(h)})$.*

Proof. $(q_i, \ldots, q_j)$ is the maximum interval of Q_b containing q_h if and only if the maximum interval of Q_b beginning from and ending at q_h are $(q_h, \ldots, q_j)$ and $(q_i, \ldots, q_h)$ respectively. □

Before we show how to find the values $\mathcal{R}(i)$ and $\mathcal{L}(i)$ for Q_b, we give some properties on the values $\mathcal{R}(i)$. The values $\mathcal{L}(i)$ also have the similar properties.

FACT 2.2.
$$\left\{ \begin{array}{ll} a. & \mathcal{R}(j) = \mathcal{R}(i) \textit{ for } i \leqslant j \leqslant \mathcal{R}(i); \\ b. & \mathcal{R}(j) \geqslant \mathcal{R}(i) \textit{ for } j \geqslant i; \\ c. & \mathcal{R}(i) = \mathcal{R}(i+1) \textit{ if } (i \neq W \textit{ and } \\ & \quad \sum_{i+1 \leqslant j \leqslant \mathcal{R}(i+1)} q_j > 0), \\ & \quad \textit{otherwise}, \mathcal{R}(i) = i. \end{array} \right\}$$

Since the sequence of values $\mathcal{R}(i)$ for $1 \leqslant i \leqslant W$ is non-decreasing, i.e., there may be consecutive repeating

$\mathcal{R}(i)$:	2	2	7	7	7	7	7	10	10	10
q_i :	2	2	−7	1	−9	5	6	−4	−1	2
$\mathcal{L}(i)$:	1	1	1	4	4	6	6	6	6	6

Figure 2: A sequence with its right-partition equal to (2, 7, 10) and its left-partition equal (1, 4, 6).

values, we adopt a simple representation for the values $\mathcal{R}(i)$, the *right-partition*, that is the sequence of distinct values among the values $\mathcal{R}(i)$. Let $\mathcal{R} = (r_1, r_2 \dots, r_d)$ be the right-partition where $r_j < r_{j+1}$ for $1 \leqslant j \leqslant d-1$. Given any integer i with $1 \leqslant i \leqslant W$, we have

$$\mathcal{R}(i) = r_j \quad \text{for } r_{j-1} < i \leqslant r_j \text{ and } 1 \leqslant j \leqslant d$$

assuming $r_0 = 0$. Similarly, the *left-partition* is the sequence of distinct values among the values $\mathcal{L}(i)$.

Instead of computing each right-partition (and left-partition) of Q_b individually for each integer b with $t \leqslant b \leqslant H$, we give a method to efficiently update the right-partition of Q_b to the right-partition of Q_{b+1}. The left-partitions can be updated similarly. Let $Q_{b+1} = (q'_1, q'_2, \dots, q'_W)$. Suppose there are k vertices which contain sources in row $(b+1)$. Denote $c_1 < c_2 < \dots < c_k$ the columns of these vertices, then $q'_{c_\alpha} > q_{c_\alpha}$ for $1 \leqslant \alpha \leqslant k$ and $q'_i = q_i$ elsewhere. We call $P_0, P_1, \dots, P_k$ a *refinement* from Q_b to Q_{b+1} where $P_0 = Q_b$, $P_k = Q_{b+1}$, and $P_{\alpha+1}$ contains the same sequence of elements as P_α except that the $c_{\alpha+1}$-th element of $P_{\alpha+1}$ is $q'_{c_{\alpha+1}}$ (instead of $q_{c_{\alpha+1}}$). For instance, $P_{\alpha+1} = (q'_1, \dots, q'_{c_{\alpha+1}}, q_{c_{\alpha+1}+1}, \dots, q_W)$. The update from the right-partition of Q_b to the right-partition of Q_{b+1} can be refined to k updates, which are from the right-partition of P_0 to that of P_1, the right-partition of P_1 to that of P_2, and so on.

Now we show how to update the right-partition $\mathcal{R}$ of P_α to the right-partition $\mathcal{R}'$ of $P_{\alpha+1}$, for $0 \leqslant \alpha \leqslant k-1$. Let $P_\alpha = (p_1, p_2, \dots, p_W)$, $P_{\alpha+1} = (p'_1, p'_2, \dots, p'_W)$, and $\mathcal{R} = (r_1, r_2, \dots, r_d)$. Recall that $p'_i = p_i$ for $1 \leqslant i \leqslant W$ except for $i = c_{\alpha+1}$ where $p'_{c_{\alpha+1}} > p_{c_{\alpha+1}}$. Let $r_y = \mathcal{R}(c_{\alpha+1})$, and r_x with $x < y$ be the largest integer in $\mathcal{R}$ such that $\sum_{r_x+1 \leqslant j \leqslant r_y} p'_j \leqslant 0$. So, $\sum_{r_i+1 \leqslant j \leqslant r_y} p'_j > 0$ for $x < i \leqslant y$. In the following lemma, we further prove that $\sum_{i \leqslant j \leqslant r_y} p'_j > 0$ for $r_x + 1 < i \leqslant r_y$. Therefore, $\mathcal{R}(i)$ for $r_x + 1 \leqslant i \leqslant r_y$ are all the same (by Fact 2.2c). Thus, we can update $\mathcal{R}$ to $\mathcal{R}'$ using r_y and r_x as follows.

LEMMA 2.4. *Let $r_y = \mathcal{R}(c_{\alpha+1})$ and r_x with $x < y$ be the largest integer in $\mathcal{R}$ such that $\sum_{r_x+1 \leqslant j \leqslant r_y} p'_j \leqslant 0$. If r_x exists, the right-partition $\mathcal{R}'$ of $P_{\alpha+1}$ is $(r_1, \dots, r_x, r_y, \dots, r_d)$; otherwise, $\mathcal{R}'$ is $(r_y, \dots, r_d)$.*

Proof. Since $p'_i = p_i$ for $r_y + 1 \leqslant i \leqslant W$, we have $\mathcal{R}'(i) = \mathcal{R}(i)$ for $r_y \leqslant i \leqslant W$. The values $r_y, r_{y+1} \dots, r_d$ are retained in $\mathcal{R}'$. In contrast, the values $r_{x+1}, r_{x+2}, \dots, r_{y-1}$ are removed in $\mathcal{R}'$. It is because $\mathcal{R}'(i) = r_y$ for $r_x + 1 \leqslant i \leqslant r_y$ which can be proved as follows. Since $\mathcal{R}(r_x) = r_x$, $\mathcal{R}(r_x + 1) = r_{x+1}$ (by the definition of right-partition), and hence $\sum_{r_x+2 \leqslant i \leqslant r_{x+1}} p'_i \geqslant \sum_{r_x+2 \leqslant i \leqslant r_{x+1}} p_i > 0$. Also, by the definition of x, we have $\sum_{r_{x+1}+1 \leqslant i \leqslant r_y} p'_i > 0$. Thus summing up the two terms, we have $\sum_{r_x+2 \leqslant i \leqslant r_y} p'_i > 0$. Also, as $\mathcal{R}'(r_y) = r_y$, $\mathcal{R}'(r_x + 1) = r_y$ (by Fact 2.2b). Therefore, we have $\mathcal{R}'(i) = r_y$ for $r_x + 1 \leqslant i \leqslant r_y$ (by Fact 2.2a). If r_x exists, the values $r_1, r_2, \dots, r_x$ are also retained in $\mathcal{R}'$. Since $\sum_{r_x+1 \leqslant i \leqslant \mathcal{R}'(r_x+1)} p'_i < 0$, $\mathcal{R}'(r_x) = r_x$ (by Fact 2.2c). For $1 \leqslant i \leqslant r_x$, since $p'_i = p_i$ and $\mathcal{R}'(r_x) = r_x$, we have $\mathcal{R}'(i) = \mathcal{R}(i)$, and thus the values $r_1, r_2, \dots, r_x$ are retained. □

Remark: Recall that we do not compute the sums of intervals directly but reduce it to finding the numbers of sources in the corresponding rectangles. For that reason, we may not refine an update from the right-partition of Q_b to that of Q_{b+1} to exactly k updates in the refinement, but less than k updates. For example, if $c_1, c_2, \dots, c_i$ are all less than or equal to $\mathcal{R}(c_1)$, we update the right-partition of P_0 directly to that of P_i because we can only compute the sum of $(q'_j, q'_{j+1}, \dots, q'_{\mathcal{R}(c_1)})$ in Q_{b+1} where $\mathcal{R}(j-1) = j-1$ and $\mathcal{R}(j) = \mathcal{R}(c_1)$, by finding the number of sources in $[t, b+1; j, \mathcal{R}(c_1)]$.

Our procedure starts from a sequence $Q_{t-1} = (-2, -2, \dots, -2)$ which represents that there is no source in each column. Its right-partition is $(1, 2, \dots, W)$. From the right-partition of Q_{t-1}, we can construct the right-partition of Q_t through the updates in the refinement defined above. Consequently, our procedure computes the right-partition of Q_b for $t \leqslant b \leqslant H$.

Now, we show that our problem of computing the right-partitions of Q_b for $t \leqslant b \leqslant H$ can be transformed to a special case of the *disjoint set union problem* proposed by Gabow and Tarjan [4]. This special case also support the FIND and UNION operations as in the general case, but the UNION operation here can only combine two sets with "special relation". For example, when the sets can be arranged in a line, the UNION operation only combine two adjacent sets. The transformation can be describe as follows. Let all the integers i for $r_{j-1} + 1 \leqslant i \leqslant r_j$ be represented by a *set* denoted by r_j. The operation of determining the value $\mathcal{R}(i)$ is transformed to finding the set representing i. An update of the right-partition in the refinement is transformed to a series of "adjacent" sets unions, i.e., the unions of sets $r_y, r_{y-1}, \dots,$ and r_{x+1}. Since one set is deleted in one UNION operation, the total number of UNION operations is less than W. The total number of FIND operations is no more than n because for each vertex containing the sources we only need to locate the set that contains the vertex once. Since this disjoint set union problem can be solved in time linear to the number of FIND and UNION operations [4], our problem of computing right-partitions and left-partitions can be solved in $O(W+n)$, or $O(n)$ time.

LEMMA 2.5. *Procedure* OVERSAT(R, S_R) *(in Algorithm 1) runs in* $O(|S_R|)$ *time.*

The procedure which finds if there are any oversaturated rectangles $[t,b;\ell,r]$ in a given rectangle R with a set of sources S_R is shown in Algorithm 1. This procedure can be called, for each $R=[t,H;1,W]$ with $1 \leqslant t \leqslant H$, to find if there are any oversaturated rectangles in G. Hence, we can determine whether (G, S) is escapable in $O(n^2)$ time.

```
Input: $R$ is a rectangle $[t,H;1,W]$ and $S_R$ is a set of sources in $R$;
Output: A set of oversaturated rectangles $[t,b;\ell,r]$ for $t \leqslant b \leqslant H$ and $1 \leqslant \ell \leqslant r \leqslant W$;
/*$Q_i = (q_1, q_2, \ldots, q_W)$ where $q_j + 2$ is the number of sources of $S_R$ in $[1,i;j,j]$.*/
Let $Q_{t-1} = (-2,-2,\ldots,-2)$ and both the right- and left-partitions of $Q_{t-1}$ be $(1,2,\ldots,W)$;
for $b \leftarrow t$ to $H$ do
    Update the right- and left-partitions of $Q_{b-1}$ to those of $Q_b$;
    foreach vertex $[b,h]$ containing sources do
        Output $[t,b;\mathcal{L}(h),\mathcal{R}(h)]$ if it is oversaturated;
```

Algorithm 1: OVERSAT(R, S_R)

3 Finding the paths

Assume (G,S) is escapable. In this section we present an algorithm which finds a solution to (G,S). Let F denote the set of sources in row 1 and $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_\beta$ denote the sequence of columns where the sources in F are located, i.e., $\mathsf{F} = \{[1,s_i] \mid 1 \leqslant i \leqslant \beta\}$. Roughly speaking, our algorithm solves the problem by solving its two subproblems, $([1,1;1,W],\mathsf{F})$ and $([2,H;1,W],\mathsf{S}-\mathsf{F})$. Obviously, the solutions to these two subproblems affect each other. In the following, we give the condition on the solutions to the two subproblems such that they do not conflict with each other. Then we show how to combine the two solutions to form a solution to (G,S).

3.1 Dividing the problem.

We represent a solution to $([1,1;1,W],\mathsf{F})$ by a *mapping* which stores the columns where the sources in F escape $[1,1;1,W]$. For a mapping $\sigma = (\sigma(1),\sigma(2),\ldots,\sigma(\beta))$, the sources in $[1,s_i]$ escapes $[1,1;1,W]$ through $[1,\sigma(i)]$. Moreover, we assume $\sigma(i) \leqslant \sigma(i+1)$ for $1 \leqslant i \leqslant \beta-1$. Since there are three outlets connecting to each of $[1,1]$ and $[1,W]$, and two outlets to other vertices, σ satisfies the following conditions:

$$(3.1)\qquad \left\{\begin{array}{l} \text{No more than three } \sigma(i)\text{'s have the} \\ \quad \text{same value } 1; \\ \text{no more than two } \sigma(i)\text{'s have the} \\ \quad \text{same value } j, \text{ for } 2 \leqslant j \leqslant W-1; \\ \text{no more than three } \sigma(i)\text{'s have the} \\ \quad \text{same value } W. \end{array}\right\}$$

Note that if the path of the first source in F escapes through $[1,1]$ horizontally, and similarly if the path of the last source in F escapes through $[1,W]$ horizontally, both paths will not conflict with any solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$. Thus, without loss of generality, we assume for all mappings σ,

$$\sigma(1) = 1 \quad \text{and} \quad \sigma(\beta) = W.$$

Moreover, since the paths must be edge-disjoint, we can assume that the path starting at $[1,s_i]$ do not escape on the left of $[1,s_{i-1}]$ nor on the right of $[1,s_{i+1}]$. Hence, for $2 \leqslant i \leqslant \beta-1$ we have

$$(3.2)\qquad s_{i-1} \leqslant \sigma(i) \leqslant s_{i+1}.$$

On the other hand, we can determine a solution to $([1,1;1,W],\mathsf{F})$ by a given mapping. Although the mapping does not specify whether a path starting from the source in $[1,s_i]$ escapes through $[1,\sigma(i)]$ upwards or downwards, we assume the path always goes upwards whenever possible, i.e., when $i=2$ or $\sigma(i) \neq \sigma(i-1)$. In the rest of the paper, finding a solution to $([1,1;1,W],\mathsf{F})$ is always referred to finding a mapping (satisfying the above conditions). Figure 3 shows an example on a solution to $([1,1;1,W],\mathsf{F})$ and the corresponding mapping.

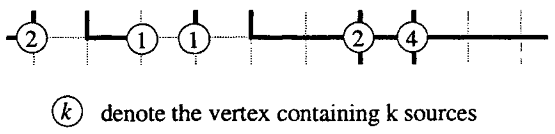


Figure 3: A solution to $([1,1;1,10],\mathsf{F})$. The corresponding mapping is $(1,1,2,4,5,7,7,8,8,10)$.

Given any mapping σ to $([1,1;1,W],\mathsf{F})$, we present a condition that guarantees the existence of a conflict-free solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$, i.e., the solution to $([2,H;1,W], \mathsf{S}-\mathsf{F})$ that can be combined with the solution to $([1,1;1,W],\mathsf{F})$ to form a solution to (G,S). In fact, the condition depends on whether the problem instance $([2,H;1,W], \mathsf{S}_\sigma \cup (\mathsf{S}-\mathsf{F}))$ is escapable where

$$\mathsf{S}_\sigma = \{[2,\sigma(i)] \mid 2 \leqslant i \leqslant \beta - 1\}.$$

Obviously, a solution to $([2,H;1,W], \mathsf{S}_\sigma \cup (\mathsf{S}-\mathsf{F}))$ contains a solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$. The set of extra sources S_σ, called *induced sources*, ensures that such solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$ are compatible with the solution to $([1,1;1,W],\mathsf{F})$ induced by σ.

Denote $\mathsf{T}_\sigma = \mathsf{S}_\sigma \cup (\mathsf{S}-\mathsf{F})$. If $([2,H;1,W],\mathsf{T}_\sigma)$ is escapable, we can combine the solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$ with the solution to $([1,1;1,W],\mathsf{F})$ induced by σ to form the solution to (G,S).

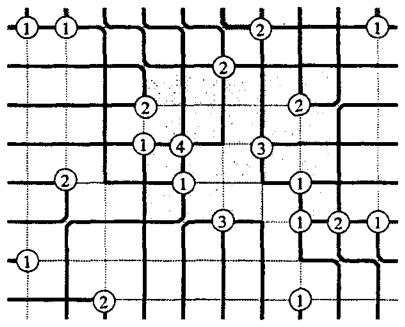
LEMMA 3.1. *Given a mapping σ and a solution to $([2,H;1,W],\mathsf{T}_\sigma)$, we can construct a solution to (G,S) in $O(W)$ time.*

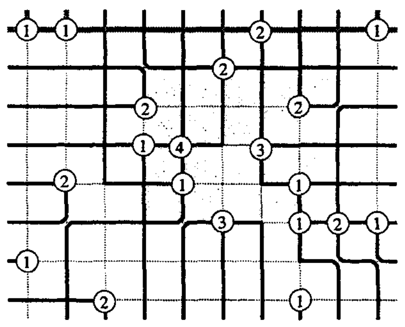
Proof. The construction is done by extending the paths in both the solution to $([1,1;1,W],\mathsf{F})$ induced by σ and the solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$ included in the solution to $([2,H;1,W],\mathsf{T}_\sigma)$. We perform two types of path extension: to extend the paths which escape downwards through $[1,j]$ towards $[2,j]$ in the solution to $([1,1;1,W],\mathsf{F})$, and to extend the paths which escape upwards through $[2,k]$ towards $[1,k]$ in the solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$. First, if a path escapes through $[1,j]$ towards $[2,j]$ in the solution to $([1,1;1,H],\mathsf{F})$, we have $[2,j] \in \mathsf{S}_\sigma$, and we need to extend this path from $[1,j]$ to $[2,j]$ and then to the boundary of G. In fact, the extension from $[2,j]$ is already done by the path starting from the induced source in $[2,j]$ in the solution to $([2,H;1,W],\mathsf{T}_\sigma)$. Second, in the solution to $([2,H;1,W],\mathsf{S}-\mathsf{F})$ if a path escapes through $[2,k]$ towards $[1,k]$, we need to extend the path to $[1,k]$. The extension is trivial when there are no other paths escaping through $[1,k]$ toward $[0,k]$ (assuming $[0,k]$ is in the infinite grid) in the solution to $([1,1;1,W],\mathsf{F})$. Otherwise, there is an induced source in $[2,k]$. In that case, we assume the path which escapes through $[2,k]$ towards $[1,k]$ indeed starts from the induced source and we can discard the path. □

The above lemma prove that the existence of an escapable $([2,H;1,W], \mathsf{T}_\sigma)$ is a sufficient condition for (G,S) to be escapable. In the following, Lemma 3.2 proves that the condition is also necessary for (G,S) to be escapable.

LEMMA 3.2. *If (G,S) is escapable, then there exists a mapping σ for F such that $([2,H;\ 1,W],\mathsf{T}_\sigma)$ is escapable.*

Proof. In a solution to (G,S), consider the paths starting from the sources in F. By the columns from which the paths escape $[1,1;1,W]$, we obtain a sequence of integers which form a mapping σ. However, σ thus formed does not guarantee an escapable $([2,H;1,W],\mathsf{T}_\sigma)$. This is because the paths are not unique due to their edge-disjointness. (See Figure 4 for an example.) In order to find the σ with an escapable $([2,H;1,W],\mathsf{T}_\sigma)$, we have to ensure that if a path p starting from a source in F escapes through $[1,j]$ toward $[0,j]$, the edge $([2,j],[1,j])$


(a)


(b)

⬚ A saturated rectangle

ⓚ denotes the vertex containing k sources

Figure 4: The sources escape from an 8×10 grid. (a) The paths starting from the sources on row 1 are: $([1,1]),([1,2]),([1,7] \to [1,6]),([1,7]),([1,10])$. Thus we obtain a mapping $\sigma = (1,2,6,7,10)$. Since the rectangle $[2,6;4,9]$ is saturated with respect to S, it is oversaturated with respect to T_σ, and hence the problem instance $([2,8;1,10],\mathsf{T}_\sigma)$ is not escapable. (b) The paths starting from the sources on row 1 are: $([1,1]),([1,2] \to [1,1]),([1,7] \to [1,6] \to \ldots \to [1,2]),([1,7] \to [1,8] \to \ldots \to [1,10]),([1,10])$. Thus we obtain a mapping $\sigma = (1,1,2,10,10)$ and an escapable problem instance $([2,8;1,10],\mathsf{T}_\sigma)$.

is not used by any path. (If a path starting from a source in F entered row 2 through column j, the edge $([2,1],[1,j])$ surely will not be used by any path.) This can always be done because if there is a path q using the edge $([2,j],[1,j])$, the paths p and q can be switched at $[1,j]$, i.e., path p will follow the remaining part of path q to escape $[1,1;1,W]$ at some other column. If necessary, the path can be switched more than once until the path escapes $[1,1;1,W]$ at a column k where edge $([2,k],[1,k])$ is not used by any path.

We show that $([2,H;1,W],\mathsf{T}_\sigma)$ is escapable as follows. In the solution to (G,S), we already have the paths starting from the sources in $\mathsf{S}-\mathsf{F}$ to the boundary of $[2,H;1,W]$. The main point is how we can construct the paths from the induced sources to the boundary of $[2,H;1,W]$. Consider an induced source in $[2,\sigma(i)]$, if it is due to a path p' starting from a source in F entering row 2 through $[2,\sigma(i)]$, the path starting from this induced source follows the remaining part of path p' at $[2,\sigma(i)]$ to the boundary of $[2,H;1,W]$. If the induced source in $[2,\sigma(i)]$ is due to a path starting from a source in F escapes through $[1,\sigma(i)]$ towards $[0,\sigma(i)]$, as there are no paths in the solution to (G,S) using the edge $([2,\sigma(i)],[1,\sigma(i)])$ (by the property of σ), the path starting from this induced source can escape through $[2,\sigma(i)]$ towards $[1,\sigma(i)]$. □

Combining Lemmas 3.1 and 3.2, we have the following theorem.

THEOREM 3.1. *(G,S) is escapable if and only if there is a mapping σ for* F *such that $([2,H;1,W],\mathsf{T}_\sigma)$ is escapable.*

DEFINITION 3.1. *A mapping σ for* F *is* feasible *if T_σ is escapable.*

In our algorithm we will find a feasible mapping for F, in particular, the "smallest" feasible one. To check whether a mapping σ is feasible, we can test whether $([2,H;1,W],\mathsf{T}_\sigma)$ is escapable. Since $([2,H;1,W],\mathsf{S}-\mathsf{F})$ is escapable and the induced sources are added to row 2 only, $([2,H;1,W],\mathsf{T}_\sigma)$ is escapable if and only if OVERSAT$([2,H;1,W],\mathsf{T}_\sigma)$ (in Algorithm 1) outputs no oversaturated rectangle. After the mapping σ is found, we will solve $([2,H;1,W],\mathsf{T}_\sigma)$ recursively. Then, we combine the solution to $([1,1;1,W],\mathsf{F})$ and the solution to $([2,H;1,W],\mathsf{T}_\sigma)$ to form the solution to (G,S).

3.2 Finding the smallest feasible mapping.

A mapping σ is *smaller* than another mapping λ, denoted by $\sigma < \lambda$, if there exists an integer i with $1 \leqslant i \leqslant \beta-1$ such that $\sigma(j)=\lambda(j)$ for $1 \leqslant j \leqslant i-1$ and $\sigma(i) < \lambda(i)$. In addition, we say $\sigma \leqslant \lambda$ if $\sigma < \lambda$ or $\sigma(j)=\lambda(j)$ for $1 \leqslant j \leqslant \beta$. The *smallest feasible mapping (SFM)* is the smallest mapping among the feasible mappings.

Our algorithm composes a number of *testing mapping* and checks their feasibilities one by one. The first testing mapping σ is the smallest mapping. If σ is not feasible, we compose the next testing mapping σ' based on σ such that $\sigma < \sigma' \leqslant$ SFM. Such σ' can be constructed based on one of the oversaturated rectangles with respect to T_σ (as in Definition 3.2). This process (called *trial*) is repeated until a feasible mapping (the SFM) is found .

DEFINITION 3.2. *Let $R=[2,b;\ell,r]$ be an oversaturated rectangle with respect to T_σ and δ_R be the difference between the number of sources of T_σ in R and the number of outlets $|O(R)|$. In order to prevent R from oversaturated, we have to construct a new testing mapping such that the last δ_R induced sources in R are introduced to the right of column r. We define θ the* threshold *of R, for $1 \leqslant \theta \leqslant \beta-1$, to be the maximum number of induced sources that can be retained to the left of column r, without making R oversaturated. To be precise, θ is defined by the condition $\sigma(\theta+\delta_R) \leqslant r < \sigma(\theta+\delta_R+1)$.*

Hence, σ' is constructed by assigning $\sigma'(i)=\sigma(i)$ for $1 \leqslant i \leqslant \theta$, $\sigma'(\theta+1)=r+1$, and $\sigma'(\theta+1),\sigma'(\theta+2),\ldots,\sigma'(\beta-1)$ to be the smallest sequence of values satisfying Conditions (3.1) and (3.2). Lemma 3.3 shows that the SFM can be obtained in a finite number of trials.

LEMMA 3.3. *σ' is a mapping and $\sigma < \sigma' \leqslant$ SFM.*

Proof. First, we prove that σ' is a mapping. Clearly, σ' satisfies Condition (3.1). Moreover, if $s_\theta \leqslant \sigma'(\theta+1) \leqslant s_{\theta+2}$, σ' satisfy Condition (3.2). Since $s_\theta \leqslant \sigma(\theta+1) < r+1=\sigma'(\theta+1)$, we shall prove that $s_{\theta+2} \geqslant r+1$. Let $R'=[2,b';\ell',r]$ be the oversaturated rectangle with respect to T_σ such that the threshold θ' of R' is no larger than θ and ℓ' is the smallest (or leftmost) possible. Consider the leftmost induced source $[2,\sigma(k)]$ in R', i.e., $\sigma(k-1) < \ell' \leqslant \sigma(k)$. We claim that $s_{k-1} \geqslant \ell'$. Suppose on the contrary that $s_{k-1} < \ell'$. By Condition (3.2), the smallest possible value of $\sigma(k)$ is s_{k-1}, but now $\sigma(k) \geqslant \ell' > s_{k-1}$. Therefore, either there is a saturated rectangle R'' with respect to T_σ with its right boundary on column $\sigma(k)-1$, or $[2,\sigma(k)-1]$ already contains two induced sources, i.e., $\sigma(k-2)=\sigma(k-1)=\sigma(k)-1=\ell'-1$. In the former case, the smallest rectangle containing both R'' and R' is also oversaturated. In the latter case, $[2,b';\ell'-1,r]$ is also oversaturated. In both cases, we have a contradiction to R' that we obtain an oversaturated rectangle with the threshold no larger than θ and the right boundary on column r, but its left boundary on the left of column ℓ'. Thus, we prove our claim that $s_{k-1} \geqslant \ell'$. Note that if $s_{k-1} \geqslant \ell'$ and $s_{\theta+2} \leqslant r$ and $[2, b'; \ell', r]$ is oversaturated, then $[1, b'; \ell', r]$ is oversaturated with respect to S which is a contradiction. Hence, we prove that $s_{\theta+2} \geqslant r+1$ and σ' is a mapping.

Second, we show that $\sigma < \sigma' \leqslant$ SFM. Obviously, we have $\sigma < \sigma'$. Now we prove that $\sigma' \leqslant$ SFM. If SFM $< \sigma'$, there is an integer i with $1 \leqslant i \leqslant \beta - 1$ such that SFM$(j) = \sigma'(j)$ for $1 \leqslant j \leqslant i-1$ and SFM$(i) < \sigma'(i)$. If $i \leqslant \theta$, it contradicts to $\sigma <$ SFM. If $i = \theta + 1$, R is oversaturated with respect to $\mathsf{T}_{\mathrm{SFM}}$. If $\theta + 2 \leqslant i \leqslant \beta$, the SFM violates Conditions (3.1) or (3.2). □

By the brute force implementation of this approach, we may need $O(W)$ trials, $O(nW)$ time for an SFM, and hence $O(nHW)$ or $O(n^3)$ time for the SFMs in all rows. In next section, we give a more efficient implementation which improves the $O(n^3)$ bound to $O(n^2)$.

4 An efficient implementation

Recall the flow of our algorithm. Given the problem instance $([1, H; 1, W], \mathsf{S})$, we reduce it to another problem instance $([2, H; 1, W], \mathsf{T}_{\mathcal{M}})$ where $\mathcal{M}$ is the SFM for row 1, and we solve $([2, H; 1, W], \mathsf{T}_{\mathcal{M}})$ recursively. Let $([x, H; 1, W], U_x)$ for $1 \leqslant x \leqslant H$ denote all problem instances handled by our algorithm. In particular, $U_1 = \mathsf{S}$ and $U_2 = \mathsf{T}_{\mathcal{M}}$. Note that it takes a number of trials to check the feasibility of the testing mappings in each row. Given one of these trials, we say it is a *successful trial* if the testing mapping is feasible; otherwise, it is a *fail trial*. Obviously, there is only one successful trial for a row, and totally H over all rows. However, by the brute force approach in previous section, the number of fail trials over all rows is $O(n^2)$. In order to reduce the number of fail trials to $O(n)$, for every row x we compose a *proper first testing mapping* for row x in the problem instance $([x, H; 1, W], U_x)$ by considering a set of saturated rectangles (with respect to U_x) having their top boundaries on row x.

Let $\mathcal{Y}_x = \{R_i = [x, b_i; \ell_i, r_i]\}$ be a set of saturated rectangles with respect to U_x. (We will show how to obtain $\mathcal{Y}_x$ in next paragraph.) Below, we show that by a given $\mathcal{Y}_x$ how to compose the first testing mapping for row x in $([x, H; 1, W], U_x)$. Denote $\mathsf{F}' = \{[x, t_1], [x, t_2], \ldots, [x, t_\gamma]\}$ the sources of U_x in row x. We compose the first testing mapping μ by the smallest mapping for F' satisfying the following requirement. For each t_j, if $t_j \leqslant r_i < t_{j+1}$ for some R_i in $\mathcal{Y}_x$, we have $\mu(t_j) = r_i + 1$. (If there are more than one r_i satisfying the condition $t_j \leqslant r_i < t_{j+1}$, take the largest r_i.) This initialization step avoids most of the fail trials. It is because if $\mu(t_j) \leqslant r_i$, $[x+1, b_i; \ell_i, r_i]$ would be oversaturated with respect to the sources including the sources of S on or below row $x+1$ and the induced sources by μ on row $(x+1)$, hence μ is not feasible. The reason to assign $\mu(t_j) = r_i + 1$ is that since R_i is saturated, there must be a path escaping R_i through the outlet $([x, r_i], [x, r_i + 1])$ of R_i. Without loss of generality, we assume that the path starts from the rightmost source in row x of R_i, which is the source in $[x, t_j]$ satisfying the condition $t_j \leqslant r_i < t_{j+1}$. Note that similar requirement is needed for the left boundaries of R_i. However, because we are testing from the smallest mapping to the smallest feasible mapping, these requirement is satisfied automatically.

We find the rectangles in $\mathcal{Y}_x$, for $2 \leqslant x \leqslant H$, by the following methods. Note that we do not aim at finding all saturated rectangles (with respect to U_x) with top boundary on row x but we find a set $\mathcal{Y}_x$ such that we can bound the number of fail trials according the initialization of first testing mappings above.

1. Consider a fail trial on a testing mapping σ for row $(x-1)$. Let V_σ be the set of sources including the sources of S on or below row x and the induced sources by σ. As σ is not feasible, there exists an oversaturated rectangle R^* having the *smallest threshold* θ^* among the oversaturated rectangles with respect to V_σ. In the efficient implementation, we construct the next testing mapping σ' based on R^*. Since R^* is saturated with respect to $V_{\sigma'}$ and $\mathcal{M}_{x-1}(i) = \sigma'(i)$ for $1 \leqslant i \leqslant \theta^*$ where $\mathcal{M}_{x-1}$ is the SFM for row $(x-1)$, we have R^* saturated with respect to U_x. We add R^* to $\mathcal{Y}_x$.
2. Consider each saturated rectangle $[x-1, b; \ell, r]$ in $\mathcal{Y}_{x-1}$. If $b \geqslant x$, the rectangle $[x, b; \ell, r]$ is saturated with respect to U_x and hence are added to $\mathcal{Y}_x$.

With the initialization of the first testing mapping in each row and the construction of subsequent testing mappings based on the smallest threshold, we bound the total number of fail trials by n (See Lemma 4.1). Therefore, the total time for all trails over all rows is $O(n^2)$.

LEMMA 4.1. *Let $\mathcal{G}$ be the set of saturated rectangles obtained after the fail trials over all rows (in method 1). We have $|\mathcal{G}| \leqslant n$.*

Proof. The lemma is proved by showing (1) every rectangle in $\mathcal{G}$ contains at least one source of S on its right boundary, and (2) the right boundary of a rectangle in $\mathcal{G}$ does not overlap with the right boundary of any other rectangles in $\mathcal{G}$.

Consider a rectangle $R \in \mathcal{G}$. Since R is saturated, it contains at least two sources on each of its four boundaries, e.g., the right boundary. The worry is that these sources are all induced sources in the top-right corner vertex of R. However, the top-right corner vertex of R must contain no more than one induced source because R was once oversaturated in a trial and in next trial at least one of the rightmost induced sources in R is "shifted" out of the right boundary. Therefore, the right boundary of R contains at least one source of S.

Suppose there are two rectangles $R_1 = [t_1, b_1; \ell_1, r]$ and $R_2 = [t_2, b_2; \ell_2, r]$ in $\mathcal{G}$ such that they overlap on column r, i.e., $t_1 \leqslant t_2 \leqslant b_1$ (or $t_2 \leqslant t_1 \leqslant b_2$ but we assume the former one). However, as both rectangles are having the same right boundary, if $t_1 = t_2$, either one of them is not an oversaturated rectangle having the smallest threshold. Hence we assume $t_1 < t_2$. In the following, we show that we can find an oversaturated rectangle in $[t_2 - 1, H; 1, W]$ with respect to U_{t_2-1}, which is a contradiction. By method 2, as $[t_1, b_1; \ell_1, r]$ is saturated, we know that $[t_2 - 1, b_1; \ell_1, r]$ is also saturated. Thus, according to the initialization of the first testing mapping for row $(t_2 - 1)$, the rightmost source on the left of column r, say in $[t_2 - 1, s]$, would be assigned to the right of column r. Let σ be the mapping for row $(t_2 - 1)$ in which we obtain the oversaturated rectangle R_2, and θ be the threshold of R_2. Denote U' the sources including the sources of S on or below row t_2 and the induced sources by σ. Let $R' = [t_2, b'; \ell', r]$ be the oversaturated rectangle with respect to U' such that the threshold θ' of R' equals θ and ℓ' is the smallest (or leftmost) possible. Consider the leftmost source on the right of column ℓ', say in $[t_2 - 1, s']$. This source would be assigned to the left of column ℓ' in σ. This can be proved in a way similar to the proof of Lemma 3.3. Consider the rectangle $R'' = [t_2 - 1, b'; \ell', r]$. Since the rightmost source $[t_2 - 1, s]$ in R'' is assigned to the right of column r, the leftmost source $[t_2 - 1, s']$ in R'' is assigned to the left of column ℓ' and $R' = [t_2, b'; \ell', r]$ is still oversaturated with respect to U', thus we show that R'' is oversaturated with respect to U_{t_2-1}. □

5 Conclusion

We have solved the edge-disjoint escape problem by giving an algorithm to detect if the solution exists and another algorithm to find a set of edge-disjoint paths which connects all n sources to the grid boundary in $O(n^2)$ time. We can also solve the (vertex-disjoint) escape problem by following the same framework in this paper and adopting the concepts of *tilted row*, *tilted column* and *tilted rectangle* as given in [1]. In particular, we show that the absence of oversaturated tilted rectangles is a necessary and sufficient condition for the existence of a solution to the escape problem. Using a similar approach shown in this paper, we can also find the vertex-disjoint paths connecting all n sources to the grid boundary in $O(n^2)$ time.

References

[1] W.-T. Chan and F. Y. L. Chin, *Efficient algorithms for finding maximum number of disjoint paths in grids*, J. Algorithms, (to appear). (A preliminary version of this paper appeared in SODA'97.).

[2] W.-T. Chan, F. Y. L. Chin, and H.-F. Ting, *A faster algorithm for finding disjoint paths in grids*, in Algorithms and Computation, 10th International Symposium, Lecture Notes in Computer Science, Springer, 1999. To appear.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1991.

[4] H. N. Gabow and R. E. Tarjan, *A linear-time algorithm for a special case of disjoint set union*, J. Comput. System Sci., 30 (1985), pp. 209–221.

[5] A. V. Goldberg and S. Rao, *Flows in undirected unit capacity networks*, in 38th Annual Symposium on Foundations of Computer Science, 1997, pp. 32–34.

[6] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, J. Comput. System Sci., 55 (1997), pp. 3–23.

[7] J. M. Y. Leung, T. L. Magnanti, and V. Singhal, *Routing in point-to-point delivery systems: Formulations and solution heuristics*, Transportation Sci., 24 (1990), pp. 245–260.

[8] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, *The point-to-point delivery and connection problems: complexity and algorithms*, Discrete Appl. Math., 36 (1992), pp. 267–292.

[9] G. L. Miller and J. S. Naor, *Flow in planar graphs with multiple sources and sinks*, SIAM J. Comput., 24 (1995), pp. 1002–1017.

[10] V. P. Roychowdhury and J. Bruck, *On finding non-intersecting paths in grids and its application in reconfiguring VLSI/WSI arrays*, in Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990, pp. 454–464.

[11] V. P. Roychowdhury, J. Bruck, and T. Kailath, *Efficient algorithms for reconfiguration in VLSI/WSI arrays*, IEEE Trans. Comput., 39 (1990), pp. 480–489.

[12] T. A. Varvarigou, V. P. Roychowdhury, and T. Kailath, *Reconfigurating processor arrays using multiple-track models: The 3-track-1-spare-approach*, IEEE Trans. Comput., 42 (1993), pp. 1281–1292.

[13] M.-F. Yu and W. W.-M. Dai, *Pin assignment and routing on a single-layer pin grid array*, in Proceedings of 1st Asia and South Pacific Design Automation Conference, IEEE, 1995, pp. 203–208.

[14] ———, *Single-layer fanout routing and routability analysis for ball grid arrays*, in Proceedings of IEEE/ACM International Conference on Computer-aided Design, 1995, pp. 581–586.