CHAPTER III

ALGORITHMS TO SOLVE UCP AND SRP

In this chapter we present algorithms to solve UCP and SRP problems in the two-dimensional grid [16, 18]. Both algorithms are interesting in their own right and are similar to each other. Algorithmic definitions are given and their time complexities are analyzed.

3.1 The Algorithms

We begin with a high-level description of the algorithms. Since, the algorithms for solving UCP and SRP are quite the same, so we present them together. The algorithms can be divided into three phases as follows:

Phase 1. We find locations of all sources in SRP and additionally locations of the two users in UCP at time *k*. We appropriately add and subtract locations of each step of the movement along each axis to find subsequent locations until we arrive at time *k*. Since each source can move with different velocities, we add and subtract positions with velocities in consideration. This takes $\theta((m+2)\cdot k)$ for UCP and take $\theta(m \times k)$ for SRP, where *m* is the number of sources and *k* is the given time.

Phase 2. In both UCP and SRP two sources i and j are currently in range if the distance between their center points is less than $c_i + c_j/2$, where c_i and c_j are the diameters of the circles of the sources i and j, respectively and their overlapping coverage area is not completely contained inside an obstacle. If any circle is completely contained in one outer circle, it suffices to consider the outer circle and ignore the inner circles. Computing an integer boundary (i.e., set of integer coordinates) of the intersected area of the two circles i and j can be done in $O(\min(c_i, c_j))$ time, respectively. Observe that, given the value of *x*(*y*) on the axis *x*(*y*), we can compute its corresponding value of

$y(x)$ of the intersected area of the two circles. Therefore, this computation can be done in constant time $O(\min(c_i, c_j))$.

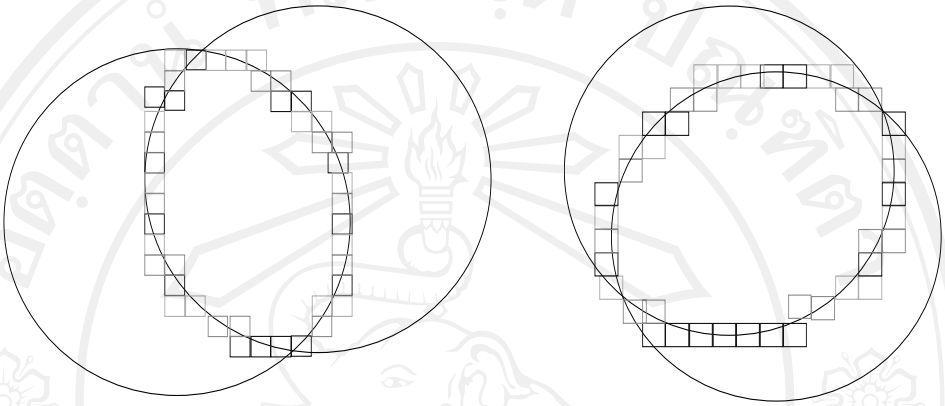

Figure 3.1 Integer boundaries of the intersected area of any two circles.

Let e be the number of intersections and $p_{ij} = \min(c_i, c_j)$. The total time complexity for checking all pairs of intersecting circles i and j is $O(e \times \max(\min(c_i, c_j)))$.

For each pair of intersecting circles, we maintain integer coordinates along its circumference of the intersected area and every integer coordinate exclusively inside it in the set I. We create a matrix A_1 of size just enough to cover the coordinates of all intersected areas of circles. We assume that all matrix elements in matrix A_1 are initially 0's. We assign 2 to every matrix element corresponding to coordinates in I. This step takes $O(e \times \max(\{p_{ij}^2\}))$. Next, we create another matrix A_2, in which all elements are initially 0's, of the same size as that of A_1 representing coordinates of the obstacles. We assign 1 to every matrix element corresponding to the coordinates of all obstacles. This step takes $O(a_{\max} \times |O|)$, where $a_{\max}$ is the area covered by the largest obstacle. For example, we show the given instance of a mobility model, matrix A_1, and matrix A_2 in Figures 3.2, 3.3 and 3.4, accordingly.

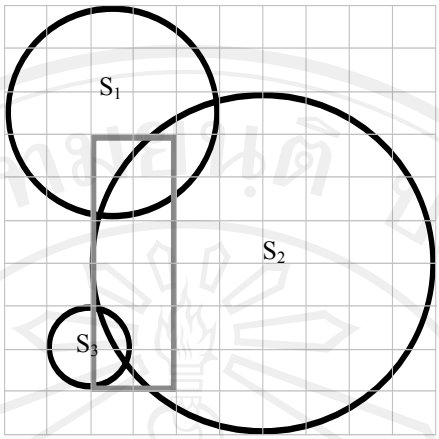

Figure 3.2 The given mobility model in two-dimensional grid.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 3.3 Matrix A_1 represents all intersected areas of circles in Figure 3.2.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 3.4 Matrix A_2 representing coordinates of the obstacles in Figure 3.2.

For each intersected area of two circles represented by a portion [A_1] of matrix A_1, let $A_3 = [A_1] - [A_2]$, where [A_2] is the corresponding portion of matrix A_2. If the resulting matrix A_3 contains at least one 2, it means that the communication is not blocked by an obstacle. For example, if we now check the intersection between sources s_1 and s_2 in Figure 3.2, we will have matrix portions [A_1], [A_2], and the resulting matrix A_3 like in Figure 3.5. It means that the communication is not blocked by an obstacle. If we now check the intersection between sources s_2 and s_3 in Figure 3.2, we will have matrix portions [A_1], [A_2], and the resulting matrix A_3 will be as shown in Figure 3.6. It shows that the communication is blocked by an obstacle.

$$\begin{bmatrix} 0 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 \\ 1 & 1 & 2 \\ 1 & 1 & 2 \end{bmatrix}$$

Figure 3.5 Matrix portions [A_1], [A_2], and the resulting matrix A_3 for the corresponding sources s_1 and s_2 in Figure 3.2.

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Figure 3.6 Matrix portions [A_1], [A_2], and the resulting matrix A_3 the corresponding sources s_2 and s_3 in Figure 3.2.

Repeat the subtraction of matrix portions [A_2] from [A_1] for every intersected area. The time complexity of this step is $O(e \times \max(\{p_{ij}^2\}))$. We model this communication based on a graph theory [11]. We build a graph representing communication. Let sources and users be vertices (points) of the graph and edges (lines) between each pair of vertices be communication.

In UCP a line also exists between a source and a user if and only if the distance of the line between them is less than half of the coverage and is not in any obstacle. Checking for an intersection of a communication line between a user and a source, and an

obstacle can be done by checking, for each axis $j \in \{x, y\}$, the coordinates of the line and the obstacle must *not* be in one of the following patterns: $o_l^j < p_l^j < o_m^j < p_m^j$, $p_l^j < o_l^j < o_m^j < p_m^j$, $o_l^j < p_l^j < p_m^j < o_m^j$, or $p_l^j < o_l^j < p_m^j < o_m^j$, where p_l^j and p_m^j are the two endpoints of the projection of the line between a source and a user on axis *j* and $p_l^j < p_m^j$, and o_l^j and o_m^j are the two endpoints of the projected line of the obstacle on axis *j* and $o_l^j < o_m^j$. To determine whether a line of communication exists between a source and a user takes $O(2 \times |O|)$, we have two axes and check the line with all obstacles. Since there are *m* sources and 2 users, the number of checked lines is equal to $2 \times m$. Therefore, the time we use for checking intersections of a communication line between a user and a source, and an obstacle is equal to $O(4 \times |O| \times m) = O(m)$ because $|O|$ is constant.

In this phase, the total time complexity for SRP is $O(e \times \max(\{p_{ij}\}))$ + $O(e \times \max(\{p_{ij}^2\}))$ + $O(a_{\max} \times |O|)$ + $O(e \times \max(\{p_{ij}^2\}))$ + $O(m)$. Since e equal to m^2 in the worst case, the time complexity is $O(m^2 \times \max(\{p_{ij}\}))$ + $O(m^2 \times \max(\{p_{ij}^2\}))$ + $O(a_{\max} \times |O|)$ + $O(m^2 \times \max(\{p_{ij}^2\}))$ + $O(m)$. We know that $\max(\{p_{ij}\})$, $\max(\{p_{ij}^2\})$, $a_{\max}$ and $|O|$ are constant. Thus, the time complexity for this phase of the algorithm is $O(m^2)$ and is the same for UCP.

Phase 3. Use breadth-first search [14] to find reachability. In UCP, let u_a be the root and find u_b. If u_b is reachable from u_a, we answer "*yes*" and "*no*," otherwise. In SRP, let s_a be the root and find s_b. If s_b is reachable from s_a, we answer "*yes*" and "*no*," otherwise. The running time of the breath-first search is $O(|V| + |E|)$, where $|V|$ is the number of vertices equal to *m* + 2 in UCP and equal to *m* in SRP and $|E|$ is the number of edges. In the worst case, the *maximum* number of edges is the number of all edges between all pairs of points but u_b. The number of edges are

$(g-2)+(g-3)+\cdots+1=\frac{(g-2)\times(g-1)}{2}=O(g^2)$, g is the number of points. In UCP and SRP, g is equal to m+2 and m, respectively. So, the time complexities for UCP is $O((m+2)^2)=O(m^2)$ and $O(m^2)$ for SRP.

Note that these algorithms can also be slightly modified to use in the three-dimensional case because the "essence" of the algorithms is identical in both two and three dimensions.

3.2 Time Complexities of UCP and SRP

In this section we show that our two algorithms are time optimal. This implies that our algorithms are *as fast as* they can be. We prove this fact by establishing an asymptotically tight lower bound to match the worst-case complexities of our algorithms.

Theorem 3.1: The time complexity for the algorithm for User Communication Problem in a two-dimensional grid is $O(m^2)$, where m is the number of sources.

Proof: From phases 1 to 3, the running time of the algorithm is $\theta((m+2)\cdot k)+O(m^2)+O(m^2)=O(m^2)$ since k and $|O|$ are at most constant. Thus, the theorem holds.■

Theorem 3.2: The time complexity for the algorithm for Source Reachability Problem in a two-dimensional grid is $O(m^2)$, where m is the number of sources.

Proof: From phases 1 to 3, the running time of the algorithm is $\theta(m\times k)+O(m^2)+O(m^2)=O(m^2)$ since k and $|O|$ are at most constant. Thus, the theorem holds.■

Theorem 3.3: The lower bound for User Communication and Source Reachability Problems in a two-dimensional grid is $\Omega(m^2)$, where m is the number of sources.

Proof: We model the communication in the mobility model with a graph. Let $G=(V,E)$ be a graph, where V is a set of vertices and $E=E_{red}\cup E_{green}$ is a set of edges.

A vertex in a graph represents a source. A pair of sources that can communicate is represented by an edge in E_{green} between the two corresponding vertices and an edge in E_{red} represents a pair of sources that cannot communicate. Hence, to find whether sources s_a and s_b can communicate is to find a path between s_a and s_b whose composite edges are *all* in E_{green}. We claim that *all* edges in *E* must be checked in order to find such a path because an unchecked edge may be in E_{green} and is part of the communicating path between s_a and s_b.

For example, the given instances of the model are shown in Figure 3.7. Let gray edges represent E_{green} and black edges represent E_{red}. If we do not check $\overline{es}_b$, as illustrated in Figure 3.8, the answer is s_a and s_b cannot communicate, and it is a wrong answer. Thus, *all* edges in *E* must be checked.

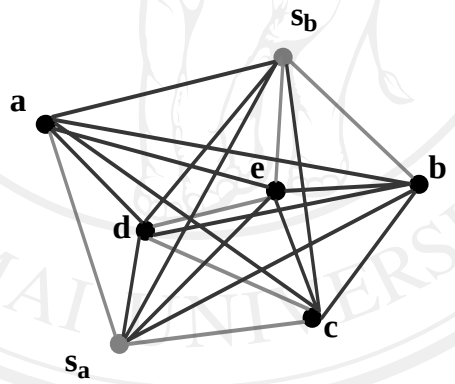

Figure 3.7 An instance of the problem.

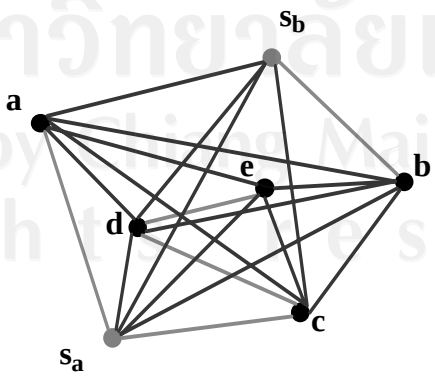

Figure 3.8 Checking all edges except $\overline{es}_b$.

Because $|E| = m(m-1)/2$, the lower bound is $\Omega(m^2)$. Therefore, the theorem holds.■

Note that we do not consider users u_a and u_b of the UCP in Theorem 3.3 because a user must be in the coverage of some source in order to communicate.

Theorem 3.4: The algorithms for User Communication and Source Reachability Problems in a two-dimensional grid is time-optimal.

Proof: This result is true by the implication of Theorems 3.1, 3.2, and 3.3.■