sensors

MDPI

Article

Efficient Algorithms for Max-Weighted Point Sweep Coverage on Lines

Dieyan Liang and Hong Shen *

School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510275, China; ldiey@mail2.sysu.edu.cn
* Correspondence: shenh3@mail.sysu.edu.cn

Abstract: As an important application of wireless sensor networks (WSNs), deployment of mobile sensors to periodically monitor (sweep cover) a set of points of interest (PoIs) arises in various applications, such as environmental monitoring and data collection. For a set of PoIs in an Eulerian graph, the point sweep coverage problem of deploying the fewest sensors to periodically cover a set of PoIs is known to be Non-deterministic Polynomial Hard (NP-hard), even if all sensors have the same velocity. In this paper, we consider the problem of finding the set of PoIs on a line periodically covered by a given set of mobile sensors that has the maximum sum of weight. The problem is first proven NP-hard when sensors are with different velocities in this paper. Optimal and approximate solutions are also presented for sensors with the same and different velocities, respectively. For M sensors and N PoIs, the optimal algorithm for the case when sensors are with the same velocity runs in $O(MN)$ time; our polynomial-time approximation algorithm for the case when sensors have a constant number of velocities achieves approximation ratio $\frac{1}{2}$; for the general case of arbitrary velocities, $\frac{1}{2\alpha}$ and $\frac{1}{2}(1-1/e)$ approximation algorithms are presented, respectively, where integer $\alpha \geq 2$ is the tradeoff factor between time complexity and approximation ratio.

Keywords: WSN; mobile sensors; sweep coverage; approximation algorithm; combinatorial mathematics

Citation: Liang, D.; Shen, H. Efficient Algorithms for Max-Weighted Point Sweep Coverage on Lines. *Sensors* **2021**, *21*, 1457. https://doi.org/10.3390/s21041457

Academic Editor: Francesco Marcelloni

Received: 7 January 2021
Accepted: 13 February 2021
Published: 19 February 2021

1. Introduction

Coverage is one of the most important applications of wireless sensor networks (WSN), where sensors are placed on an area of interest to monitor the environment and detect extraordinary activities. There have been many studies on this topic across different subject areas including discrete points [1,2], 2-dimensional surfaces [3], 3-dimensional surfaces [4], 3-dimensional spaces, fences [5,6], and so on. Based on sensors' characteristic, special factors should be considered, such as energy efficiency, maintaining connectivity, and so on [7–10].

However, most of the existing works have mainly focused on continuous coverage, where sensors stay still after begin placed on their objective locations. Only recently has sweep coverage been brought up for the periodical coverage situation, which arises in many applications. For example, guards need to patrol a barrier periodically according to the time needed for intruders to cross it; Information collectors should collect information from the objective sensors periodically to avoid their memory overflow. In those situations, the objects do not need to be covered continuously. So, mobile sensors can move around to cover more objects than in the continuous coverage situation to reduce monitoring cost. For its cost effectiveness, sweep coverage has attracted increasing attention [11–15].

Point sweep coverage was first brought up in Reference [11], in which the authors introduced the problem of deploying the minimum number of mobile sensors to cover a given set of points of interest (PoIs) in the Euclidean space. The problem was shown Non-deterministic Polynomial Hard (NP-hard), and could not be approximated within the factor of 2, even for the case that all sensors are with the same velocity. In practice, with the limited energy, the velocity of a mobile sensor decreases as energy consumption increases. It is more realistic to consider the general case that mobile sensors are with arbitrary velocities. However, in Reference [16], the authors proved that no polynomial-time constant-factor approximation algorithm exists to solve this problem for mobile sensors with different velocities unless $P = NP$. In this paper, we study a variant of this problem, named Max-weighted Point Sweep Coverage, to find the set of PoIs that have the maximum sum of weight, periodically covered by a given set of sensors with different velocities. We show that it is NP-hard even when the PoIs are distributed on a line. In applications, since resources are always limited, it is desirable to use a given set of sensors to cover as many PoIs as possible. The Max-weighted Point Sweep Coverage problem aims at maximizing the utilization of the given mobile sensors for addressing these application requirements. In addition, PoIs located on a line is a common scenario in many applications. For example, as illustrated in Figure 1, for ocean information stored in static sensors placed at key locations along a line, we need to deploy a set of mobile sensors to collect the data from the static sensors periodically to avoid static sensors memory overflow. Other applications can be found in security, forest conservation, resource exploration, and so forth. Therefore, we focus on the Max-weighed Point Sweep Coverage on the line (MPSCL) problem and discuss the proper polynomial-time algorithms in different cases.

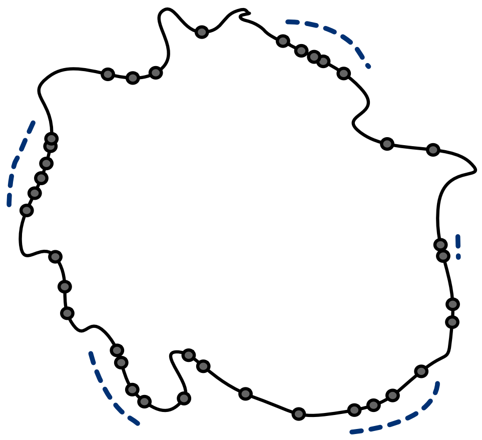


Figure 1. Mobile sensors sweep coverage around island to collect the data from the static sensors.

In this paper, we define the MPSCL problem, prove that it is NP-hard by showing that a special case of its decision version is NP-complete (NPC), and present optimal and approximation algorithms for the cases of mobile sensors with the same and different velocities, respectively.

The main contributions of this paper are summarized as follows:

- We define the MPSCL problem and prove it is NP-hard through a reduction from the 3-Partition problem.
- For the special cases of the MPSCL problem when sensors have the same velocity, we present an optimal algorithm applying dynamic programming.
- For the special cases of the MPSCL problem when sensors have a constant number of velocities, we present a $\frac{1}{2}$-approximation algorithm by extending the solution for the same-velocity case.
- For the general cases of the MPSCL problem when sensors have arbitrary velocities, we propose three approximation algorithms. One achieves approximation ratio $\frac{1}{2\alpha}$ by velocity rounding, where integer $\alpha \geq 2$ is the tradeoff factor between time complexity and approximation ratio. The second and third one are, respectively, a random and a deterministic $\frac{1}{2}(1 - 1/e)$ approximation algorithm by randomized rounding and derandomized technique.

The rest of this paper is organized as follows: Section 2 describes some related work. In Section 3, the definition and NP-hardness proof of the MPSCL problem are given.

In Section 4, we present our optimal and approximation algorithms for different cases of the MPSCL problem. Section 5 presents the simulation results and investigates the performance of the algorithms. Section 6 concludes the paper.

2. Related Work

The point sweep coverage was firstly brought up in Reference [11]. The authors presented the Min-Sensor Sweep Coverage problem (MSSC) to find the minimum number of sensors to sweep cover PoIs in Eulerian graph, which was proven NP-hard by transforming the Traveling Salesman Problem (TSP) to it when the mobile sensors were with the same velocity. The problem could not be approximated within ratio 2 and local algorithms could not work. In Reference [17], the authors distinguished sensors' strategies, proposed MinExpand algorithm for un-cooperated sensors and Osweep algorithm for cooperated sensors, respectively. A mistake of approximation analysis of prior papers was rectified by Gorain et al. [16], in which a 3-approximation algorithm for the MSSC problem was proposed and it was still the best approximation algorithm for MSSC till now. Non-existence of polynomial-time constant-factor approximation algorithms for MSSC when sensors were with different velocities was also proven. Some variants of MSSC were presented. When PoIs had different sweep periods, an $O(log\rho)$-approximation algorithm was proposed, where ρ was the ratio of the maximum and minimum sweep periods among PoIs. The area sweep coverage problem and line sweep coverage problem were proposed and shown NP-hard, and approximation algorithms of ratio $(\sqrt{2} + \frac{2-\sqrt{2}}{mn})$ and 2 were proposed, respectively [18,19]. In Reference [20], a variation of the MSSC problem called the DistanceSensitive-Route-Scheduling problem was studied, where the impact of sensing range was taken into account. The impact of sensing range in the sweep coverage problems shortened the trajectory length of mobile sensors to reduce needed sensors. In Reference [21], the authors assumed the consumption of energy was different between mobile sensors and static sensors, and proposed two variations of the MSSC problem. One was the energy efficient sweep coverage problem to minimize the total energy consumption in every unit of time, which could not be approximated within a factor of 2, and an 8-approximation algorithm for that was proposed. Another was the energy restricted min-sensor sweep coverage problem, for which a $(5 + \frac{2}{\alpha})$-approximation algorithm was proposed. Gorain et al. took barrier sweep coverage into account [22]. They presented a energy restricted barrier sweep coverage problem and proposed $\frac{13}{3}$ approximation algorithm.

The concept of sweep coverage appeared in the contexts of robotics concerned sweep covering continuous lines, and the problem was called boundary patrolling or fence patrolling. In these contexts, the mobile sensors might be with different velocities and the aim was to find the minimum idleness, i.e., the longest time interval during which there was at least one point on the boundary uncovered by any mobile sensors. In Reference [23], the authors firstly studied boundary patrolling problem and proposed two intuitive algorithms for open and close fence patrolling. The optimality of these algorithms was disproved in Reference [24–26], in which the examples were proposed to illustrate that the idleness could be reduced to 41/42, 24/25, and even 3/4 by special design, assuming the idleness of proportional solution presented in Reference [23] was 1. Even though the optimality of algorithms presented by Czyzowicz et al. was disproved, the optimal solution had not been brought up yet. In Reference [27], the authors extended the scenes of the min-idleness sweep coverage problem to chains, trees, and cyclic roadmaps when the mobile sensors were identical. Within tolerance ϵ, they could get an optimal idleness when PoIs were on a chain in time complexity $O(nlog(\epsilon^{-1}))$. And an 8-approximation algorithm was proposed to find min-idleness when PoIs were on cyclic roadmap for which the problem was NP-hard. In Reference [28], the authors described a fragmented boundary environment and found the optimal patrolling for min-idleness in that environment when the sensors were with the same velocity. Min idleness problem was also called as min-period problem. Gao et al. studied Min-peroid Sweep Coverage (MPSC) problem when the sensors did not cooperate with each other [29], in which, they proposed a nearly 5-appr algorithm for MPSC when the sensors with the same velocity covered target points on 2-D plane and a 5α-appr algorithm when the sensors have different velocities, where α was the ratio of the maximum velocity to the minimum one. They also had extended it to the scene where a graph needed to be covered. In Reference [30], Gao et al. studied Cooperative Sweep Coverage (CSC) problem, when the sensors with the same velocity, and proposed a 4-appr algorithm for general case and an optimal solution for the special case when PoIs were on a line. They also considered the situation in which each track cycle must cover at least one sink.

3. Preliminary and Problem Statement

In this section, we will study the Max-weighted Point Sweep Coverage on Lines (MPSCL) problem. We first give its definition and then prove it is NP-hard.

Definition 1. *(Max-weighted Point Sweep Coverage on Lines) Given a set of M mobile sensors $\mathcal{S} = \{s_1, s_2, .., s_M\}$ with velocities $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$ and a set of PoIs $\mathcal{P} = \{p_1, p_2, .., p_N\}$ distributed on a line, where each p_i needs to be monitored every time period T, i.e., T-sweep covered, find a set of PoIs of maximum summed weight that is T-sweep covered by the given set of mobile sensors.*

Note that static sensors are regarded as special mobile sensors with velocity 0. For simplicity, we sometimes say "cover" or "sweep cover" to replace "T-sweep cover".

In the existing work, two kinds of strategies are proposed, separation strategy [28] and cooperation strategy [27], respectively. Their definitions are below:

Definition 2. *(separation strategy) Mobile sensors move back and forth on the line segment to sweep cover PoIs without cooperating with others, where each PoI would meet the same sensor within time period T.*

Definition 3. *(cooperation strategy) Sensors cooperate with others to cover the same line segment, i.e., the PoIs on the line segment would meet another sensor within T time after covered by one sensor last time.*

Under separation strategy, every sensor $s_i \in \mathcal{S}$ has its own coverage $r_i = v_i T/2$. So, a separation strategy can be expressed by the first PoI's location of every sensor's coverage. Wthout loss of generality, we assume that the set of PoIs are located on the x-axis at coordinates $\mathcal{X} = \{x_1 = 0, x_2, \ldots, x_N\}(x_1 < x_2 < \cdots < x_N)$, and the coordinate of PoI l is x_l. The deployment of sensor s_i can be described as $[x_l, x_l + v_i T/2]$, $l \in [1, N]$.

For cooperation strategy, the path of each sensor is a periodical folded line, which is more complicated to be expressed than the separation strategy. Separation strategy may not result in an optimal solution. Some examples show that separation strategy may be slightly worse than cooperation strategy in some special cases, i.e., the covering range covered by a set of mobile sensors under separation strategy is slightly shorter [24,25]. For example, in Reference [25], six sensors with velocities $\{1, 1, 1, 1, 7/3, 1/2\}$ can 1-sweep cover PoIs on a line segment of length $7/2$ under a cooperation strategy, which is longer than that of separation strategy, $(1 + 1 + 1 + 1 + 7/3 + 1/2)/2 = 41/12$. However, compared to the complication of coorperation strategy, separation strategy is easier to be analyzed.

The notations are summarized in Table 1.

Table 1. Notations.

Symbol	Definition
N	the number of the PoIs
M	the number of the sensors
$\mathcal{P}$	the set of PoIs $\{p_1, p_2, .., p_N\}$
$\mathcal{S}$	the set of mobile sensors $\{s_1, s_2, .., s_M\}$
$\mathcal{X}$	the locations of PoIs $\{x_1, x_2, \ldots, x_N\}$ $(x_1 < x_2 < \cdots < x_N)$
T	the sweep period of PoIs
$\mathcal{V}$	velocities of the sensors $\{v_1, v_2, \ldots, v_M\}$
$\mathcal{W}$	the weight of the PoIs $\{\omega_1, \omega_2, \ldots, \omega_N\}$
$\mathcal{P}_{ij}$	the set of PoIs located on coordinate interval $[x_j, x_j + Tv_i/2]$
n_{ij}	the number of PoIs in $\mathcal{P}_{ij}$
ω_{ij}	the summed weight of PoIs in $\mathcal{P}_{ij}$
W	a random variable of the summed weight of the PoIs covered
Y_{ij}	random variables denote that sensor i covers the set $\mathcal{P}_{ij}$
$v_{\max}$	the largest velocity in $\mathcal{V}$
$v_{\min}$	the smallest non-zero velocity in $\mathcal{V}$

3.1. Problem Hardness

In this section, we present the definition of a special case of the MPSCL problem when PoIs' weight is the same as Definition 4. By proving that it is NP-complete, we show the MPSCL problem is NP-hard.

Definition 4. *(Point Sweep Coverage on Lines) Given a set $\mathcal{P}$ of PoIs on lines, a set $\mathcal{S}$ of mobile sensors, and a positive integer $K \le |\mathcal{P}|$, is there a strategy for sensors $\mathcal{S}$ such that no less than K PoIs are sweep covered?*

Our NP-completeness proof is based on a reduction of the following 3-Partition problem, which is well-known NP-complete to the Point Sweep Coverage on Line (PSCL) problem.

Definition 5. *(3-Partition) [31]. INSTANCE: Set A of $3m$ elements, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$. QUESTION: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \le k \le m$, $\sum_{a \in A_k} s(a) = B$ (note that each A_i must therefore contain exactly three elements from A) ?*

Theorem 1. *Point Sweep Coverage on Lines problem is NP-complete.*

Proof. Given a 3-Partition instance, like Definition 5, we construct a MPSCL instance. Given a set of N PoIs with their positions $\{x_1, x_2, \ldots, x_N\}(x_1 < x_2 < \cdots < x_N)$, $N = (2B+2) \times m$. The positions of PoIs satisfy the equation below, where $d_i = x_{i+1} - x_i (1 \le i < N)$ means the distance between the $(i+1)$th PoI and ith PoI.

$$x_1 = 0 \tag{1}$$

$$d_i = B, \quad (i = k \times (2B+2); 1 \le k \le m-1) \tag{2}$$

$$d_i = B/(2B+1), \quad ((k-1) \times (2B+2) + 1 \le i < k \times (2B+2); 1 \le k \le m). \tag{3}$$

Given a set $\mathcal{S}$ of $M = 3m$ mobile sensors, the velocity of mobile sensor s_j is $v_j = 2 \times s(a_j)/T$ for $a_j \in A$ $(1 \le j \le M)$. As mentioned before, if under separation strategy, each mobile sensor has its own covering range $r_j = v_j T/2 = s(a_j)$, $1 \le j \le M$. Then, $B/4 < r_j < B/2$, $\sum_{1 \le j \le M} r_j = mB$.

The object of the PSCL instance is to cover at least $K = N$ PoIs.

Let B_k denote the line segment from $x_{(k-1)\times(2B+2)+1}$ to $x_{k\times(2B+2)}$ for $1 \leq k \leq m$. Because of Equation (3), we get $|B_k| = B$. If the 3-Partition instance is satisfied, we get $\sum_{a\in A_k} s(a) = B$, where each A_k contains exactly three elements from A. It means, if we can obtain proper 3 mobile sensors to sweep cover each B_k for $1 \leq k \leq m$, then we get a proper deployment for the PSCL problem.

Conversely, for the other side, if we have an unique solution for the PSCL problem, because of Equation (2), mobile sensors s_j $(1 \leq j \leq M)$ must cover some part of line segment B_k $(1 \leq k \leq m)$ since the gap between B_k and $B_{k'}$ $(k \neq k')$ is too big. Because $B/4 < r_j < B/2$ and $r_j \in Z^+$ for $1 \leq j \leq M$, more than 2 sensors are needed to cover each segment B_k $(1 \leq k \leq m)$. Considering there are $3m$ sensors for m line segments $B_k(1 \leq k \leq m)$, we get exactly 3 mobile sensors to cover each segment B_k $(1 \leq k \leq m)$. The separation strategy is optimal for three mobile sensors [25]. Thus, for arbitrary B_k $(1 \leq k \leq m)$, assuming covering ranges of the 3 mobile sensors are r_{j1}, r_{j2}, r_{j3}, respectively, the best total covering range of the 3 mobile sensors is

$$r_{j1} + r_{j2} + r_{j3} \geq B - 2B/(2B+1).$$

Otherwise, the three sensors are not enough to cover all the PoIs on B_k. And because $r_{j1}, r_{j2}, r_{j3} \in Z^+$, we get $r_{j1} + r_{j2} + r_{j3} \geq B$. If $r_{j1} + r_{j2} + r_{j3} > B$, there must exist $r_{j'1} + r_{j'2} + r_{j'3} < B(j' \neq j)$, making a contradiction. So, $r_{j1} + r_{j2} + r_{j3} = B$, satisfying the solution to the 3-Partition problem.

Note that the 3-Partition problem is strongly an NPC problem, which means it is also NPC even if B is bounded by polynomial in m. So, when B is bounded by polynomial in m, the instance of PSCL can be constructed from an arbitrary 3-Partition instance in polynomial time. Now, we have shown the reduction from the 3-Partition problem to the PSCL problem. An example is illustrated in Figure 2.

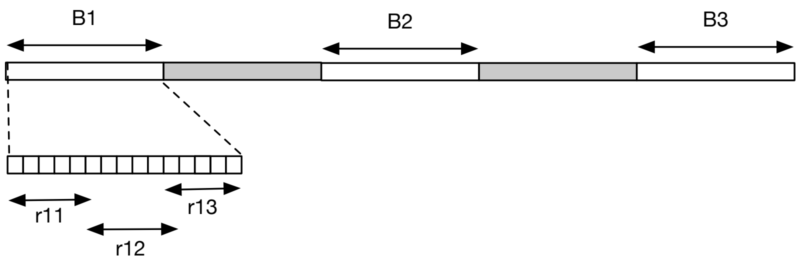


Figure 2. The instance of Point Sweep Coverage on Line (PSCL) when $m = 3$, $B = 7$, $K = 48$. It contains 48 points of interest (PoIs) and 9 sensors. The PoIs are distributed on subsegment B_j $(1 \leq j \leq 3)$. When the covering range of sensor s_i is $\frac{7}{4} \leq r_i \leq \frac{7}{2}$ $(1 \leq i \leq 9)$, the proper deployment is to deploy 3 sensors to each B_j.

Given the strategies of sensors, it is easy to check if or not there are K PoIs are in the coverage of the sensors in polynominal time. Thus, the decision version of the PSCL problem is in NPC.

The theorem is proven. □

Hence, we have:

Corollary 1. *The Max-weighted Point Sweep Coverage Problem is NP-hard.*

4. Algorithms

In this section, we present an optimal solution for the special case of sensors with same velocity and approximation solutions for sensors with different velocities, respectively. We assume that the number of PoIs is greater than that of mobile sensors, i.e., $N > M$; otherwise, the problem has a trivial solution.

4.1. Optimal Solution for Sensors with Same Velocity

In this subsection, we show that there is a polynomial-time optimal solution for MPSCL when sensors have the same velocity.

Reference [27] showed that separation strategy can yield an optimal solution when PoIs on the line are sweep covered by mobile sensors with the same velocity as Theorem 2 because, when two sensors in opposite directions meet each other, they can "exchange" roles.

Theorem 2. *Separation strategy yields an optimal solution for the Max-weight Sweep Coverage on Lines problem if the given set of mobile sensors have the same velocity.*

In our algorithm, we first apply a dynamic programming algorithm to find the optimal separation strategy of MPSCL. According to Theorem 2, it is also an optimal solution for MPSCL. In the separation strategy, sensors have the same size of coverage with the same velocity v. So, in our algorithm, we can first judge whether the given set of mobile sensors can sweep cover all the PoIs by examining the coverage from the 1st to the Nth PoI without overlapping. If "yes", the given set of mobile sensors can sweep cover all the PoIs and the maximum weight is the summed weight of all the PoIs. Otherwise, we apply dynamic programming to obtain the optimal solution as follows.

Let $OPT(i,j)$ denote the maximum summed weight of the PoIs covered by i sensors from the jth to PoI N. n_j denotes the number of PoIs covered by one sensor from the jth PoI, i.e., the number of PoIs located in coordinate interval $[x_j, x_j + vT/2]$, and ω_j is the summed weight of the n_j PoIs. The recursive formulation of $OPT(i,j)$ is given below:

$$OPT(i,j) = \quad max\left\{ \begin{array}{c} OPT(i,j+1), \\ OPT(i-1,j+n_j)+\omega_j \end{array} \right\}, \quad 1 \le i \le M, 1 \le j < N-1,$$

with boundary conditions

$$\begin{cases} OPT(0,j) = 0 & 0 \le j \le N \\ OPT(i,N) = 1 & 1 \le i \le M \end{cases}.$$

The first step of our algorithm, deciding whether all the PoIs can be covered, takes $O(N)$ time. The time complexity of the dynamic programming is $O(MN)$. Thus, the time complexity of the optimal algorithm for MPSCL with the same velocity is $O(MN)$. The algorithm is straightforward; hence, its description is omitted.

4.2. $\frac{1}{2}$-Approximation Solution for Sensors with a Constant Nunber of Velocities

Now, we discuss the MPSCL problem when sensors have K different velocities, called K-velocity MPSCL, where K is a constant. In fact, it is not hard to see that the uniform velocity case of the MPSCL problem is a special case of K-velocity MPSCL that $K = 1$. The only difference is that separation strategy is not an optimal strategy any more when $K \ge 2$ and $M \ge 4$ [25]. However, it is easy to find the set of PoIs covered with the maximum summed weight under separation strategy since it has the nature of optimal substructure. So, we use the dynamic programming method to find an optimal separation strategy and prove it is a $\frac{1}{2}$-approximation algorithm for the K-velocity MPSCL problem considering the difference between the separation strategy and cooperation strategy.

Let m_i be the number of sensors with velocity v_i, then $M = \sum_{i=1}^{K} m_i$. $OPT_S(i_1, i_2, \ldots, i_K, j)$ denotes the maximum summed weight of the covered PoIs when there are $\sum_{h=1}^{K} i_h$ sensors to cover the line segment from the PoI j to PoI N under separation strategy, where i_h is the number of sensors with velocity v_h $(1 \le h \le K)$. Denote the number of PoIs covered by a sensor with velocity v_i from the PoI j by n_{ij}, i.e., the number of PoIs located in coordinate interval $[x_j, x_j + v_i T/2]$, and the summed weight of the n_{ij} PoIs by ω_{ij}. Below is the recursive formulation of the solution:

$$OPT_S(i_1, i_2, \ldots, i_K, j) = max \left\{ \begin{array}{c} OPT_S(i_1, i_2, \ldots, i_K, j+1), \\ OPT_S(i_1 - 1, i_2, \ldots, i_K, j + n_{1j}) + \omega_{1j}, \\ OPT_S(i_1, i_2 - 1, \ldots, i_K, j + n_{2j}) + \omega_{2j}, \\ \ldots, \\ OPT_S(i_1, i_2, \ldots, i_K - 1, j + n_{Kj}) + \omega_{kj} \end{array} \right\}, \quad \begin{array}{c} 1 \le i_h \le m_h \ \& \\ 1 \le h \le K \ \& \\ 1 \le j \le (N-1) \end{array} .$$

The boundary conditions are

$$\begin{cases} OPT_S(0, 0, \ldots, 0, j) = 0 & 0 \le j \le N \\ OPT_S(i_1, i_2, \ldots, i_K, N) = 1 & 0 \le i_h \le m_h, 1 \le h \le K, \sum_{h=1}^{K} i_h \ge 1 \end{cases}.$$

In Algorithm 1, we maintain a $(K+1)$-dimensional table Tb to record the optimal value $OPT_S(i_1, i_2, \ldots, i_K, j)$ for $i_h \in [0, m_h]$ and $1 \le h \le K$, $1 \le j \le N$. By tracing back the information of table Tb, we can obtain the optimal separation strategy, in which every sensor's location is recored in an array entry $Lc(h)$ for $h = 1, \ldots, K$. Each entry $Lc(h)$ is a list of m_h locations for the m_h sensors with velocity v_h. We say the location of a sensor is the location of the first PoI in the its coverage. In separation strategy, given the velocity v_i and its location, the deployment of sensor s_i is set.

In Algorithm 1, it takes $O(N \times \prod_i (m_i + 1))$ time to construct table Tb, and takes $O(N \times K)$ time to trace back to get the optimal solution. $N \times \prod_i (m_i + 1) \le N \times (M/K + 1)^K$, so the time complexity of Algorithm 1 is $O(N \times (M/K + 1)^K)$. Because K is a constant integer, the algorithm is polynomial time algorithm. Now, we show that Algorithm 1 is an $\frac{1}{2}$-approximation algorithm for the K-velocity MPSCL problem. Before that, we give a more general theorem, showing that the β-approximation algorithm for the optimal separation strategy of MPSCL can yield a $\frac{\beta}{2}$-approximation algorithm for MPSCL.

Theorem 3. *A β-approximation algorithm for the optimal separation strategy of MPSCL can be turned to be a $\frac{\beta}{2}$-approximation algorithm for MPSCL, where $\beta \le 1$.*

Proof. Denote by A the β-approximation algorithm for the optimal separation strategy of MPSCL, and by A_o the optimal algorithm for MPSCL. W.l.o.g., assume A_o covers two sets of line segments L_1 and L_2 by the separation strategy and cooperation strategy, respectively.

For L_1, clearly the sum of weight of the covered PoIs by A_o, OPT_1, cannot exceed that covered by the optimal separation strategy, OPT_1^S, and hence $1/\beta$ of that by A, $APPROX_1^S$, using the same set of sensors : $OPT_1 \le OPT_1^S = \frac{1}{\beta} \times APPROX_1^S \implies APPROX_1^S \ge \beta \times OPT_1$.

For L_2, because the coverage $r(S')$ of any set of sensors $\mathcal{S}' \subseteq S$ by the cooperation strategy is bounded by $\sum_{i=1}^{|S'|} v_i T$, which is twice of the coverage by the separation strategy, the sum of weight of the covered PoIs by A_o, OPT_2, cannot exceed twice of the optimal separation strategy, OPT_2^S, and hence $2/\beta$ of that by A,$APPROX_2^S$, using the same set of sensors : $OPT_2 \le 2 * OPT_2^S = \frac{2}{\beta} \times APPROX_2^S \implies APPROX_2^S \ge \frac{\beta}{2} \times OPT_2$.

Summing up the above immediately yields the $\frac{\beta}{2}$ approximation ratio of algorithm A to the optimal algorithm for MPSCL. □

Theorem 4. *Algorithm 1 is a $\frac{1}{2}$-approximation algorithm for the K-velocity MPSCL problem.*

Proof. Algorithm 1 is for the optimal separation strategy of the K-velocity MPSCL. According to Theorem 3, Algorithm 1 is a $\frac{1}{2}$-approximation algorithm for the K-velocity MPSCL problem. □

Algorithm 1 MPSCL-K-Velocities

```
Input: A set of sensors $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$, A set of PoI $\mathcal{P}$ with locations $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and weight $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_N\}$, sweep period $T$, locations $^*Lc$.
Output: The sweep covered set $\mathcal{P}_S$ of PoIs.
1:  //initialization
2:  group the velocities $\mathcal{V}$ into $K$ different velocities $\{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_K\}$
3:  let $m_i$ is the number of sensors with velocity $\bar{v}_i$
4:  for $i \leftarrow 1$ to $K$ do
5:    for $j \leftarrow N$ to 1 do
6:      count the number $n_{ij}$ of PoIs covered by a sensor with velocity $\mathcal{V}_i$ and location $x_j$
7:      let $\omega_{ij}$ be the sum of weight of the $n_{ij}$ PoIs;
8:    end for
9:  end for
10: //dynamic programming to obtain the maximum summed weight
11: initial table $Tb$ according to boundary conditions
12: for $j \leftarrow N$ to 1 do
13:   for $i_1 \leftarrow 1$ to $m_1$ do
14:     for $i_2 \leftarrow 1$ to $m_2$ do
15:       for ... do
16:         for $i_K \leftarrow 1$ to $m_K$ do
17:           call the recursive formulation to calculate $OPT_S(i_1, i_2, \ldots, i_K, j)$, recorded in $Tb$
18:         end for
19:       end for
20:     end for
21:   end for
22: end for
23: // tracing back to obtain the optimal solution
24: set $i_1 = m_1, i_2 = m_2, \ldots, i_K = m_K$
25: initial array entry $Lc$
26: for $j \leftarrow 1$ to $N - 1$ do
27:   if $OPT_S(i_1, i_2, \ldots, i_K, j)! = OPT_S(i_1, i_2, \ldots, i_K, j+1)$ then
28:     for $h \leftarrow 1$ to $K$ do
29:       if $i_h > 0$ && $OPT_S(i_1, i_2, \ldots, i_h, \ldots, i_K, j) == O(i_1, i_2, \ldots, i_h - 1, \ldots, i_K, j + n_{i_h j}) + \omega_{i_h j}$ then
30:         add $x_j$ to $Lc(h)$; $j += n_{i_h j}$
31:         $i_h --$
32:       end if
33:     end for
34:   end if
35: end for
36: if exist any sensor with velocity $\bar{v}_i$ unoccupied then
37:   add $x_N$ to $Lc(i)$
38: end if
39: place sensors to their corresponding locations in $Lc$, the union set of covered PoIs is $\mathcal{P}_S$
40: return $\mathcal{P}_S$
```

4.3. Approximation Solutions for the General Case of Arbitratry-Velocity Sensors

In this subsection, we discuss MPSCL for the general case, i.e., when sensors have arbitrary velocities and propose three methods. The first method uses rounding and dynamic programming technique and yields a $\frac{1}{2\alpha}$-approximation scheme solution, where integer $\alpha \geq 2$. The second uses linear programming relaxation and randomization technique to yield a randomized $\frac{1}{2}(1 - 1/e)$-approximation algorithm. Applying the conditional expectations method, we can get the third one, a deterministic $\frac{1}{2}(1 - 1/e)$-approximation algorithm by derandomization.

4.3.1. Dynamic-Programming Solution with Velocity Rounding

In the previous subsection, we know Algorithm 1 is the optimal algorithm for the separation strategy of the K-velocity MPSCL problem and takes $O(N \times (M/K+1)^K)$ time. In the general case, M sensors would have M different velocities, i.e., $K = M$. Thus, Algorithm 1 would find the optimal separation strategy within the time complexity $O(N * 2^M)$, which is too high. So, we use the rounding method to reduce the number of velocities in order to reduce the time complexity.

In Algorithm 2, for a given set of sensor $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \dots, v_M\}$, we first round the velocities $\mathcal{V}$ to $\mathcal{V}' = \{v'_1, v'_2, \dots, v'_M\}$ by applying the formula (4), which contains $K \ll M$ different velocities. Then, we run Algorithm 1 on a set of M sensors $\mathcal{S}$ with rounded velocities $\mathcal{V}'$ to obtain their locations. Finally, we shift the locations of sensors to avoid their coverage overlapping. That is, since $v_i \geq v'_i$ for $1 \leq i \leq M$, there may be overlap between two sensors' coverage. If so, move the right sensor a minimum distance toward right so that it covers from the next PoI. The shift would increase the number of PoIs covered and the summed weight without changing the approximation ratio .

Algorithm 2 MPSCL-Velocity-Rounding

Input: A set of sensors $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \dots, v_M\}$, A set of PoI $\mathcal{P}$ with locations $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ and weight $\mathcal{W} = \{\omega_1, \omega_2, \dots, \omega_N\}$, sweep period T.
Output: The sweep covered set $\mathcal{P}_S$ of PoIs.

```
1: set $d^{(\alpha)}_{\min} = x_{\alpha+1} - x_1$;
2: for $i \leftarrow 2$ to $N - \alpha$ do
3:     $d^{(\alpha)}_{\min} = min\{d^{(\alpha)}_{\min}, x_{i+\alpha} - x_i\}$;
4: end for
5: set $v_d = max\{v_{\min}, \frac{d^{(\alpha)}_{\min}}{2T}\}$;
6: for $i \leftarrow 1$ to $M$ do
7:     if $v_i \geq v_d$ then
8:         // For the sensors with velocities no less than $v_d$
9:         set $v'_i = \alpha^{\lfloor log_\alpha(v_i/v_d) \rfloor} \times v_d$;
10:    else
11:        //For the sensors with velocities less than $v_d$
12:        $v'_i = 0$
13:    end if
14: end for
15: set $\mathcal{V}' = \{v'_i | 1 \leq i \leq M\}$
16: set $K = \lfloor log_\alpha(v_{\max}/v_d) \rfloor + 2$
17: initial an entry array $Lc(j)$ for $1 \leq j \leq K$
18: Run Algorithm 1 on sensors $\mathcal{S}$ with velocities $\mathcal{V}'$ and get their locations $Lc$
19: adjust the coverage of sensors $\mathcal{S}$ to avoid overlapping, the union set of PoI covered
    is $\mathcal{P}_S$
20: return $\mathcal{P}_S$
```

In the rounding step, we round $v_i \in \mathcal{V}$ to $v'_i \in \mathcal{V}'$ as follows. Let $v_{\max}$ and $v_{\min}$ be the largest and smallest non-zero velocities in $\mathcal{V}$, $v_d = max\left\{\frac{d^{(\alpha)}_{\min}}{2T}, v_{\min}\right\}$, where integer $\alpha \geq 2$ and $d^{(\alpha)}_{\min}$ is the minimum distance among every segment of $\alpha + 1$ PoIs, i.e., $d^{(\alpha)}_{\min} = \min_{1 \leq j \leq N-\alpha}\{x_{j+\alpha} - x_j\}$; namely, there are no more than α PoIs on a segment with length less than $d^{(\alpha)}_{\min}$. We round a group of velocities $v_i \in \mathcal{V}$ in some interval to the same velocity v'_i by applying the following mapping:

$$\begin{cases} v'_i = 0 & \text{for } v_i < v_d, \\ v'_i = \alpha^j v_d & \text{for } \alpha^j v_d \leq v_i < \alpha^{j+1} v_d, j = 0, 1, 2, \dots, \lfloor \log_\alpha(v_{\max}/v_d) \rfloor. \end{cases} \tag{4}$$

Clearly, the above mapping rounds all $v_i \in [\alpha^j v_d, \alpha^{j+1} v_d)$ to $v'_i = \alpha^j v_d$ for $0 \leq j \leq \lfloor \log_\alpha(v_{\max}/v_d) \rfloor$, and all $v_i \in [0, v_d)$ to $v'_i = 0$. This effectively reduces the number of velocities from M to $K = \lfloor \log_\alpha(v_{\max}/v_d) \rfloor + 2 \ll M$. By setting the rounding parameter α, we can achieve any desired value of K accordingly so as to obtain the algorithm with desired approximation ratio and time complexity.

Now, we prove the performance guarantee of Algorithm 2.

Theorem 5. *Algorithm 2 is a $\frac{1}{2\alpha}$-approximation algorithm for MPSCL, where integer $\alpha \geq 2$.*

Proof. We show that the summed weight of the output of Algorithm 2, $A_2(S)$, is $\frac{1}{\alpha}$ of that of Algorithm 1, $A_1(S)$, which is the optimal separation strategy for sensors $\mathcal{S}$. Then, according to Theorem 3, that Algorithm 2 is a $\frac{1}{2\alpha}$-approximation algorithm for MPSCL.

For two sets of sensors, $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$ and $\mathcal{S}'$ with velocities $\mathcal{V}' = \{v'_1, v'_2, \ldots, v'_M\}$ following the mapping according to Equation (4). We need to show that $A_1(S)$ can be covered by α copies of sensors $\mathcal{S}'$; hence, one copy of sensor $\mathcal{S}'$ can sweep cover PoIs with weight more than $A_1(S)/\alpha$, i.e., $A_1(S') \geq \frac{1}{\alpha} A_1(S)$.

For sensor $s_i \in \mathcal{S}$ in velocity range $\alpha^j v_d \leq v_i < \alpha^{j+1} v_d$ is mapped to sensor $s'_i \in \mathcal{S}'$ with velocity $v'_i = \alpha^j v_d$ according to Equation (4). $v_i \in [v'_i, \alpha v'_i)$; thus, the coverage of s_i, $R(s_i)$ can be covered by deploying α copies of sensors s'_i with velocities v'_i. For sensor $s_i \in \mathcal{S}$ with velocity $v_i < v_d$ is mapped to sensor $s'_i \in \mathcal{S}'$ with velocity $v'_i = 0$. Note that $v_d = max\left\{\frac{1}{2T} d^{(\alpha)}_{\min}, v_{\min}\right\}$, the coverage $R(s_i)$ includes no more than α PoIs; thus, it can be covered by α copies of the sensor with velocity 0. According to the Pigeonhole principle, at least one section of $R(s_i)$ with summed weight not less than $w(s_i)/\alpha$ is covered by one sensor s'_i, where $\omega(s_i)$ is the sum of weight of PoIs on range $R(s_i)$. Assigning this sensor s'_i for $R(s_i)$ yields the approximation ratio $\frac{1}{\alpha}$ for Algorithm 1, and hence $\frac{1}{2\alpha}$-approximation for MPSCL by Theorem 3. The shift to avoid overlapping in the last past of Algorithm 2 would not reduce the performance guarantee. So, Algorithm 2 is $\frac{1}{2\alpha}$-approximation for MPSCL. □

Since Algorithm 2 calls Algorithm 1, its time complexity is $O(N * (M/K+1)^K)$, where $K = \lfloor log_\alpha(v_{\max}/v_d) \rfloor + 2$ and integer $\alpha \geq 2$ is a tradeoff between time complexity and performance ratio.

4.3.2. Linear-Programming Solution with Randomized Rounding

The above rounding technique gives a polynomial-time solution with approximation ratio capped by $\frac{1}{4}$ (when $\alpha = 2$). In this subsection, we apply linear programming relaxation and randomized rounding technique to improve the approximation ratio to $\frac{1}{2}(1 - 1/e) \approx 0.31606$.

We use the following integer program (IP) to formulate the optimal separation strategy of MPSCL and get its $(1 - 1/e)$-approximation algorithm, thus achieving $\frac{1}{2}(1 - 1/e)$-approximation of the optimal solution to MPSCL according to Theorem 3. In the integer program, variable z_l indicates whether the PoI l is covered (1 or 0). ω_l is the weight of the PoI l. $\mathcal{P}_{ij}$ indicates the set of PoIs covered by sensor s_i when it sweep covers the coordinate interval $[x_j, x_j + v_i T/2]$, where x_j is the coordinate of PoI j. Variable y_{ij} indicates whether the set $\mathcal{P}_{ij}$ is covered; namely, $y_i j = 1$ indicates that the sensor i is deployed to the coordinate interval $[x_j, x_j + v_i T/2]$. Constraint (5) means that, for each PoI l covered $(z_l = 1)$, at least one set $\mathcal{P}_{ij}$ of PoIs containing the PoI l must be selected, and no set containing the PoI l is included otherwise $(z_l = 0)$. Constraint (6) means one sensor can be used only once.

$$\begin{aligned}
& max \sum_{l=1}^{n} \omega_l z_l && \\
s.t.\quad & \sum_{(i,j):l\in\mathcal{P}_{ij}} y_{ij} \geq z_l && \forall l \in [1,N], \qquad (5)\\
& \sum_j y_{ij} \leq 1 && \forall i \in [1,M] \qquad (6)\\
& y_{ij} \in \{0,1\} && \forall i \in [1,M], j \in [1,N]\\
& z_l \in \{0,1\} && \forall l \in [1,N].
\end{aligned}$$

Replacing the integer constraints $y_{ij} \in \{0,1\}$ and $z_l \in \{0,1\}$ with $0 \leq y_{ij} \leq 1$ and $z_l \leq 1$ relaxes the above integer program (IP) to the following linear program (LP):

$$\begin{aligned}
& max \sum_{l=1}^{n} \omega_l z_l && \\
s.t.\quad & \sum_{(i,j):l\in\mathcal{P}_{ij}} y_{ij} \geq z_l && \forall l \in [1,N]\\
& \sum_j y_{ij} \leq 1 && \forall i \in [1,M]\\
& 0 \leq y_{ij} \leq 1 && \forall i \in [1,M], j \in [1,N]\\
& z_l \leq 1 && \forall l \in [1,N].
\end{aligned}$$

Let (y^*, z^*) be the optimal solution to the linear program. We apply randomized rounding to make sensor i to cover the set $\mathcal{P}_{ij}$ with probability y^*_{ij} independently, i.e., we set $y_{ij} = 1$ with probability y^*_{ij}. That yields a randomized $(1 - 1/e)$-approximation algorithm for the optimal separation strategy of MPSCL, shown in Algorithm 3. We prove its performance ratio in Theorem 6. Then, using the method of conditional expectations to derandomize Algorithm 3, we can obtain a deterministic approximation algorithm, Algorithm 4, with the same performance ratio. According to Theorem 3, Algorithm 3 and Algorithm 4 are the randomized $\frac{1}{2}(1 - 1/e)$-approximation and the deterministic $\frac{1}{2}(1 - 1/e)$-approximation algorithm for MPSCL.

Algorithm 3 MPSCL-Random

```
Input: A set of sensors $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$, A set of PoI $\mathcal{P}$ with locations
    $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and weight $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_N\}$, sweep period $T$.
Output: The sweep covered set $\mathcal{P}_S$ of PoIs.
 1: for $i \leftarrow 1$ to $M$ do
 2:   for $j \leftarrow 1$ to $N$ do
 3:     Let $\mathcal{P}_{ij}$ be the set of PoIs located in coordinate interval $[x_j, x_j + v_i T/2]$;
 4:   end for
 5: end for
 6: compute an optimal solution $(y^*, z^*)$ to the linear programming relaxation (LP);
 7: for $i \leftarrow 1$ to $M$ do
 8:   Make sensor $s_i$ cover from the PoI $j$ independently with probability $y^*_{ij}$, the set of
      covered PoIs is $\mathcal{P}_{s_i}$;
 9: end for
10: $\mathcal{P}_S = \bigcup_{i=1}^{M} P_{s_i}$;
11: return $\mathcal{P}_S$
```

Theorem 6. *Algorithm 3 is a randomized $\frac{1}{2}$(1-1/e)-approximation algorithm for MPSCL.*

Proof. The proof is similar to the proof of Theorem 5.10 in Reference [32]. In Algorithm 3, the fractional value y^*_{ij} is interpreted as the probability that $\mathcal{P}_{ij}$ is chosen. Let random variable $Z_l = 1$ if the PoI l is covered $(1 \leq l \leq N)$; $Z_l = 0$ otherwise. Then, the probability that the PoI l is not covered is the probability that all the sets including the PoI l are not chosen:

$$
\begin{aligned}
Pr[Z_l = 0] &= \prod_{(i,j):l\in\mathcal{P}_{ij}} (1 - y^*_{ij}) \\
&\le [\frac{1}{n_l} \sum_{(i,j):l\in\mathcal{P}_{ij}} (1 - y^*_{ij})]^{n_l} \\
&= (1 - \frac{\sum_{(i,j):l\in\mathcal{P}_{ij}} y^*_{ij}}{n_l})^{n_l} \\
&\le (1 - \frac{z^*_l}{n_l})^{n_l},
\end{aligned}
$$

where n_l indicates the number of sets in which the lth PoI is included, the second inequality follows from Arithmetic-geometric mean inequality, and the last inequality follows from Constraint (5).

When $k \ge 1$, the function $f_k(x) = 1 - (1 - \frac{x}{k})^k$ $(0 \le x \le 1)$ is concave. So, the probability that the PoI l is covered is

$$
\begin{aligned}
Pr[Z_l = 1] &\ge 1 - (1 - \frac{z^*_l}{n_l})^{n_l} \\
&\ge (f_{n_l}(1) - f_{n_l}(0)) \times z^*_l + f_{n_l}(0) \\
&= [1 - (1 - \frac{1}{n_l})^{n_l}] z^*_l.
\end{aligned}
$$

Let W be a random variable of the summed weight of the covered PoIs, and let OPT_{LP} and OPT_{IP} be the optimal value of the linear program (LP) and the integer program (IP), respectively. The expected value of the summed weight is:

$$
\begin{aligned}
E[W] &= \sum_{l=1}^{n} \omega_l E[Z_l] \\
&= \sum_{l=1}^{n} \omega_l Pr(Z_l = 1) \\
&\ge \sum_{l=1}^{n} \omega_l z^*_l [1 - (1 - \frac{1}{n_l})^{n_l}] \\
&\ge min_{k\ge 1}[1 - (1 - \frac{1}{k})^k] \sum_{l=1}^{n} \omega_l z^*_l \\
&\ge (1 - \frac{1}{e}) OPT_{LP} \\
&\ge (1 - \frac{1}{e}) OPT_{IP}.
\end{aligned}
$$

In Algorithm 3, there is only one linear program needed to solve, so it is a polynomial-time algorithm. Now, we have proven that Algorithm 3 is a randomized $(1 - 1/e)$-approximation algorithm for the optimal separation strategy of MPSCL. According to Theorem 3, Algorithm 3 is a randomized $\frac{1}{2}(1 - 1/e)$-approximation algorithm for MPSCL. □

Now, we show how to use the method of conditional expectations to derandomize Algorithm 3 to obtain Algorithm 4. In Algorithm 4, let random variable $Y_{ij} = 1$ denote that sensor i covers the set $\mathcal{P}_{ij}$, i.e., for sensor s_i, $y_{ij} = 1, y_{i\bar{j}} = 0$, where $\bar{j} = \{j' \in [1, N], j' \ne j\}$ and $y_{i\bar{j}} = 0$ denote that $y_{ij'} = 0, \forall j' \in \bar{j}$. Then, $Pr[Y_{ij} = 1] = y^*_{ij}$. In hth interation, $\mathcal{Y}_{h-1} = \{y_{ij} | i \in [1, h-1], j \in [1, N]\}$ is fixed. Set $y_{hj_h} = 1$ to let the current conditional expectation maximized, i.e., $j_h = argmax_{j\in[1,N]} E[W | Y_{hj} = 1; \mathcal{Y}_{h-1}]$. After M iterations, all y_{ij} for $i \in [1, M]$, $j \in [1, N]$ are set. We can get a deterministic solution with the same approximation ratio to that of Algorithm 3. The output in Algorithm 4 is the union of the sets $\mathcal{P}_{ij_i}$ for $1 \leq i \leq M$.

Algorithm 4 MPSCL-Derandomized

```
Input: A set of sensors $\mathcal{S}$ with velocities $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$, A set of PoI $\mathcal{P}$ with locations
    $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and weight $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_N\}$, sweep period $T$.
Output: The sweep covered set $\mathcal{P}_S$ of PoIs.
 1: for $i \leftarrow 1$ to $M$ do
 2:     for $j \leftarrow 1$ to $N$ do
 3:         Let $\mathcal{P}_{ij}$ be the set of PoIs located in coordinate interval $[x_j, x_j + v_i T/2]$;
 4:     end for
 5: end for
 6: compute an optimal solution $(y^*, z^*)$ to the linear programming relaxation (LP).
 7: for $i \leftarrow 1$ to $M$ do
 8:     set $j_i = argmax_{j \in [1,N]} E(W|Y_{ij} = 1; \mathcal{Y}_{i-1})$;
 9:     make sensor $s_i$ to cover the set $\mathcal{P}_{ij_i}$ of PoIs;
10: end for
11: $\mathcal{P}_S = \bigcup_{i=1}^{M} P_{ij_i}$;
12: return $\mathcal{P}_S$
```

Theorem 7. *Algorithm 4 is a deterministic* $\frac{1}{2}(1 - 1/e)$*-approximation algorithm.*

Proof. As the explanation above, we prove the theorem by induction. Without loss of generality, we assuming all the sensors will be occupied, i.e., $\sum_{j \in [1,N]} Pr[Y_{ij} = 1] = \sum_{j \in [1,N]} y_{ij} = 1$ for $i \in [1, M]$.

In the first step, we choose $j_1 = argmax_j\{E[W|Y_{1j} = 1] | 1 \leq j \leq N\}$ and set $y_{1j_1} = 1$, $y_{1\bar{j}_1} = 0$. By the definition of conditional expectations,

$$E[W] = \sum_{j=1}^{N} E[W|Y_{1j} = 1] Pr[Y_{1j} = 1]$$

and $\sum_{j=1}^{N} Pr[Y_{1j} = 1] = \sum_{j=1}^{N} y_{1j} = 1$, then

$$E[W|Y_{1j_1} = 1] = max_j\{E[W|Y_{1j} = 1]\} \geq E[W].$$

We assume that $E[W|\mathcal{Y}_h] \geq E[W]$. In the $(h+1)$th step $(1 \leq h < M)$, we choose $j_{h+1} = argmax_j\{E[W|Y_{h+1,j} = 1; \mathcal{Y}_h]\}$ and set $y_{h+1,j_{h+1}} = 1$, $y_{h+1,\bar{j}_{h+1}} = 0$. By the definition of conditional expectations ,

$$E[W|\mathcal{Y}_h] = \sum_{j=1}^{N} E[W|Y_{h+1,j} = 1; \mathcal{Y}_h] \times Pr[Y_{h+1,j} = 1]$$

and $\sum_{j=1}^{N} Pr[Y_{h+1,j} = 1] = \sum_{j=1}^{N} y_{h+1,j} = 1$, then

$$\begin{aligned} & E[W|Y_{h+1,j_{h+1}} = 1; \mathcal{Y}_h] \\ & = max_j\{E[W|Y_{h+1,j} = 1; \mathcal{Y}_h]\} \\ & \geq E[W|\mathcal{Y}_h] \\ & \geq E[W]. \end{aligned}$$

After M iterations, all $\{y_{ij} \mid i \in [1, M], j \in [1, N]\}$ are set. We get a deterministic solution W_d satisfying

$$W_d \geq E[W] \geq (1 - 1/e) OPT_{IP}.$$

Since it takes polynomial time to solve the linear programming formulation, and there are $M \times N$ linear programming formulations need to be solved, Algorithm 4 runs in polynomial time.

According to Theorem 3, Algorithm 4 is a deterministic $\frac{1}{2}(1 - 1/e)$-approximation algorithm. $\square$

5. Simulation Experiments

In this section, simulation experiments are conducted by using MATLAB to compare the algorithms including MPSCL-Velocity-Rounding, MPSCL-Random, and MPSCL-Derandomized. Since all the studies so far have not given the optimal cooperation strategy, we discuss the approximation ratios of the above algorithms to the optimal separation strategy, which are proven in the above theorems. Theorems 5–7 have shown that the approximation ratios of the three algorithms to the optimal separation strategy are separately $\frac{1}{\alpha}$, $1 - 1/e$, and $1 - 1/e$. The optimal separation strategy can be obtained by solving IP with the toolbox "Yalmip" of MATLAB, which is a free optimization solution tool developed by Lofberg.

In the experiments, there are N PoIs randomly distributed on a line of length 500, in which sweep period is 1. The velocities of M sensors are uniformly, randomly generated at range $[v_{\min}, v_{\max})$. Set the parameter $\alpha = 2$. We let N vary from 200 to 1000, M vary from 5 to 30, $v_{\min}$ vary from 5 to $v_{\max}$, and $v_{\max}$ vary from 10 to 50. For each combination of network parameters, we randomly generate ten instances to obtain the average performance and the lower bound of the performance of each algorithm. In the experiment, none of the parameters showed a significant effect on the approximation ratio. As shown in Table 2, the performance lower bounds of the three algorithms all satisfy the theoretical analysis. Algorithm MPSCL-Random is a randomized approximation algorithm. It only needs to satisfy the algorithm performance at a high probability. Thus, it has a fluctuating performance, as shown in Table 2. However, it also shows better performance than Algorithm MPSCL-Velocity-Rounding. Algorithm MPSCL-Derandomized derandomized Algorithm MPSCL-Random, so it must have higher computational complexity and better performance than MPSCL-Random, as shown in Table 2.

Table 2. The experimental performance of the algorithms.

Name of the Algorithm	Lower Bound of the Performance	Average Performance
MPSCL-Velocity-Rounding	0.66	0.81
MPSCL-Random	0.66	0.92
MPSCL-Derandomized	0.89	0.98

The influence of parameters on running time is shown below. Figure 3 shows the influence of the minimum velocity $v_{\min}$. In this experiment, the number of targets is $N = 1000$, the number of sensors is $M = 20$, the maximum velocity is fixed, $v_{\max} = 50$, and the minimum velocity varies from 5 to 30. When the minimum velocity increases, the running time of Algorithm MPSCL-Velocity-Rounding decreases. Remind that the computational complexity of Algorithm MPSCL-Random is $O(N \times (M/K + 1)^K)$, where $K = \lfloor log_\alpha(v_{\max}/v_d) \rfloor + 2$ and $v_d = max\left\{\frac{1}{2T} d_{\min}^{(\alpha)}, v_{\min}\right\}$. When $\alpha = 2$, the value of K is proportional to $v_{\max}/v_{\min}$. Therefore, when $v_{\min}$ increases, K decreases, which has a great impact on the complexity of Algorithm MPSCL-Velocity-Rounding. Thus, the running time becomes shorter. On the other hand, when the minimum velocity increases, the running time of Algorithm MPSCL-Random and MPSCL-Derandomized increased slightly. That is because, when $v_{\min}$ increases, so does the average velocity of the sensors, the coverage range of a sensor becomes larger, i.e., $|P_{ij}|$ in formula LP becomes larger, which makes the running time of Algorithm MPSCL-Random and MPSCL-Derandomized slightly increased. This can be shown more clearly in Figure 4. In Figure 4a,b, $K = 2$ or $K = 3$ is fixed, respectively, where K is the number of velocity groups. Let $N = 1000$, $M = 10$, $v_{\min} = 3{:}3{:}15$. It is shown that the running time of Algorithm MPSCL-Velocity-Rounding would not be affected too much, and that of Algorithm MPSCL-Random, Algorithm MPSCL-Derandomzied would increase as $v_{\min}$ increases.

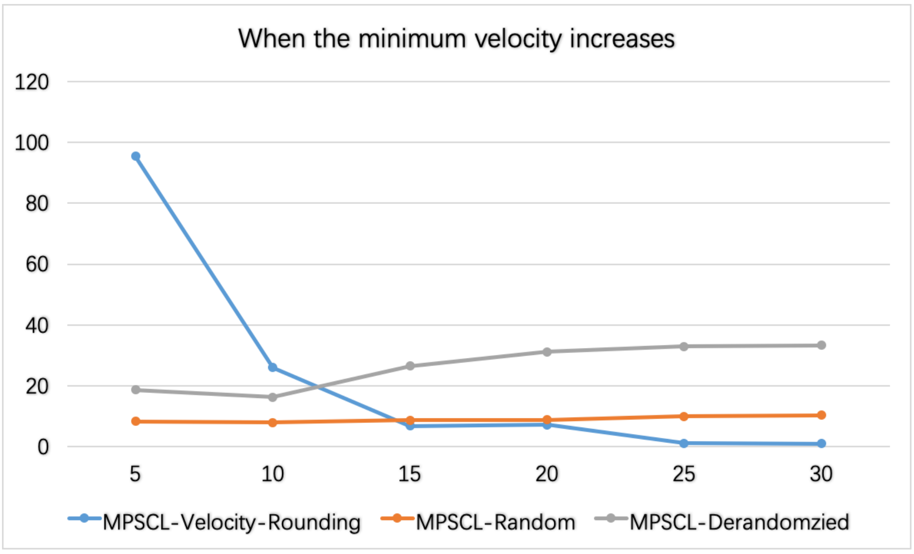


Figure 3. When the number of sensors remains the same, the influence of the minimum velocity of the sensors on running time. ($N = 1000, M = 20, v_{\max} = 50$.)

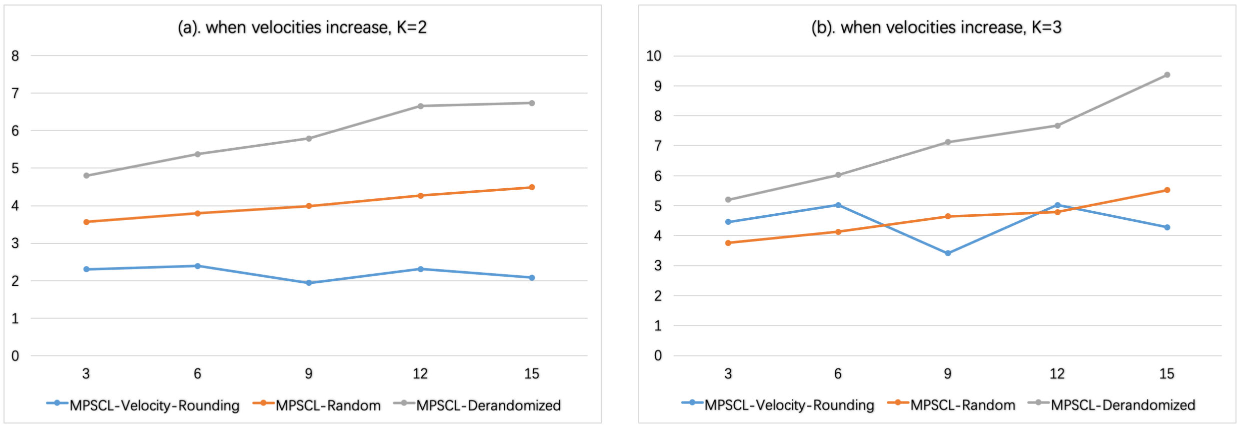
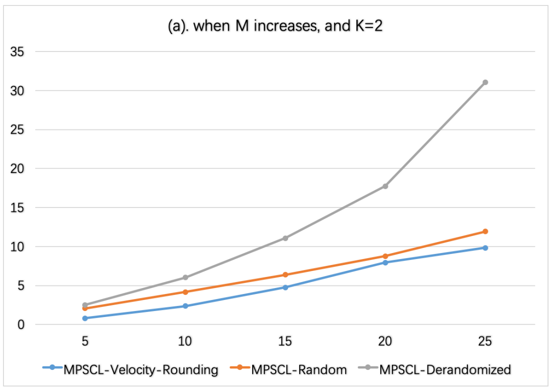


Figure 4. The influence of the velocities $[v_{\min}, v_{\max})$ on running time. ($N = 1000, M = 10$) (**a**). when $K = log_2(v_{\max}/v_{\min}) = 2$. (**b**). when $K = log_2(v_{\max}/v_{\min}) = 3$.

In Figure 5, set $v_{\min} = 10$, $v_{\max} = 40$, then the velocities of sensors can be divided into $K = 2$ groups, $[10, 20)$, $[20, 40)$. Recall that the length of the line is 500. In Figure 5a, $N = 1000$, the number of sensors M varies from 5 to 25. In Figure 5b, $M = 20$, the number of targets N varies from 500 to 2500. Figure 5 shows that when $K \le 2$, i.e., $v_{\max}/v_{\min} \le 4$, the running time of Algorithm MPSCL-Velocity-Rounding is the shortest. That is reasonable because there is still no algorithm for solving linear programming with computational complexity less than $O(n^{2+\epsilon})$ (https://en.wikipedia.org/wiki/Linear_programming (accessed on 10 February 2021)).

Where n is the number of variables of linear programming, $\epsilon > 0$ is a fraction.



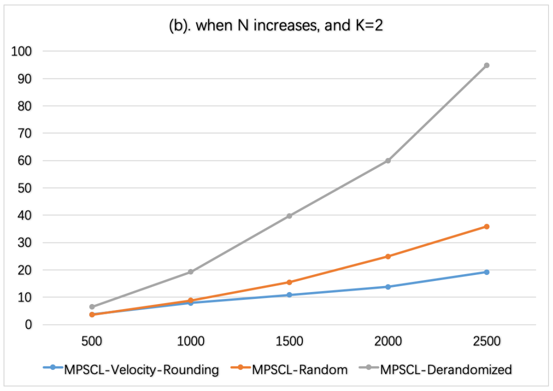
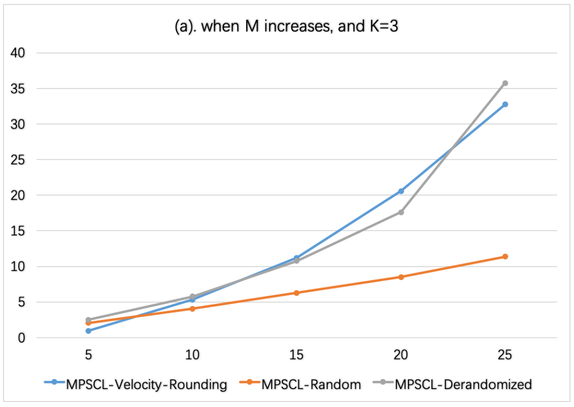


Figure 5. When $K = 2$ ($v_{\min} = 10$, $v_{\max} = 40$). (**a**) The influence of the number of sensors M on running time. (**b**) The influence of the number of targets N on running time.

In Figure 6, set $v_{\min} = 5$, $v_{\max} = 40$; similarly, the velocities are divided into $K = 3$ groups, $[5, 10)$, $[10, 20)$, $[20, 40)$. In Figure 6a, $N = 1000$, the number of sensors M varies from 5 to 25. In Figure 6b, $M = 20$, the number of targets N varies from 500 to 2500. There is the same parameter configuration except $v_{\min} = 3$ and $v_{\max} = 48$ in Figure 7, i.e., $K = 4$. As shown in Figures 6 and 7, when $K \geq 3$, the running time of Algorithm MPSCL-Random is shorter than Algorithm MPSCL-Velocity-Rounding. And as K or M increases, the running time of Algorithm MPSCL-Velocity-Rounding increases rapidly.



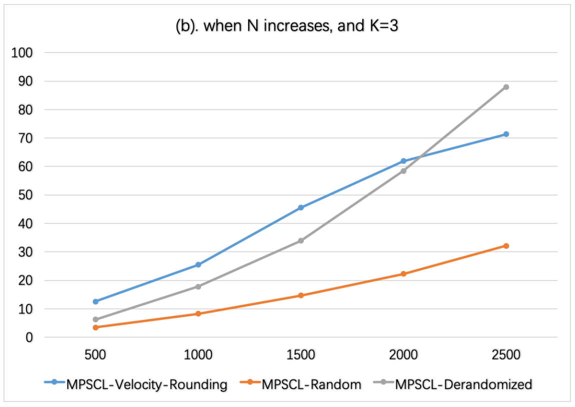
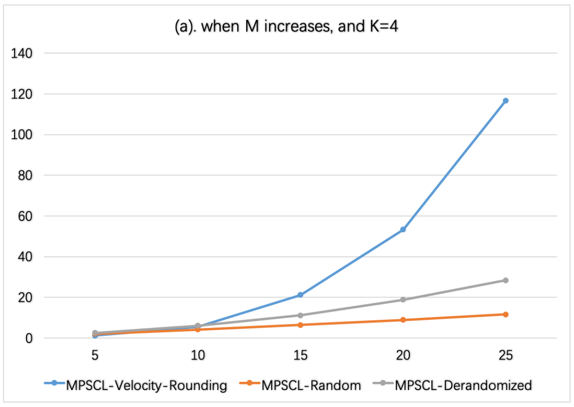


Figure 6. When $K = 3$ ($v_{\min} = 5$, $v_{\max} = 40$). (**a**) The influence of the number of sensors M on running time. (**b**) The influence of the number of targets N on running time.



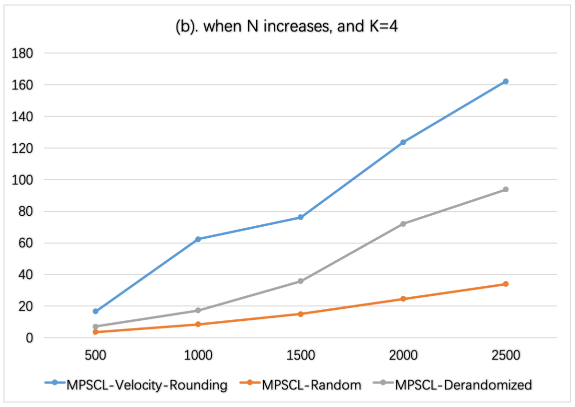


Figure 7. When $K = 4$ ($v_{\min} = 3$, $v_{\max} = 48$). (**a**) The influence of the number of sensors M on running time. (**b**) The influence of the number of targets N on running time.

According to the experimental results, the performance and the computational complexity of the three algorithms all satisfy the theoretical analysis. Algorithm MPSCL-Random (Algorithm 3) shows excellent average performance. When $v_{\max} > 4v_{\min}$, it would be a good choice if shortest running time is required. Or, if higher algorithm performance is asked for, Algorithm MPSCL-Derandomized (Algorithm 4) is better. When $v_{\max} \leq 4v_{\min}$, if the algorithm performance is acceptable, Algorithm MPSCL-Velocity-Rounding (Algorithm 2) can obtain faster results.

6. Conclusions

In this paper, we are the first to prove the PSCL problem is NPC by reducing 3-Partition problem to it and provide optimal and approximation algorithms for the MPSCL problem in different cases. For the special case when the velocities of the sensors are the same, we propose an optimal algorithm with a computational complexity of $O(MN)$. For the case when the sensors have a constant number of different velocities, we use the dynamic programming method to find the optimal separation strategy and prove that it is a $\frac{1}{2}$ approximation algorithm for MPSCL. For the general case when the sensors have arbitrary velocities, we propose three approximation algorithms: one uses dynamic programming after velocity rounding to get an approximation ratio $\frac{1}{2\alpha}$; the second is a random approximation algorithm with an expected approximation ratio $\frac{1}{2}(1 - 1/e)$; and the third one derandomizes the random algorithm to get a deterministic algorithm with the same approximation ratio to the second one. All theoretical analyses are verified in experiments. Our future work is to study the Max-weighted Point Sweep Coverage problem in other types of graphs, such as trees and Eulerian graphs, and reduce the approximation ratio between the optimal value of MPSCL and that of the optimal separation strategy. Min-Sensor Sweep Coverage problem on Lines is also an interesting topic.

Author Contributions: Conceptualization, D.L. and H.S.; Methodology, D.L.; Software, D.L.; Validation, D.L., H.S.; Investigation, D.L.; Resources, H.S.; Writing—original draft preparation, D.L.; Writing—review, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by Key-Area Research and Development of Guangdong Province (No. 2020B010164003), The corresponding author is Hong Shen.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, of in the decision to publish the results.

References

1. Nguyen, N.T.; Liu, B.H.; Wang, S.Y. On new approaches of maximum weighted target coverage and sensor connectivity: Hardness and approximation. *IEEE Trans. Netw. Sci. Eng.* **2019**, *7*, 1736–1751. [CrossRef]
2. Liang, D.; Shen, H.; Chen, L. Maximum Target Coverage Problem in Mobile Wireless Sensor Networks. *Sensors* **2021**, *21*, 184. [CrossRef] [PubMed]
3. Khalesian, M.; Delavar, M.R. Wireless sensors deployment optimization using a constrained Pareto-based multi-objective evolutionary approach. *Eng. Appl. Artif. Intell.* **2016**, *53*, 126–139. [CrossRef]
4. Kong, L.; Zhao, M.; Liu, X.Y.; Lu, J.; Liu, Y.; Wu, M.Y.; Shu, W. Surface coverage in sensor networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 234–243. [CrossRef]
5. Li, S.; Shen, H. Minimizing the maximum sensor movement for barrier coverage in the plane. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 244–252.
6. Kim, D.; Wang, W.; Son, J.; Wu, W.; Lee, W.; Tokuta, A.O. Maximum lifetime combined barrier-coverage of weak static sensors and strong mobile sensors. *IEEE Trans. Mob. Comput.* **2016**, *16*, 1956–1966. [CrossRef]
7. Guo, J.; Jafarkhani, H. Movement-Efficient Sensor Deployment in Wireless Sensor Networks With Limited Communication Range. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 3469–3484. [CrossRef]
8. Gao, X.; Chen, Z.; Wu, F.; Chen, G. Energy Efficient Algorithms for k -Sink Minimum Movement Target Coverage Problem in Mobile Sensor Network. *IEEE/ACM Trans. Netw.* **2017**, *25*, 3616–3627. [CrossRef]
9. Elhoseny, M.; Tharwat, A.; Yuan, X.; Hassanien, A.E. Optimizing K-coverage of mobile WSNs. *Expert Syst. Appl.* **2018**, *92*, 142–153. [CrossRef]

10. Cinque, M.; Cotroneo, D.; Di Martino, C.; Russo, S.; Testa, A. Avr-inject: A tool for injecting faults in wireless sensor nodes. In Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, Italy, 23–29 May 2009; pp. 1–8.
11. Cheng, W.; Li, M.; Liu, K.; Liu, Y.; Li, X.; Liao, X. Sweep coverage with mobile sensors. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–9.
12. Yu, Z.; Zhou, J.; Guo, W.; Guo, L.; Yu, Z. Participant selection for t-sweep k-coverage crowd sensing tasks. *World Wide Web* **2017**, *21*, 741–758. [CrossRef]
13. Huang, P.; Lin, F.; Liu, C.; Gao, J.; Zhou, J.l. ACO-Based Sweep Coverage Scheme in Wireless Sensor Networks. *J. Sens.* **2015**, *2015*, 484902. [CrossRef]
14. Li, J.; Xiong, Y.H.; She, J.H.; Wu, M. A Path Planning Method for Sweep Coverage With Multiple UAVs. *IEEE Internet Things J.* **2020**, *7*, 8967–8978. [CrossRef]
15. Wu, L.G.; Xiong, Y.H.; Wu, M.; He, Y.; She, J.H. A Task Assignment Method for Sweep Coverage Optimization Based on Crowdsensing. *IEEE Internet Things J.* **2019**, *6*, 10686–10699. [CrossRef]
16. Gorain, B.; Mandal, P.S. Approximation algorithm for sweep coverage on graph. *Inf. Process. Lett.* **2015**, *115*, 712–718. [CrossRef]
17. Du, J.; Li, Y.; Liu, H.; Sha, K. On sweep coverage with minimum mobile sensors. In Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems, Shanghai, China, 8–10 December 2010; pp. 283–290.
18. Gorain, B.; Mandal, P.S. Approximation algorithms for sweep coverage in wireless sensor networks. *J. Parallel Distrib. Comput.* **2014**, *74*, 2699–2707. [CrossRef]
19. Gorain, B.; Mandal, P.S. Line sweep coverage in wireless sensor networks. In Proceedings of the 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 6–10 January 2014; pp. 1–6.
20. Chen, Z.; Zhu, X.; Gao, X.; Wu, F.; Gu, J.; Chen, G. Efficient Scheduling Strategies for Mobile Sensors in Sweep Coverage Problem. In Proceedings of the 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), London, UK, 27–30 June 2016; pp. 1–4.
21. Gorain, B.; Mandal, P.S. Solving energy issues for sweep coverage in wireless sensor networks. *Discret. Appl. Math.* **2017**, *228*, 130–139. [CrossRef]
22. Gorain, B.; Mandal, P.S. Approximation Algorithms for Barrier Sweep Coverage. *Int. J. Found. Comput. Sci.* **2019**, *30*, 425–448. [CrossRef]
23. Czyzowicz, J.; Gąsieniec, L.; Kosowski, A.; Kranakis, E. Boundary patrolling by mobile agents with distinct maximal speeds. In *European Symposium on Algorithms*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 701–712.
24. Dumitrescu, A.; Ghosh, A.; Tóth, C.D. On fence patrolling by mobile agents. *arXiv* **2014**, arXiv:1401.6070.
25. Kawamura, A.; Kobayashi, Y. Fence patrolling by mobile agents with distinct speeds. *Distrib. Comput.* **2015**, *28*, 147–154. [CrossRef]
26. Kawamura, A.; Soejima, M. Simple strategies versus optimal schedules in multi-agent patrolling. In *International Conference on Algorithms and Complexity*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 261–273.
27. Pasqualetti, F.; Franchi, A.; Bullo, F. On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Trans. Robot.* **2012**, *28*, 592–606. [CrossRef]
28. Collins, A.; Czyzowicz, J.; Gasieniec, L.; Kosowski, A.; Kranakis, E.; Krizanc, D.; Martin, R.; Morales Ponce, O. Optimal patrolling of fragmented boundaries. In Proceedings of the Twenty-Fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures, Montreal, QC, Canada, 23–25 July 2013; ACM: New York, NY, USA, 2013; pp. 241–250.
29. Gao, X.F.; Fan, J.H.; Wu, F.; Chen, G.H. Approximation Algorithms for Sweep Coverage Problem With Multiple Mobile Sensors. *IEEE-Acm Trans. Netw.* **2018**, *26*, 990–1003. [CrossRef]
30. Gao, X.; Fan, J.; Wu, F.; Chen, G. Cooperative Sweep Coverage Problem with Mobile Sensors. *IEEE Trans. Mob. Comput.* **2020**. [CrossRef]
31. Gary, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman and Company: New York, NY, USA, 1979.
32. Williamson, D.P.; Shmoys, D.B. *The Design of Approximation Algorithms*; Cambridge University Press: Cambridge, UK, 2011.