A Projected Lloyd's Algorithm for Coverage Control Problems

Yoav Palti* and Daniel Zelazo†

This work presents a new approach for a coverage problem for large areas and a limited number of sensors. We consider the problem where there are not enough sensors to provide complete coverage of the designated area at once, and therefore the sensors should coordinate their coverage in a sequential manner. Furthermore, we require that at each stage of coverage, a sub-region in the area is constantly monitored. Our solution approach is a combination of two algorithms, the first one is Lloyd's algorithm, which is well known and used in this field, and the second one is a modified Lloyd's algorithm that we term the *projected Lloyd's Algorithm*. Lastly, we combine Lloyd's algorithm (and its projected version) with a distance based formation controller. We demonstrate our results with numerical simulations.

I Introduction

The mission of monitoring some range (for example, area) using sensors is widely spread in various fields - it can be used for photographing a large area, receiving or sending electromagnetic signals, tracking targets, or even for designing an algorithm for a robotic vacuum cleaner [1–3]. This research area is known in the scientific community as the *coverage control* problem [4]. Sensor coverage can be generally described as the reflection of how well a given range is monitored by sensors. This field is well explored in the scientific community, especially in the recent years when the fields of distributed algorithms and cooperative systems were developed.

A concept that repeats in many researches is finding a set of trajectories (at least one route for every sensor), allowing the mobile sensors to achieve the required coverage goal. For example, when trying to cover an area using two mobile sensors, the algorithm will define a trajectory for each sensor such that the whole area will be covered [5, 6]. Another idea that appears in most of the relevant literature is the use of Voronoi diagrams for optimizing the sensor location [6, 7].

When coverage is required for large areas, and specifically when a single sensor cannot provide full coverage of that area, the idea of using a *deployment* of mobile sensors is presented [4, 7]. In [4], the problem is formed using a probabilistic network model and some density function, to model evens. Then, an optimization algorithm is proposed to find the optimal deployment such that maximum coverage is obtained with minimal communications cost. On [7], Cortes *et al.* propose an optimization algorithm for maximal coverage using Centroidal Voronoi Tessellations [8]. The former concept will be utilized and modified in this paper to achieve new coverage capabilities that will be further described.

In this work, we formulate the coverage problem from another angle. We firstly assume that the given range can be at most *partially* covered, however we also define some sub-range

*Faculty of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa, yoavp10@gmail.com

†Associate Professor, Faculty of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa, dzelazo@technion.ac.il

that must be at least partially covered at all times (for example, surveillance or home-base communications constraint). We present an algorithm for partitioning this range into tiles, such that each tile intersects with this sub-range constraint, and then, utilizing [7] method to achieve coverage of this tile. This method has two main benefits - first, we can achieve full coverage if we *sequentially* moving the agents from tile to tile, and second - the operator can return the agents to any tile whenever he desires to.

There are some cases where maintaining spatial properties is important. One example is when the communications between the sensors are limited to some range. Another example is when the spatial shape is critical for the mission, as in geolocation missions. A well known solution for this problem is the distance-based formation control [9, 10].

Following this chapter, the problem will be formally defined. Then, mathematical background on the main concepts of this paper will be given, followed by the solution algorithms and control strategies. Finally, numerical simulations will be shown.

II Problem Formulation

Let us consider an area $A \subset \mathbb{R}^2$ that we aim to cover with $n \in \mathbb{N}$ *mobile* sensors. Those sensors are defined as the set $S = \{s_1, \dots, s_n\}$ located at positions $P(t) = \{p_i(t) \in \mathbb{R}^2 \mid i = 1, \ \dots, \ n\}$:

- The mobile sensors are modelled using integrator dynamics $\dot{p}_i(t) = u$.
- Each sensor can cover a region described by the abstract set $C_i\left(p_i(t)\right) \subset \mathbb{R}^2$.

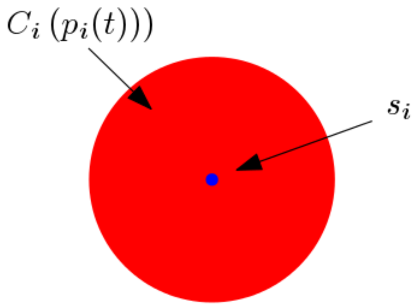


Figure 1: A sensor and its coverage region

II.A Configuration

A configuration c of the mobile sensors at time t is the stack of the sensor positions at time t:

$$c\left(t\right) = \begin{bmatrix} p_1^T\left(t\right) & \cdots & p_n^T\left(t\right) \end{bmatrix}^T \in \mathbb{R}^{2n}$$

Thus the coverage of a configuration:

$$D\left(c\left(t\right)\right) = \cap C_i(p_i(t))$$

Let us consider a sub-area of A, $A_m \subset A$ which must be partially covered (e.g. ground station).

Assumption 1. $D\left(c\left(t\right)\right) \subset A$ - *a single configuration* can't *provide full coverage!*

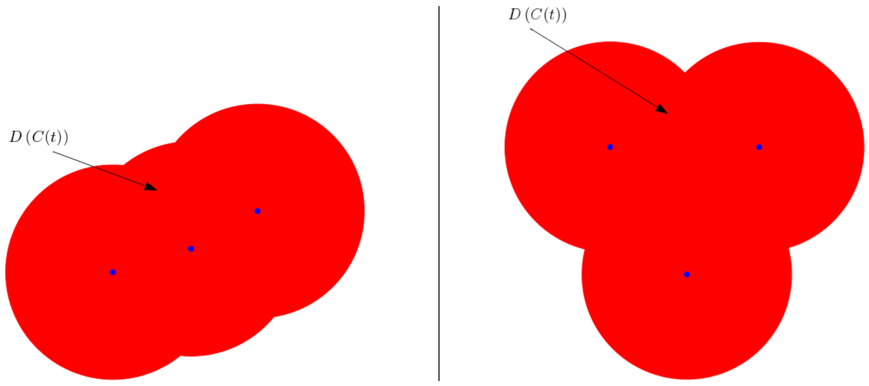


Figure 2: Two configurations built from the same sensors set

II.B Partition

The partition of $A \in \mathbb{R}^2$, $PR(A)$, is a finite set built from n subset $pr_i \subset \mathbb{R}^2,\ i = 1,\ \dots,\ n$ such that:

- $pr_i \cap pr_j = \emptyset \ \forall i \neq j$.
- $\cup pr_i = A$.

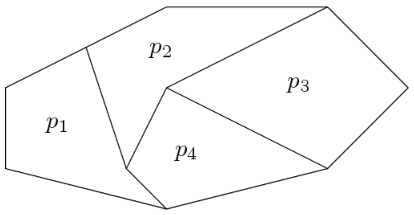


Figure 3: An example of a partition of some area

II.C Problem Definition

Problem 1. *1. Find a partition $PR(A)$ such that each subsets $pr_i \in PR(A),\ pr_i \cap A_m \neq \emptyset$*

2. Find a deployment controller, such that

$$pr_i \subseteq \lim_{t\to\infty} D(c(t)), \textit{for } i = 1, \dots, n.$$

III Voronoi Diagrams and Distributed Coverage Algorithms

In this chapter we will introduce two main concepts that were used in this work. First, Voronoi Diagrams, and later Lloyd's algorithm.

III.A Voronoi Diagrams

The basic mathematical concept that we employ in this work is the *Voronoi Diagram* (also known as Voronoi partition or Voronoi tessellation). While being a method to partition an area with some cost function, it is a widely-used in the optimal coverage problem [6–8]. The

Voronoi Diagram of a region $\Omega \subset \mathbb{R}^2$ is the set of partitions $\mathcal{V} = \{V_i \mid \cup V_i = \Omega\}$, generated by the generators $\mathcal{Z} = \{z_1, \ldots, z_n \mid z_i \in \Omega\}$, such that

$$V_i = \{q \in \Omega \mid \|q - z_i\| \leq \|q - z_j\| \, \forall z_i, z_j \in \mathcal{Z}\}, \tag{1}$$

where V_i corresponds to the i-th element of $\mathcal{Z}$, and $\|\cdot\|$ denotes the Euclidean distance.

A more intuitive and non-formal definition of the Voronoi Diagram is as follows. If we take some area and place points p_i in it, then each partition is the set of all points that are closer to p_i than p_j, when $j \neq i$. An example is the problem where there are post offices in some city, and it is needed to decide which house will be served by which specific post office. Using Voronoi partitioning, we can determine which houses are the nearest to each office and therefore conclude which houses each branch will serve.

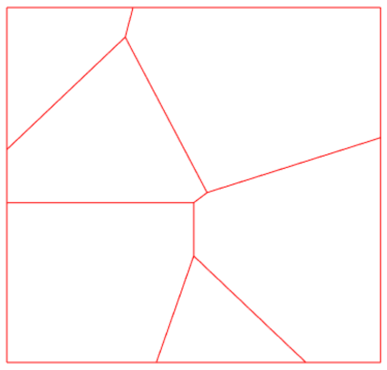
Centroidal Voronoi Tessellation

One can define a density function, ρ_i, for each Voronoi partition V_i. Then, we can define the center of mass for each partition as:

$$z_i^* = \frac{\int_{V_i} y\rho(y)dy}{\int_{V_i} \rho(y)dy} \tag{2}$$

If a generator $z_i = z_i^* \, \forall \, V_i$, we call this partitioning a *centroidal Voronoi tessellation* (CVT). Such tessellations are useful in terms of location optimization [5, 7, 8]. A Voronoi tessellation illustration can be seen in Figure 5.

While Voroni diagrams are useful for partitioning some existing range, CVT is a technique used for (but not only) for *planning* in an optimal way. If in the Voronoi diagram explanation, it was used to plan the delivery areas for each office, we will use the CVT to find where are the optimal locations to place the post offices!


(a) Voronoi Diagram.

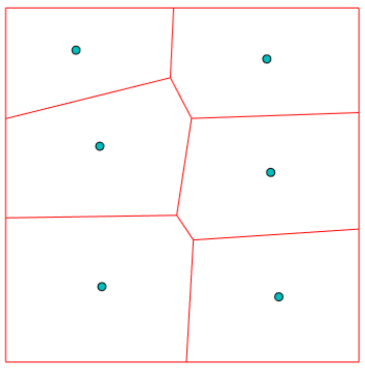

(b) Centroidal Voronoi Tessellations. The points represent the center of mass, calculated using the density function $\phi = 1$.

Figure 4: Illustration of Voronoi Diagram and a CVT, both from the same initial conditions

As can be seen in equation (1), the calculation of each Voronoi cell for some agent depends on the other agents positions. Therefore, any sort of calculation required *sharing information*. This can be done in a centralized approach (as it was done for this research), but there exists also decentralized calculations, for example [11]. Using decentralized calculation, each agent have to have the info only on their neighbours (or any limited amount of agents), which allows for calculating huge networks, in dynamic situations where agents can, for example, connect and disconnect from the network in real time.

III.B Lloyd's Algorithm

As mentioned above, the CVT's are very useful for locational optimization. However, calculating the CVT might be a complicated task. Lloyd proposed a very simple way of calculating the CVT [12], presented in Alogirthm 1.

Algorithm 1 Lloyd's Algorithm

```
1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Move the agents to the center of mass.
4: Repeat until converge.
```

Continuous-time Lloyd's Algorithm

Cortes et al. proposed a control algorithm based on Lloyd's algorithm. According to [7], let us define agent i position as p_i and the i's partition centroid as C_{V_i}. Of course, this centroid is defined for some given density function, which allow us a degree of freedom in the calculations. For some proportional constant k_{prop}, the controller can be defined as:

$$u_i = -k_{prop}\left(p_i - C_{V_i}\right) \tag{3}$$

Cortes et al. also proved that the controller (3) is locally asymptotic stable. It can be proved using the Direct Lyaponuv method, and the proof is given on [7].

IV Proposed Solution

In this chapter, problem (1) will be addressed. First, we will propose an algorithm to address the special partition requirements, and then a solution for the whole problem will be shown.

IV.A Projected Lloyd's Algorithm

A short reminder to the first part of the problem: *Partition the area A such that $pr_i \cap A_m \neq \emptyset$.*

To show the solution, we will first define a *Projection* onto the area A_m:

Definition 1.

$$\text{PROJ}_{A_m}(x) = \arg\min_{y \in A_m} \|x - y\|^2$$

Applying the projection operator in conjuction with Lloyd's algorithm yield the *Projected Lloyd's Algorithm* (PLA):

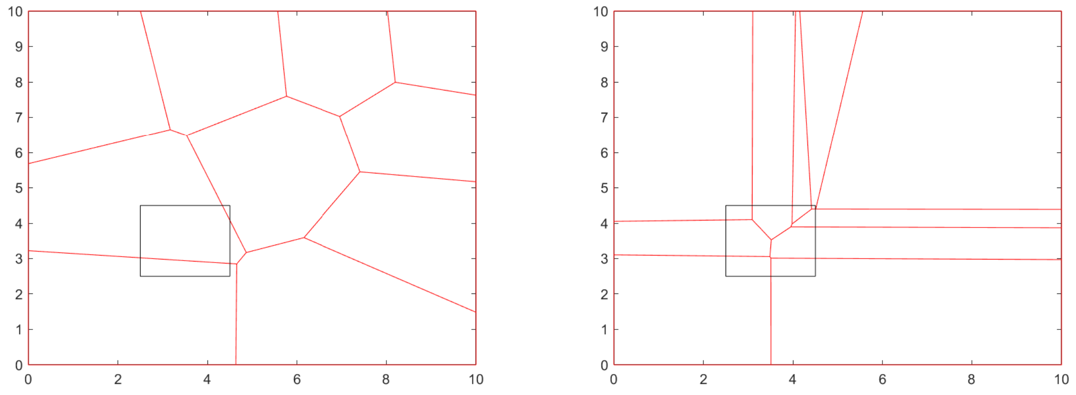
As one can see, we added two more steps over the original Lloyd's algorithm - where we project the cell center of mass to the polygon that defines the area constraint (steps number 3-4), and then moving the agents to those points. Those step allows us to ensure that at least one agent will be *on* the limiting polygon, thus we will have coverage within the constraint.

Algorithm 2 Projected Lloyd's Algorithm

```
1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Project the center of mass of every cell to the area constraint limiting polygon.
4: Move the agents the projected center of mass.
5: Repeat until converge.
```

This algorithm can be also formed in a continuous time form, based of (1):

$$u_i = -k_{prop} \left(p_i - \textit{proj} \left(C_{V_i} \right) \right) \tag{4}$$



(a) CVT calculated using Lloyd's algorithm (b) PLA solution

Figure 5: CVT and PLA solutions for initial conditions. The black box represents A_m.

Stability of the Projected Lloyd's Algorithm

Theorem 1. *The controller proposed in (4) is locally asymptotic stable.*

Proof. TODO! □

One can notice a major difference between PLA and Loyd's algorithm - while Lloyd's algorithm is an optimal calculation (that converge to some local minima), the PLA is *some* solution, and we do not provide any optimal condition.

IV.B Full Problem Solution Algorithm

The PLA is answering the question of how to partition such that $pr_i \cap A_m \neq \emptyset$. The second part of the problem, which requires a deployment controller, was answered in [7]. It is possible to utilized their solution, proposing a solution algorithm for problem (1).

Algorithm 3 Problem (1) Solution Algorithm

```
1: Using some random initial guess, calculate the PLA for the whole area.
2: For each partition (assuming that the agents can actually cover each partition with their
   coverage radius), calculate the CVT. The initial positions for the CVT calculation is the
   previous partition CVT.
```

We should notice the following things:

1. In stage 1, the number of the partition that the area will be divided to is somewhat arbitrary.
2. In stage 2, we assume that the coverage radius of the agents is big enough to cover the whole partition. If it's not the case, then we can dived the area for more partitions in stage 1. Some more advanced work can propose an algorithmic solution for this problem.
3. The Voronoi diagram can be calculated even if the agents are outside of the required area. It does require sometimes more iterations.
4. This version of the algorithm is centralized.

IV.C Results

In this sub-section, we will show results for problem (1), using algorithm 3.

In the following simulations, the initial conditions were identical.

IV.C.1 Results without *PLA*

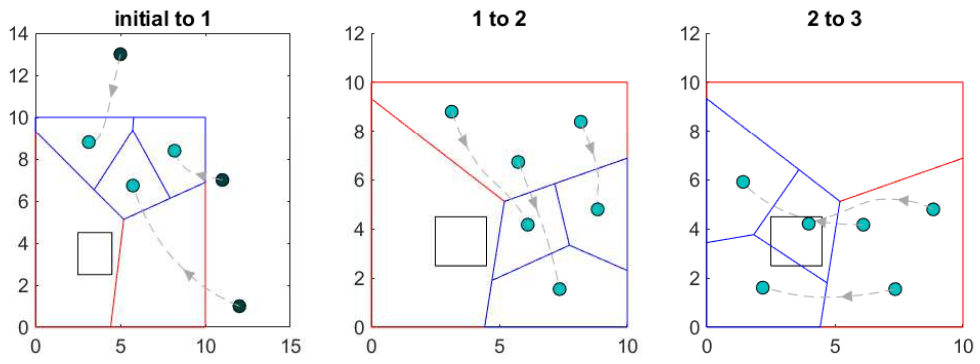


Figure 6: Results without PLA. The box represents A_m

In figure **??**, we can see a simulation ran for 3 agents. The area is partitioned into 3 tiles using Lloyd's algorithm (that is CVT), and each tile is covered using the algorithm proposed in [7]. The agents moved between the tiles in arbitrary order, and one can notice that only in the last step there was coverage of A_m.

IV.C.2 Results with *PLA*

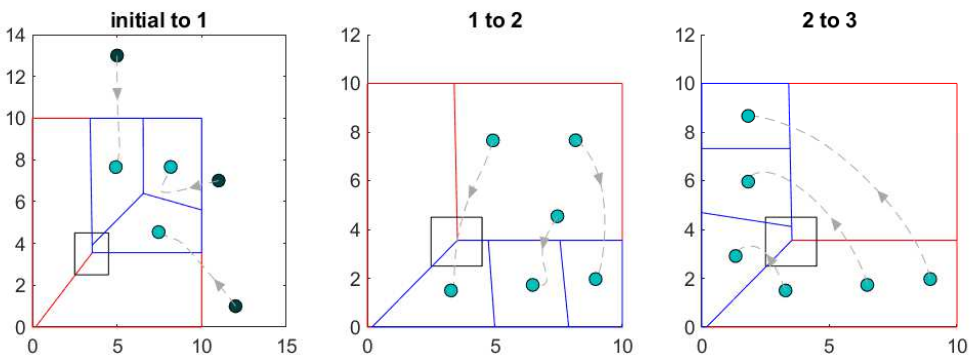


Figure 7: Results with PLA. The box represents A_m

In figure **??**, we can see a simulation ran for 3 agents. The area is partitioned into 3 tiles using Projected Lloyd's algorithm, and each tile is covered using the algorithm proposed in [7]. The agents moved between the tiles in arbitrary order, and each tile has some partial coverage of A_m.

V Combining Lloyd's Algorithm and Formation Control

VI Concluding Remarks

References

[1] Nigam, N., Bieniawski, S., Kroo, I., and Vian, J., "Control of multiple UAVs for persistent surveillance: Algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, Vol. 20, No. 5, 2012, pp. 1236–1251, doi:10.1109/TCST.2011.2167331.

[2] Palacios-Gasós, J. M., Montijano, E., Sagüés, C., and Llorente, S., "Multi-robot persistent coverage with optimal times," in "2016 IEEE 55th Conference on Decision and Control (CDC)," , 2016, pp. 3511–3517, doi:10.1109/CDC.2016.7798796.

[3] Loizou, S. G. and Constantinou, C. C., "Multi-robot coverage on dendritic topologies under communication constraints," in "2016 IEEE 55th Conference on Decision and Control (CDC)," , 2016, pp. 43–48, doi:10.1109/CDC.2016.7798244.

[4] Cassandras, C. G. and Li, W., "Sensor Networks and Cooperative Control," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pp. 4237–4238.

[5] Atinç, G. M., Stipanović, D. M., Voulgaris, P. G., and Karkoub, M., "Supervised coverage control with guaranteed collision avoidance and proximity maintenance," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3463–3468, doi:10.1109/CDC.2013.6760414.

[6] Hussein, I. I. and Stipanovic, D. M., "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Transactions on Control Systems Technology*, Vol. 15, No. 4, 2007, pp. 642–657, doi:10.1109/TCST.2007.899155.

[7] Cortes, J. and Martinez, S., “Coverage control for mobile sensing networks,” *Robotics and Automation, . . .* , Vol. 20, No. 2, 2004, p. 13, doi:10.1109/TRA.2004.824698.

[8] Du, Q., Faber, V., and Gunzburger, M., “Centroidal Voronoi Tessellations: Applications and Algorithms,” *SIAM Review*, Vol. 41, No. 4, 1999, pp. 637–676, doi:10.1137/S0036144599352836.

[9] Krick, L., Broucke, M. E., and Francis, B. A., “Stabilization of infinitesimally rigid formations of multi-robot networks,” *Proceedings of the IEEE Conference on Decision and Control*, pp. 477–482, doi:10.1109/CDC.2008.4738760.

[10] Kwang-Kyo Oh and Hyo-Sung Ahn, “Distance-based formation control using Euclidean distance dynamics matrix: General cases,” *Proceedings of the 2011 American Control Conference*, pp. 4816–4821, doi:10.1109/ACC.2011.5990609.

[11] Adams, J. S., Sun, W., and Chen, Y. Q., *Formations with decentralized Centroidal Voronoi tessellation algorithm*, Vol. 42, IFAC, 2009, doi:10.3182/20091006-3-US-4006.0021.

[12] Lloyd, S., “Least squares quantization in PCM,” *IEEE Transactions on Information Theory, Information Theory, IEEE Transactions on, IEEE Trans. Inform. Theory*, Vol. 28, No. 2, 1982, pp. 129–137, doi:10.1109/TIT.1982.1056489.