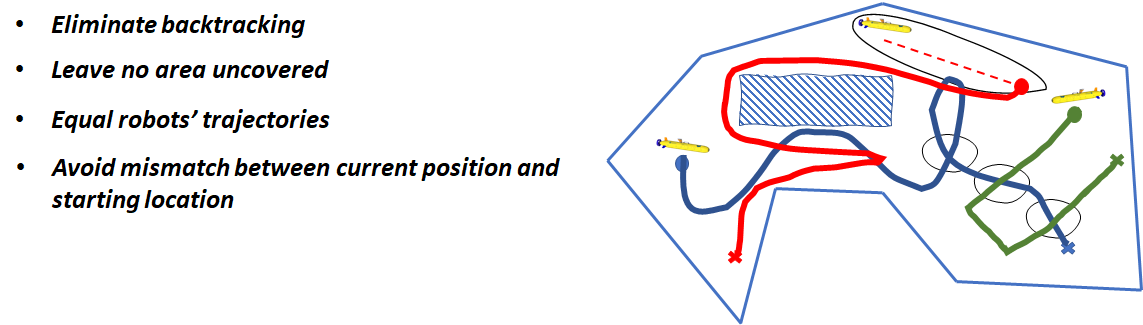
**DARP Algorithm Analysis**

**OBJECTIVE:**

The most fundamental problem in path planning is determining the path each robot would take to optimise time as well as avoid obstacles. Of course, this can even be achieved by a single robot moving randomly in the space after a (probably) long period of time. However, the limited battery capabilities of autonomous robots, in addition to the constantly increased areas that need to be covered/monitored, have motivated the deployment of several autonomous devices with advanced path planning mechanisms.

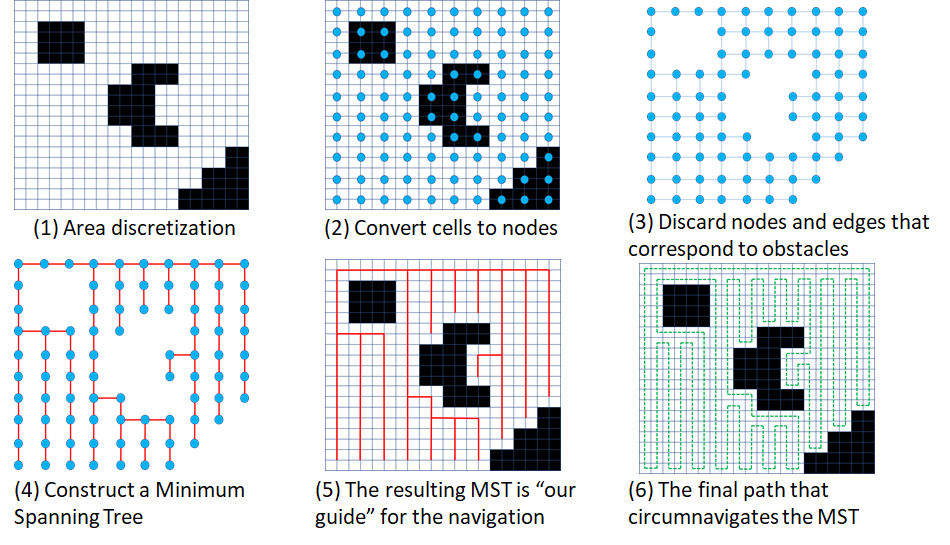
So our objective is to: Design robot paths that completely cover this area of interest in the minimum possible time called coverage path planning problem (CPP).



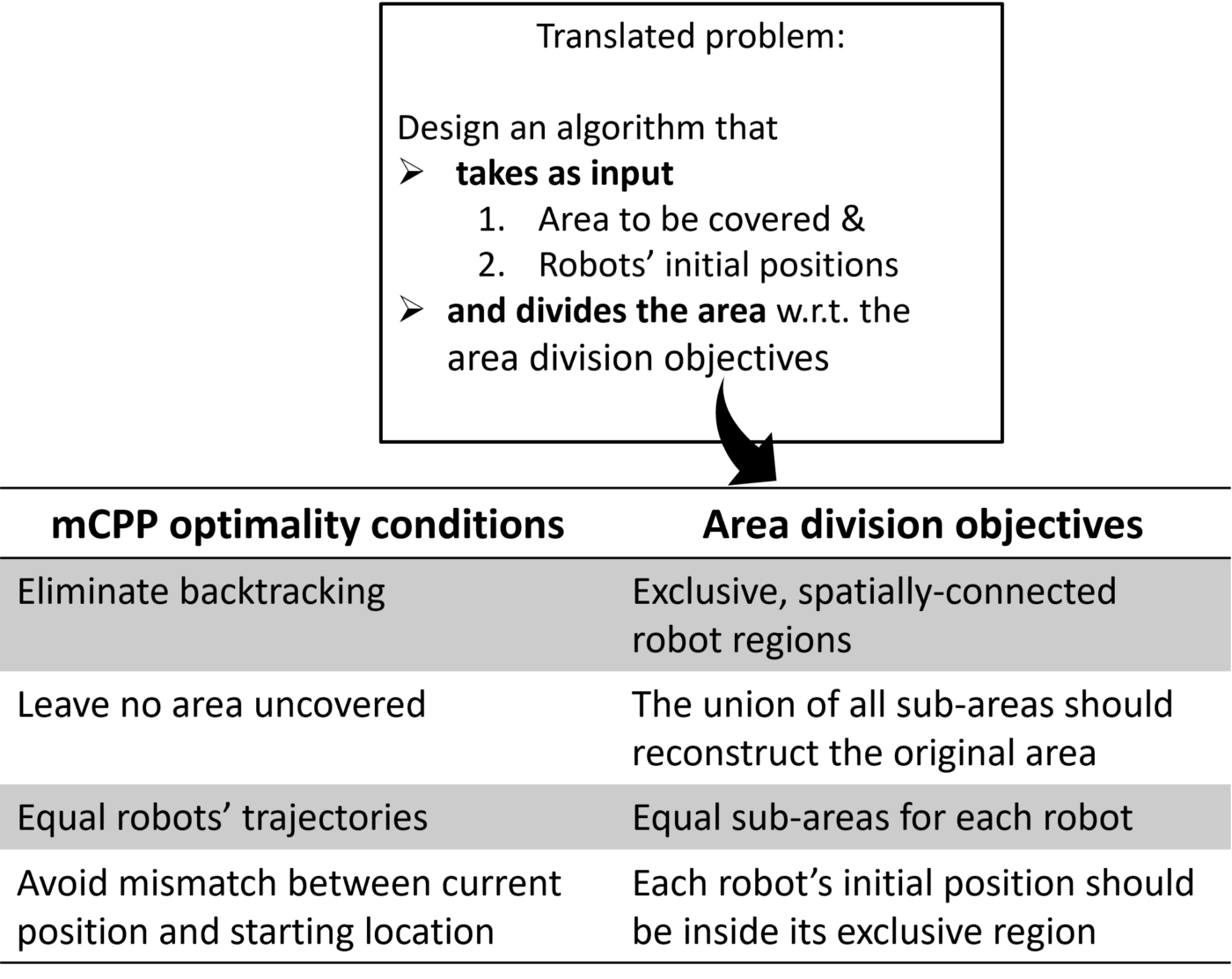
**BASIC BUILDING BLOCKS:**

Firstly we need to represent our area of interest into small blocks. Therefore, one of the most common areas representation techniques is to separate the field into identical cells (e.g. in the size of the robot), such that the coverage of each cell can be easily achieved. Apparently, for any arbitrarily shaped area, the union of the cells only approximates the target region. Thus this technique, which is also adopted in our approach, is termed **cellular decomposition**.

Now, let us assume we have only one robot to cover the whole area, and we need to find the optimal path for that one robot for the already given map. This is a relatively simple problem, i.e. “single robot coverage planning problem (inside an already known terrain)”. There are, of course, multiple methods to solve this problem. However, one of the dominant approaches is the **spanning tree coverage (STC)** algorithm, which is able to guarantee an optimal covering path in linear time. The term *optimal* encapsulates that the generated path does not revisit the same cell (non-backtracking property), completely covers the area of interest and achieves all the above without any preparatory effort (the robot can be initiated at any non-occupied cell). It is based on constructing a minimum spanning tree on the already cellular decomposed region, as discussed previously.



However, as mentioned above, we are concerned with finding the solution for multiple robots called mCPP (multiple CPP). Unfortunately, the mCPP problem was proven to be extremely difficult to be adequately addressed. As a matter of fact, solving mCPP with minimal covering time has been proven to be NP-hard. We shall now use the above-mentioned building blocks to try and tackle this problem.

​​

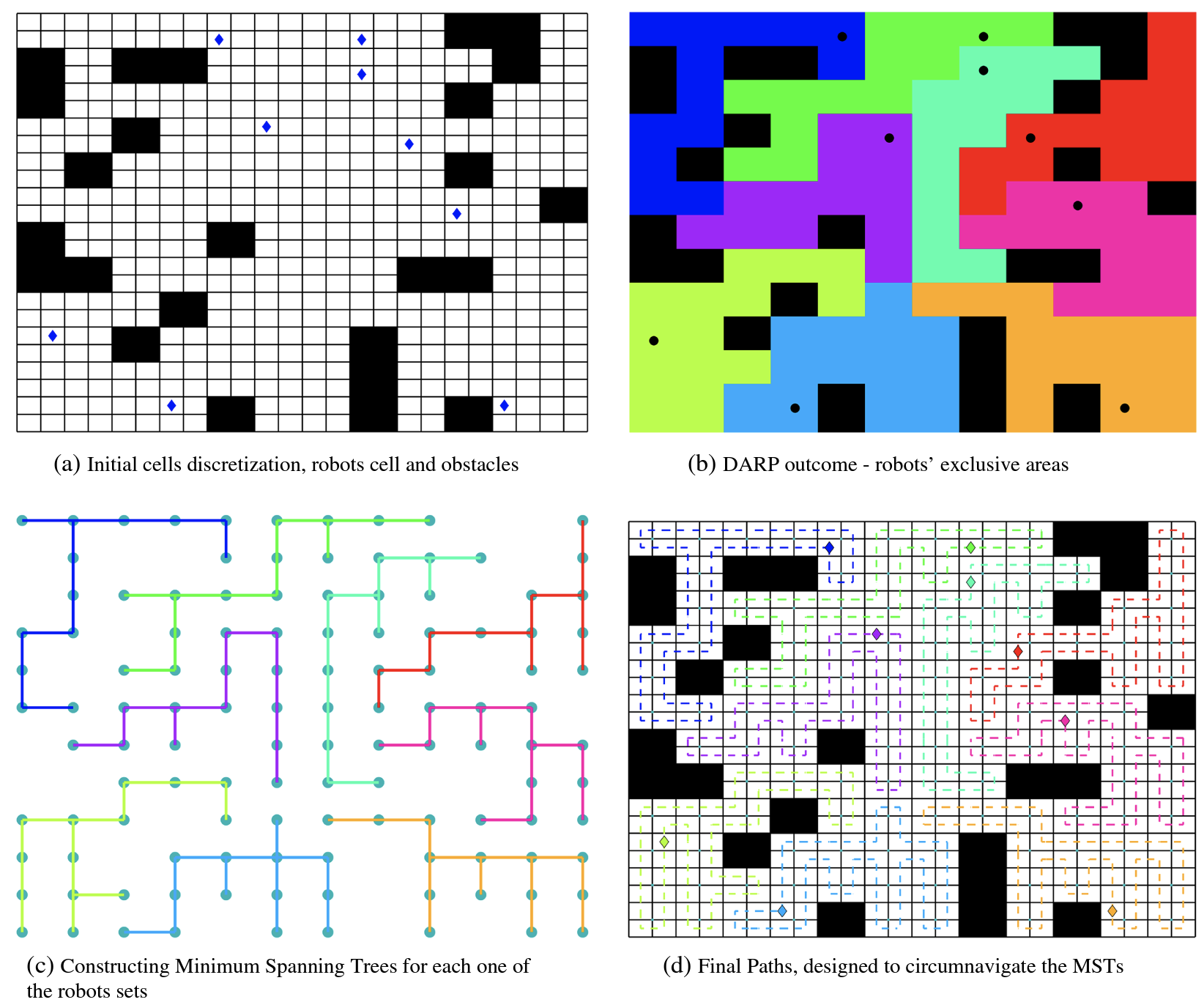
**BUILDING TOWARDS DARP:**

Now, the first thought that comes to mind is, can we use the STC method for mCPP. The answer is Yes, we can by

1. Extract a single path using the STC algorithm
2. Each robot follows the segment that starts from its initial position till it reaches the starting position of any other robot.

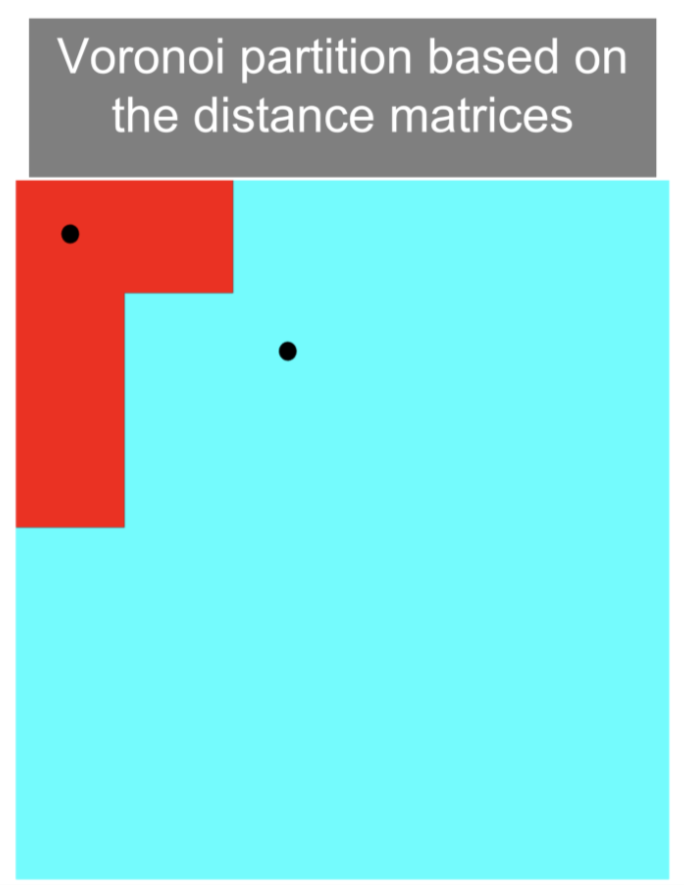
However, this results in Uneven robot trajectories, and the final solution is highly dependent on the initial positions of the robots.

So, the main idea of DARP is to apply STC individually for each robot by first assigning a certain area of the map to each robot. Thus here, we can manually control how much of the area do we want each robot to cover (usually equal areas for all robots).



Now the question is how do we divide the map and give individual regions to each robot optimally.

Again the most natural way seems to use the idea of Voronoi partitioning, i.e. assign a block to the robot with the closest initial position.



However, if we use this method, we again run into the problem where the area that each bot covers becomes dependent on the initial position of each bot.

Therefore we need to tune our process such that eventually, all robots have equal areas. We can basically use the concept of error control, i.e. in case the area of a bot is lesser than what it is supposed to be, we reduce its ‘distance’ to all cells so that more are now assigned to that robot, and similarly, for robots which have higher area, we increase their ‘distance’.

Let us now formally define this method,

,

**DARP 0.5:**

For every operational robot, an evaluation matrix is maintained. This evaluation matrix expresses the level of reachability (e.g. distance) between the cells of L (the map) and robot’s initial position . During each iteration, the assignment matrix A is constructed as follows:

Let the values of the matrix be initialised with some distance metric (e.g., Euclidean distance). Now, as mentioned earlier, the DARP algorithm’s core idea is that each evaluation matrix Ei can be appropriately “corrected” by a term as follows:

where is a scalar correction factor for the robot.

Now, let us define an error function for each robot J,

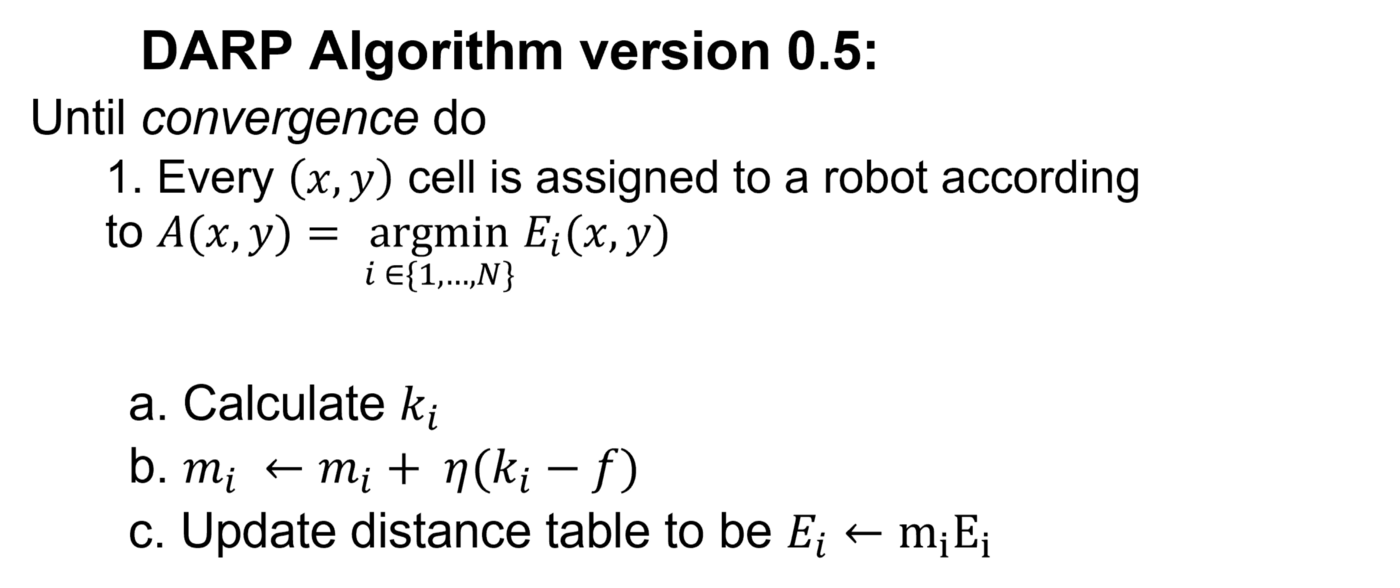
Where denotes the number of assigned cells to the robot, while denotes the number of cells that the has to cover.

Now, depending on the value of J for each robot, we can change the value of appropriately for all robots by using a standard gradient descent method,

Which can eventually be written as,

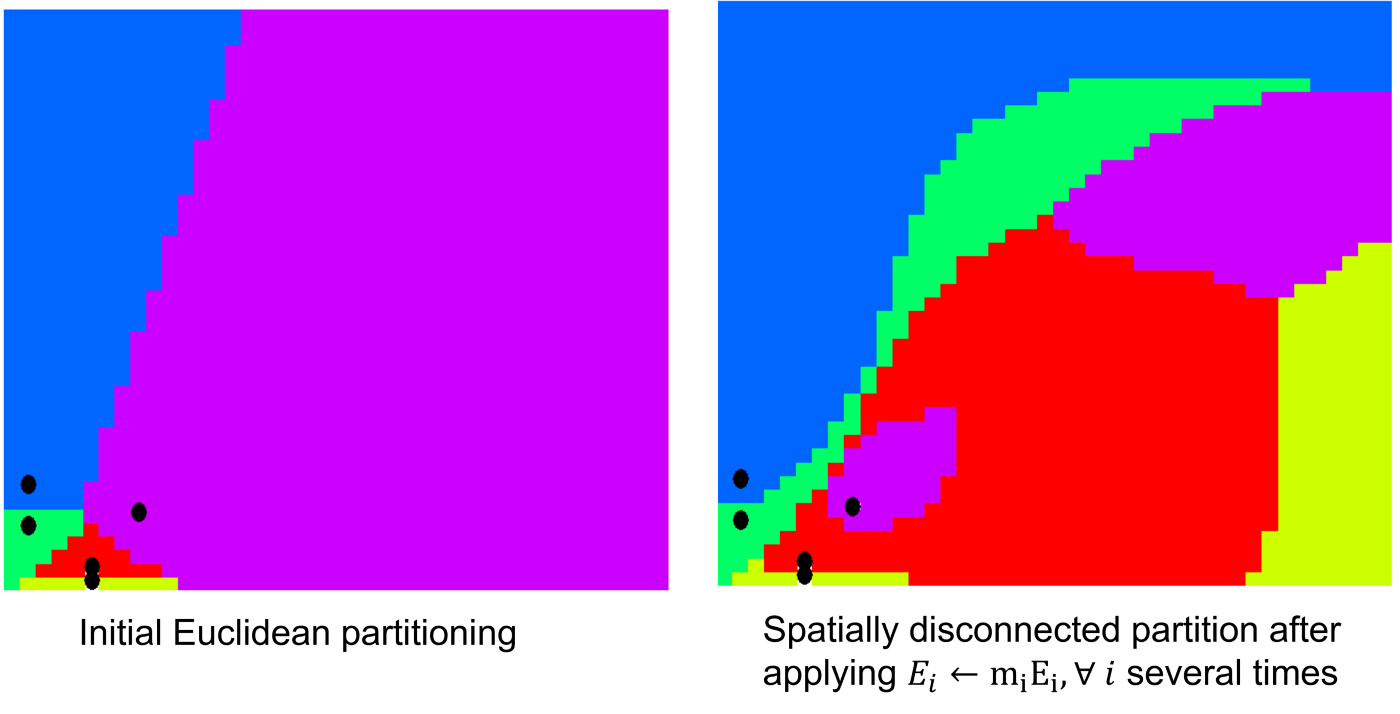
where c denotes a positive tunable parameter.

Thus, to summarise the algorithm so far,

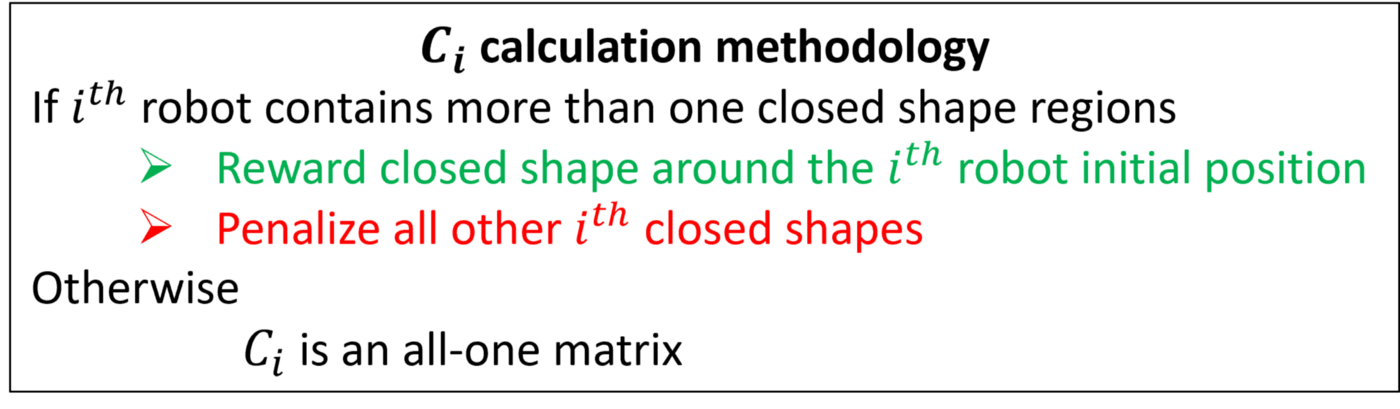


## **DARP 1.0:**

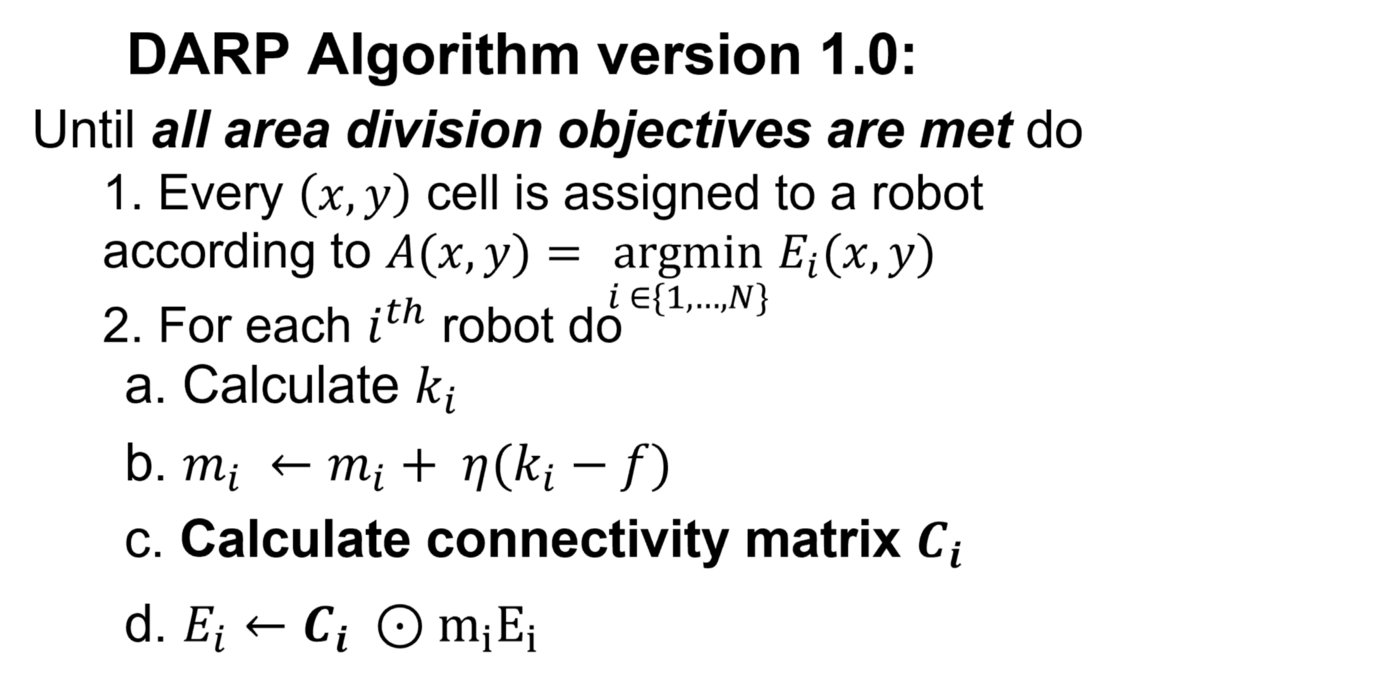
Although in the original Voronoi partitioning, spatial connectivity was guaranteed by design, now, in some cases is violated. This phenomenon is due to the fact that now the metric function for each robot, over which the assignment process is implemented, is no longer a distance function. For example, the following figure illustrates the initial partitioning (left-hand side) based on the Euclidean distance, and the subfigure on the right-hand side illustrates the assignment after some iterations.



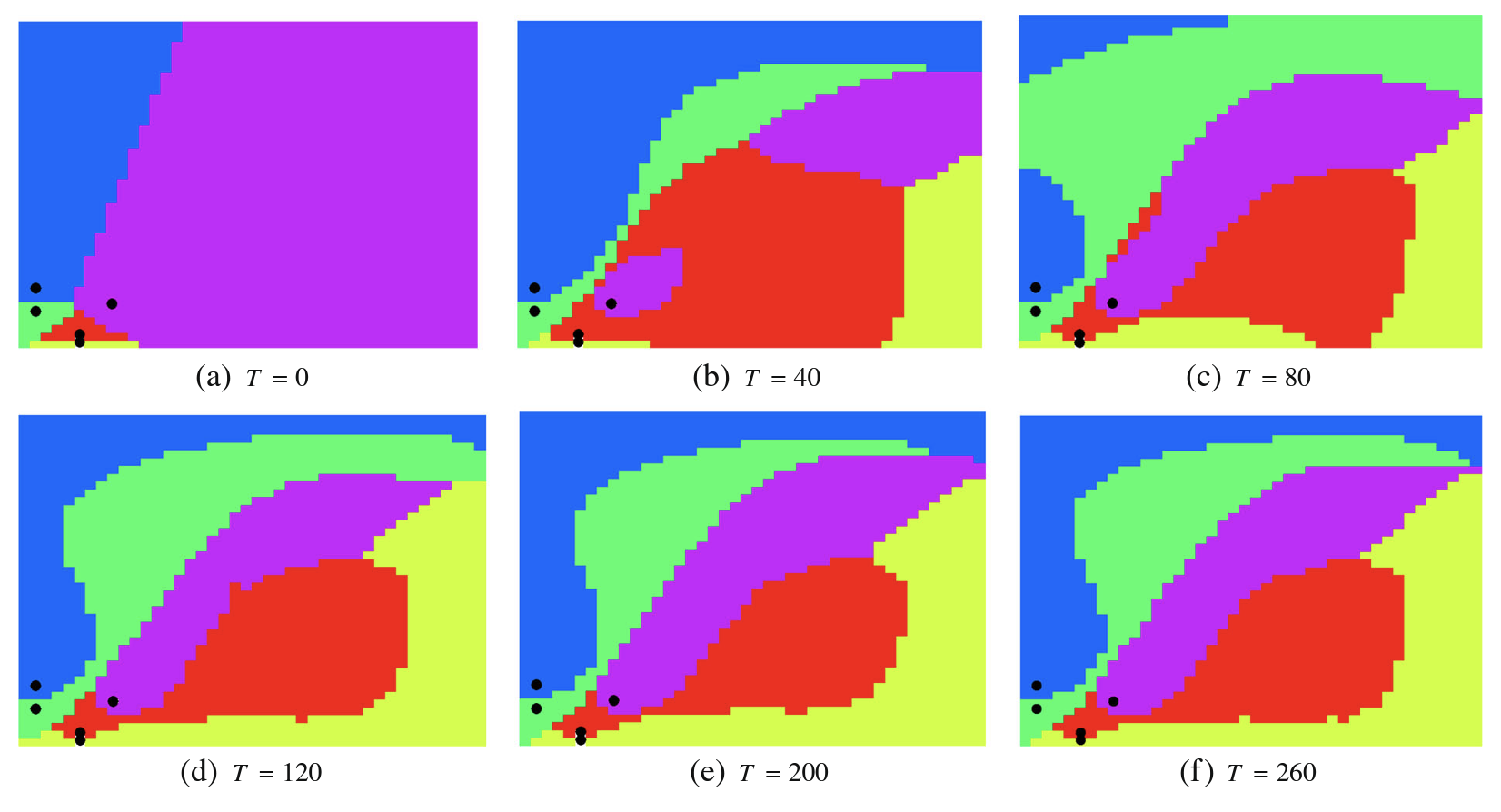
It is evident that the purple region is disconnected, and therefore with such an assignment, the non-backtracking guarantee is now at risk. To tackle this issue, an extra connectivity matrix is introduced. The rationale behind the calculation of this connectivity matrix is as follows:



Thus, The final update in the ith evaluation matrices is calculated as,



Even after this update, the DARP algorithm might give unconnected regions for some iterations. However, the algorithm claims that in most cases, this will eventually help find the optimal solution. More precisely, the DARP algorithm is capable of escaping the local minima by temporarily violating the condition about the connectivity of each ith robot assignment matrix. Afterwards, the algorithm gradually eliminates unconnected areas by reinforcing the robot’s evaluation Ei around the original (the one that the robot lies in) subarea. By the time the connectivity inside the exclusive robot sets Li is restored, the evaluation matrices will have completely changed their forms, and ideally towards the optimal cells assignment.



**DEALING WITH SPATIAL DISCONNECTIVITY:**

When we used DARP in practice, we came across multiple corner cases where the algorithm failed to converge, one of the cases being when the initial positions of the robots were too close to each other. We are of the belief that the spatial disconnectivity problem of DARP is responsible for this, and once the regions become disconnected, either it stays that way or it takes too many interactions to join back. To tackle this problem we introduced a filter in the sense that only if all pairs of robots were separated a constant minimum distance will we try and find the optimal path using DARP.

In some other cases, even though the bots were far spaced, the algorithm was unable to converge. We found out using trial and error that if we give some leeway in terms of area covered by each bot (e.g., for two bots instead of exactly 50%, we use 45% and 55% areas), the algorithm is able to converge. Basically, sometimes the algorithm trying to assign exactly equal areas to all bots fails to converge. Thus, using slightly unequal areas seems the way forward for those cases.