Appendix
Multi-Robot Adversarial Coverage

This documents contains the full proofs and the pseudocode of the algorithms mentioned in the paper "Multi-Robot Adversarial Coverage", submitted to the IJCAI 2016 workshop on Multi-Agent Path Finding.

Theorem 1. *(Completeness) Algorithm MRAC generates paths for the robots that together cover every cell accessible from their starting locations.*

Proof. MRAC partitions the target area into k connected areas, whose union is equal to the target area. Each of these areas is eventually assigned to one of the robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). The areas are covered by using the GAC algorithm. Previous work has shown that GAC is complete, i.e. that it produces a path that covers all the accessible cells in its given area (Theorem 8 in [**?**]). Thus, the union of the coverage paths of the k connected areas covers every accessible cell in the target area. □

Theorem 2. *(Robustness) Algorithm MRAC guarantees that the coverage will be completed as long as at least one robot remains active.*

Proof. The target area is divided into a set of connected areas. Each area is eventually assigned to one of the robots. If a robot is stopped while covering its assigned area, the uncovered parts of this area are returned to the pool of unassigned areas (in procedure Reallocate_Area). These sub-areas are eventually assigned to one of the remaining robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). □

Theorem 3. *Let $\mathcal{R}$ be a group of k robots $\mathcal{R} = \{R_1, ..., R_k\}$. Let l be the number of dangerous threat levels. Let $\mathcal{A}$ be the group of connected areas and let C be the set of cells on the connecting path between two areas. Then the expected number of cells the team $\mathcal{R}$ will be able to cover before all robots in $\mathcal{R}$ are stopped is at least:*

$$E(\mathcal{R}) \geq \sum_{i=1}^{k} E(R_i) - |C| \tag{1}$$

Proof. Since different robots in the team cover different areas (when one area is assigned to more than one robot, it is split between them), the expected number of cells that will be covered by the entire team is equal to the total number of cells that will be covered by each robot individually minus the number of cells on the connecting paths between areas, since those might be visited multiple times by different robots. □

Theorem 4. *The worst-case coverage time for k robots is equal to that of a single robot and the best-case coverage time is approximately $1/k$ the coverage time of a single robot.*

Proof. The worst-case scenario is where all the robots start at locations that are all inside dangerous areas, and thus they will be stopped by threats in the beginning of their coverage paths. In such case, using more than one robot will not improve the total coverage time. The best-case scenario is when the target area consists of only one contiguous safe area (or a few large safe areas with connecting paths that have very low risk). In such scenario, the area is split into k almost equally-sized components that are covered concurrently by the k different robots. In such scenario, the coverage time of MRAC highly depends on the quality of the graph partitioning algorithm, which has no theoretical guarantee, however, in practice, it almost always generated equally-sized subgraphs. The coverage time in this case also depends on the initial locations of the robots, e.g., if the robots start the coverage at the same location, it will take them some time to move to the their assigned areas. □

We now provide the pseudocode of the algorithms.

Algorithm 1 Multi_Robot_Adversarial_Coverage (MRAC)(G, $\mathcal{R}$)

```
input: $G$ - a grid representing the target area, $\mathcal{R}$ - group of $k$ robots
output: $\mathcal{A}$ - list of connected areas
 1: $\mathcal{A} \leftarrow \emptyset$
 2: Group the cells in $G$ into $l+1$ threat levels, $T_0, ..., T_l$
 3: for each threat level $i$, $0 \leq i \leq l$ do
 4:     Build the graph $H_i$ induced from the cells in $T_i$
 5:     Find the connected components (areas) of $H_i$ using DFS
 6:     Let $A_1, ..., A_k$ be the connected areas of $H_i$
 7:     for each area $A_j$, $1 \leq j \leq k$ do
 8:         $A_j.state \leftarrow S_{unassigned}$
 9:         $A_j.level \leftarrow i$
10:         Add $A_j$ to $\mathcal{A}$
11: Allocate_Areas_To_Robots($\mathcal{A}$, $\mathcal{R}$)
```

```
1: procedure ALLOCATE_AREAS_TO_ROBOTS($\mathcal{A}$, $\mathcal{R}, d$)
   input: $\mathcal{A}$ - list of connected areas, $\mathcal{R}$ - the group of robots, $d$ - maximal area density
2:     $\mathcal{S} \leftarrow \{A | A \in \mathcal{A} \land A.level = 0\}$          ▷ the safe areas
3:     for each robot $R \in \mathcal{R}$ do
4:         {Find safest paths to all safe areas}
5:         for each area $A \in \mathcal{S}$ do
6:             $P_A \leftarrow$ Find_Safest_Path_To_Area($A$, $R.location$)
7:         {Find safest non-dense area}
8:         Let $A_1, ..., A_k$ be the areas sorted by $P_A.cost$
9:         $i \leftarrow 1$
10:        while $i \leq k$ and $R.area = null$ do
11:            if $|A.initial\_robots| \cdot d \leq |A.cells|$ then ▷ check if area is not too dense with robots
12:                Add $R$ to $A.initial\_robots$
13:            else
14:                $i \leftarrow i + 1$
15:        {Split the areas that are allocated to multiple robots}
16:        for each area $A \in \mathcal{S}$ do
17:            if $|A.initial\_robots| > 1$ then
18:                Split_Area_Between_Robots($A$)
19:            else if $|A.initial\_robots| = 1$ then
20:                $R \leftarrow A.initial\_robots[0]$
21:                $R.path \leftarrow P_A$
22:                Assign_Area_To_Robot($A$, $R$)
```

```
1: procedure FIND_SAFEST_PATH_TO_AREA($A, s$)
   input: $A$ - a connected area, $s$ - a cell in the grid
   output: $P$ - the safest path connecting $s$ to a cell in $A$
2:     Build the graph $H$ induced from the grid's cells
3:     Run Dijkstra shortest paths algorithm on $H$ from $s$
4:     for each cell $c \in A$ do
5:         $P_c \leftarrow$ the safest path from $s$ to $c$
6:     return $\arg\min_{P_c} cost(P_c)$
```

```
1: procedure SPLIT_AREA_BETWEEN_ROBOTS($A, \mathcal{R}$)
   input: $A$ - a connected area, $\mathcal{R}$ - the group of robots globals: $\mathcal{A}$ - list of
   connected areas
2:     Build the graph $G$ induced from the area $A$'s cells
3:     $k \leftarrow |\mathcal{R}|$
4:     Partition_Graph($G$, $k$)
5:     Let $G_1, G_2, ...G_k$ be the subgraphs created from the partition
6:     for $i \leftarrow 1$ to $k$ do
7:         Create a subarea $A_i$
8:         $A_i.cells \leftarrow$ the nodes in $G_i$
9:         $A_i.level \leftarrow A.level$
10:        Add $A_i$ to $\mathcal{A}$
11:        {Compute the safest path of each robot to the sub-area}
12:        for $j \leftarrow 1$ to $k$ do
13:            $P_{ij} \leftarrow$ Find_Safest_Path_To_Area($A_i, R_j.location$)
14:            $C_{ij} \leftarrow P_{ij}.cost$                      ▷ the cost matrix
15:    $O \leftarrow$ Hungarian_Method($C$)   ▷ $O$ is the optimal assignment of robots to
   sub-areas
16:    $O_i \leftarrow$ optimal assignment of robot $i$
17:    for $i \leftarrow 1$ to $k$ do
18:        $R_i.path \leftarrow P_{O_i,i}$
19:        Assign_Area_To_Robot($O_i$, $R_i$)
20:    Remove $A$ from $\mathcal{A}$
```

```
1: procedure ASSIGN_AREA_TO_ROBOT($A, R$)
   input: $A$ - a connected area, $R$ - a robot
2:     $A.robot \leftarrow R$
3:     $A.state \leftarrow S_{assigned}$
4:     $R.area \leftarrow A$
5:     $R.state \leftarrow S_{traveling}$
```

Algorithm 2 Robot_Action(R)

```
1:  switch $R.state$ do
2:      case $S_{traveling}$
3:          $c \leftarrow$ next cell on $R.path$
4:          Mark $c$ as visited
5:          $A \leftarrow$ the area that contains $c$
6:          if all cells in $A$ are visited then
7:              $A.state \leftarrow S_{completed}$
8:          if robot was hit by a threat in $c$ then
9:              $R.state \leftarrow S_{dead}$
10:             Reallocate_Area($R.area$)
11:         else if $c$ is the last cell on $R.path$ then
12:             $R.path \leftarrow$ Area_Coverage($R.area, R.location$)
13:             $R.state \leftarrow S_{covering}$
14:     case $S_{covering}$
15:         $c \leftarrow$ next cell on $R.path$
16:         Mark $c$ as visited
17:         if robot was hit by a threat in $c$ then
18:             $R.state \leftarrow S_{dead}$
19:             Reallocate_Area($R.area$)
20:         else if $c$ is the last cell on $R.path$ then
21:             $R.area.state \leftarrow S_{completed}$
22:             Allocate_Next_Area($R$)
23:     case $S_{done}$, $S_{idle}$
24:         if an unassigned area exists in $\mathcal{A}$ then
25:             Allocate_Next_Area($R$)
26:     case $S_{dead}$
27:         $R.area.state \leftarrow S_{unassigned}$
```

```
procedure ALLOCATE_NEXT_AREA($R$)
input: $R$ - a robot
globals: $\mathcal{A}$ - the list of connected areas
    $U \leftarrow \{A | A \in \mathcal{A} \wedge A.state = S_{unassigned}\}$
    if $|U| > 0$ then
        {Allocate the safest unassigned area with the safest path from the robot's current location}
        $s \leftarrow \min_{level} U$
        $S \leftarrow \{A | A \in \mathcal{U} \wedge A.level = s\}$
        for each area $A \in S$ do
            $P_A \leftarrow$ Find_Safest_Path_To_Area($R, A$)
        $A \leftarrow \arg\min_A cost(P_A)$
        $R.path \leftarrow P_A$
        Assign_Area_To_Robot($A$, $R$)
    else
        {If all the areas are assigned, find an area that the robot can help covering}
        if not Find_Area_To_Share($R$) then
            $R.state \leftarrow S_{done}$
```

```
procedure ALLOCATE_NEXT_AREA($R$)
input: $R$ - a robot
globals: $\mathcal{A}$ - the list of connected areas
    $U \leftarrow \{A | A \in \mathcal{A} \wedge A.state = S_{unassigned}\}$
    if $|U| > 0$ then
        {Allocate the safest unassigned area with the safest path from the robot's current location}
        $s \leftarrow \min_{level} U$
        $S \leftarrow \{A | A \in \mathcal{U} \wedge A.level = s\}$
        for each area $A \in S$ do
            $P_A \leftarrow$ Find_Safest_Path_To_Area($R, A$)
        $A \leftarrow \arg\min_A cost(P_A)$
        $R.path \leftarrow P_A$
        Assign_Area_To_Robot($A$, $R$)
    else
        {If all the areas are assigned, find an area that the robot can help covering}
        if not Find_Area_To_Share($R$) then
            $R.state \leftarrow S_{done}$
```

```
1: procedure FIND_AREA_TO_SHARE($R$)
   input: $R$ - a robot
   output: a flag indicating whether an area was found
   globals: $\mathcal{A}$ - the list of connected areas
2:     $D \leftarrow \{A | A \in \mathcal{A} \wedge A.state = S_{assigned}\}$
3:     for each area $A \in D$ do
4:         $P_A \leftarrow$ Find_Safest_Path_To_Area($R, A$)
5:         $n \leftarrow$ number of unvisited cells in $A$
6:         if $|P_A| > n$ then
7:             Remove $A$ from $D$            ▷ no point of sending the robot there
8:     if $D = \emptyset$ then return false
9:     {Send the robot to the area with the safest path}
10:    $A \leftarrow \arg\min_A cost(P_A)$
11:    {Find the unvisited parts of the designated area}
12:    Build the graph $H$ induced from the unvisited cells in $A$
13:    Find the connected components (areas) $A_1, ..., A_k$ of $H$ using DFS
14:    if $k = 1$ then
15:        {Split the area into two balanced parts}
16:        Build the graph $G$ induced from the area $A$'s cells
17:        $(G_1, G_2) \leftarrow$ Partition_Graph($G$, 2)
18:        Let $A_1, A_2$ the two sub-areas induced from $G_1, G_2$
19:    {Replace the designated area with its unvisited parts}
20:    Remove $A$ from $\mathcal{A}$
21:    for each subarea $A_i$ do
22:        $A_i.level \leftarrow A.level$
23:        $A_i.state \leftarrow S_{unassigned}$
24:        Add $A_i$ to $\mathcal{A}$
25:    {Reassign the robot covering area $A$ to the part with the safest path
   from its current location}
26:    Assign_Robot_To_Safest_Area($A_i$, $A.robot$)
27:    Assign_Robot_To_Safest_Area($A_i$, $R$)
28:    return true
```

```
1: procedure ASSIGN_ROBOT_TO_SAFEST_AREA($\mathcal{A}, R$)
   input: $\mathcal{A}$ - group of connected areas, $s$ - a robot
2:     for each area $A \in \mathcal{A}$ do
3:         $P_A \leftarrow$ Find_Safest_Path_To_Area($R, A$)
4:     $A \leftarrow \arg\min_A cost(P_A)$
5:     Assign_Area_To_Robot($A$, $R$)
```

Algorithm 3 Area_Coverage(A, c)

```
input: an area $A$, and a starting cell $s$
output: a coverage path $P$ that covers all unvisited cells in $A$
1: $P \leftarrow \emptyset$
2: Build the graph $H$ induced from $A$'s cells with the weight function $w$ from eq. (??)
3: Add the starting cell $s$ to $P$
4: Mark $s$ as visited
5: while there are unvisited cells in $A$ do
6:     Run Dijkstra's shortest paths algorithm on $H$ from $s$
7:     $v \leftarrow$ an unvisited node in $A$ with minimum weighted distance from $s$
8:     Add the path $s \rightsquigarrow v$ to $P$
9:     Mark $v$ as visited
10:    $s \leftarrow v$
11: return $P$
```

```
1: procedure Reallocate_Area($A$)
   input: $A$ - a connected area
   globals: $\mathcal{A}$ - the list of connected areas, $\mathcal{R}$ - the group of robots
2:     Build the graph $H$ induced from the unvisited cells in $A$
3:     Find the connected components (areas) $A_1, ..., A_k$ of $H$ using DFS
4:     Remove $A$ from $\mathcal{A}$
5:     for each subarea $A_i$ do
6:         $A_i.level \leftarrow A.level$
7:         $A_i.state \leftarrow S_{unassigned}$
8:         Add $A_i$ to $\mathcal{A}$
```