SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan for Formational Positioning

PATRICK MACALPINE, ERIC PRICE, PETER STONE
The University of Texas at Austin
Austin, TX 78712 USA
{patmac, ecprice, pstone}@cs.utexas.edu

Problem Formulation

How to assign agents to target positions in a 1-to-1 mapping? Want to minimize time for all agents to reach targets (makespan) and avoid collisions.

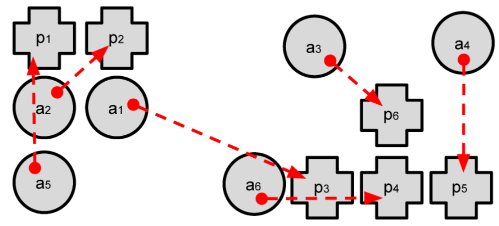
Required properties of a role assignment function to be *CM Valid* (Collision-avoiding with Minimal-makespan):

1. **Minimizing makespan** - it minimizes the maximum distance from an agent to target, with respect to all possible mappings
2. **Avoiding collisions** - agents do not collide with each other

Desirable but not necessary property:

3. **Dynamically consistent** - role assignments don't change or switch as agents move toward target positions

Example Problem and Solution



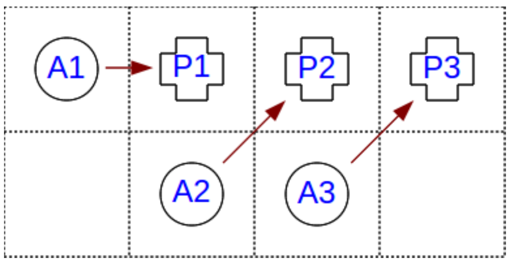
$a_1 \rightarrow p_3$ is minimal longest distance across all possible mappings

Assumptions:

- Agents are interchangeable: any agent can be assigned to any target position
- No two agents or targets occupy the same position
- Agents are treated as zero width point masses
- Agents move at **same constant speed** along **straight line paths** to assigned targets

Minimal Maximum Distance Recursive (MMDR) Role Assignment Function

Recursively minimize longest distance any agent must travel



Lowest lexicographical cost (shown with arrows) to highest cost ordering of mappings from agents (A1,A2,A3) to role positions (P1,P2,P3). Each row represents the cost of a single mapping.

1: $\sqrt{2}$ (A2→P2), $\sqrt{2}$ (A3→P3), 1 (A1→P1)
2: 2 (A1→P2), $\sqrt{2}$ (A3→P3), 1 (A2→P1)
3: $\sqrt{5}$ (A2→P3), 1 (A1→P1), 1 (A3→P2)
4: $\sqrt{5}$ (A2→P3), 2 (A1→P2), $\sqrt{2}$ (A3→P1)
5: 3 (A1→P3), 1 (A2→P1), 1 (A3→P2)
6: 3 (A1→P3), $\sqrt{2}$ (A2→P2), $\sqrt{2}$ (A3→P1)

***CM Valid* and dynamically consistent**

Implementation

Goal: **Transform edge distances** to be set of weights such that the weight of any edge e is greater than the sum of weights of all edges with distances less than e.

Lemma 1. *Denote* $W_n := \{w_0, ..., w_n\}$ *where* $w_i := 2^i$. *Then* $\forall W \in \mathcal{P}(W_{n-1}) : w_n > \sum W$.

1. **Transform edge distances** to new weights:
   - Sort edges in ascending order of distance
   - Set weights to be 2^i where i is the index of an edge in this sorted list

Example: 100_2 (4) $> 010_2$ (2) $+ 001_2$ (1) $= 011_2$ (3)

2. **Run Hungarian algorithm** with modified weights
   - Returns MMDR mapping

O(n^5) Polynomial Time Algorithm

Require:
Agents $:= \{a_1, ..., a_n\}$; *Positions* $:= \{p_1, ..., p_n\}$
Edges $:= \{\overline{a_1p_1}, \overline{a_1p_2}, ..., \overline{a_np_n}\}$; $|\overline{a_ip_j}| :=$ `euclideanDist`(a_i,p_j)

edgesSorted := `sortAscendingDist`(*Edges*)
lastDistance := -1
rank, *currentIndex* := 0
for all $e \in$ *edgesSorted* **do**
    if $|e| >$ *lastDistance* **then**
        rank := *currentIndex*
    lastDistance := $|e|$
    $|e| := 2^{rank}$
    currentIndex := *currentIndex* + 1
return `hungarianAlg`(*edgesSorted*)

Time: O(n^2) bits weights X O(n^3) Hungarian algorithm = **O(n^5)***
Space: O(n^2) bits weights X O(n) weights stored at a time = **O(n^3)**
There **exists O(n^4) algorithm** [1]

*Processors can compare bits in weights in parallel reducing running time by factor of word length (e.g. 64 on a 64-bit processor).

Minimal Maximum Distance + Minimum Sum Distance² (MMD+MSD²) Role Assignment Function

Find a perfect matching M that:

1. Has a **minimum-maximal edge**
2. **Minimizes the sum of distances squared**

$$\mathbb{M}'' := \{X \in \mathbb{M} \mid \|X\|_\infty = \min_{M \in \mathbb{M}}(\|M\|_\infty)\} \quad (1)$$

$$M^* := \underset{M \in \mathbb{M}''}{\operatorname{argmin}}(\|M\|_2^2) \quad (2)$$

***CM Valid* but not dynamically consistent**

Implementation

Minimal-maximum Edge Perfect Matching Algorithm: O(n^3) breadth-first search using Ford-Fulkerson algorithm to find the minimal maximum length edge in a perfect matching

1. **Find minimal-maximum edge** in perfect matching with weight w
2. **Remove all edges with weight greater than w** from graph
3. Use Hungarian algorithm to **compute perfect matching with max sum of distances squared**

O(n^3) Polynomial Time Algorithm

Require:
Agents $:= \{a_1, ..., a_n\}$; *Positions* $:= \{p_1, ..., p_n\}$
Edges $:= \{\overrightarrow{a_1p_1}, \overrightarrow{a_1p_2}, ..., \overrightarrow{a_np_n}\}$; $|\overrightarrow{a_ip_j}| :=$ `euclideanDist`$(a_i,p_j)^2$

longestEdge := `getMinMaxEdgeInPerfectMatching`(*Edges*)
minimalEdges := $e \in$ *Edges*, s.t. $|e| \leq |$*longestEdge*$|$
return `hungarianAlg`(*minimalEdges*)

Time: O(n^3) Min-max Edge Alg. + O(n^3) Hung. Alg. = **O(n^3)**
Space: Breadth-first search of Ford-Fulkerson = **O(n^2)**

Role Assignment Function Properties

Function Properties

Function	Min. Make.	No Coll.	Dyn. Con.
MMD+MSD²	Yes	Yes	No
MMDR	Yes	Yes	Yes
MSD²	No	Yes	No
MSD	No	No	No
Random	No	No	No
Greedy	No	No	No

Assigning 10 robots to 10 targets on a 100 X 100 grid

Function	Avg. Make.	Avg. Dist.	Dist. StdDev
MMD+MSD²	45.79	27.38	10.00
MMDR	45.79	28.02	9.30
MSD²	48.42	26.33	10.38
MSD	55.63	25.86	12.67
Random	90.78	52.14	19.38
Greedy	81.73	28.66	18.95

MSD: Minimize sum of distances between robots and targets.
MSD²: Minimize sum of distances² between robots and targets.
Greedy: Assign robots to targets in order of shortest distances.
Random: Random assignment of robots to targets.

Role Assignment Algorithm Analysis

Time and space complexities

Algorithm	Time Complexity	Space Complexity
MMD+MSD²	O(n^3)	O(n^2)
MMDR O(n^4)	O(n^4)	O(n^2)
MMDR O(n^5)	O(n^5)	O(n^3)
MMDR dyna	O($n^2 2^{(n-1)}$)	O($n\binom{n}{n/2}$)
brute force	O($n!n$)	O(n)

Running time in milliseconds for different values of n

Algorithm	$n = 10$	$n = 20$	$n = 100$	$n = 300$	$n = 10^3$	$n = 10^4$
MMD+MSD²	0.016	0.062	1.82	21.2	351.3	115006
MMDR O(n^4)	0.049	0.262	17.95	403.0	14483	—
MMDR O(n^5)	0.022	0.214	306.4	40502	—	—
MMDR dyna	0.555	2040	—	—	—	—
brute force	317.5	—	—	—	—	—

RoboCup Robot Soccer Case Studies

SCRAM role assignment performed better than static role assignment in both the RoboCup 2D and 3D Simulation Leagues

Future Work

- Task specialization: agents assigned to subset of targets
- Heterogeneous agents moving at different varying speeds
- Have agents also avoid known fixed obstacles
- Model robots as having non-zero width mass
- Make algorithms distributed

References

[1] P. Sokkalingam and Y. P. Aneja. Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, 23(1):27–33, 1998.

More Information

Videos and C++ Code: **http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2013/html/scram.html**