To appear in *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS), Paris, France, May 2014.

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

(Extended Abstract)

Patrick MacAlpine
Univ. of Texas at Austin
Austin, TX , USA
patmac@cs.utexas.edu

Eric Price
MIT
Cambridge, MA, USA
ecprice@mit.edu

Peter Stone
Univ. of Texas at Austin
Austin, TX, USA
pstone@cs.utexas.edu

ABSTRACT

Teams of mobile robots often need to divide up subtasks efficiently. In spatial domains, a key criterion for doing so may depend on distances between robots and the subtasks' locations. This research considers a specific such criterion, namely how to assign interchangeable robots to a set of target locations such that the makespan (time for all robots to reach their target locations) is minimized while also preventing collisions among robots. We provide an overview of a scalable multiagent dynamic role assignment system known as SCRAM (Scalable Collision-avoiding Role Assignment with Minimal-makespan). SCRAM uses a graph theoretic approach to map agents to target locations such that our objectives for both minimizing the makespan and avoiding agent collisions are met. SCRAM scales to thousands of agents as role assignment algorithms run in polynomial time.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and tree search strategies*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

Coordination; Formations; Positioning; Mobile Robots

1. INTRODUCTION

Previous work on assigning agents to target positions has focused on minimizing the sum of distances agents travel which is the *assignment problem* [4]. Our work differs as we instead minimize the *makespan* (time for all agents to reach goal positions) Minimizing the makespan is a decisive factor in performance when agents are moving to target positions to complete a shared task where all agents must be in place before the task can be completed and/or started. Such tasks include those requiring agents be synchronized when they start jobs at their target positions (e.g. mobile robots assuming necessary positions on an assembly line) and scenarios for which the bottleneck is the time it takes for the last agent to get to its target position (e.g. warehouse robots fetching items for an order to be shipped).

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

We refer to our role assignment and positioning system as SCRAM (Scalable Collision-avoiding Role Assignment with Minimal-makespan).[1] It provides a collision free mapping of agents to target positions, minimizes the makespan, and scales to thousands of agents. This abstract provides an overview of SCRAM, and summarizes our initial theoretical and empirical analysis of the role assignment problem. Our deeper analysis of SCRAM, including its application to the RoboCup[2] robot soccer domain, is reserved for future publications.

2. ROLE ASSIGNMENT PROBLEM

Let there be n homogeneous mobile agents with current positions $A := \{a_1, ..., a_n\}$, and we want to assign them to move to n specified target positions or roles $P := \{p_1, ..., p_n\}$ such that the time for agents to have reached every goal position is minimized under the constraint that no agents collide with each other. This problem can be thought of as finding a perfect matching M^* of a weighted bipartite graph $G := (A, P, E)$ that meets the above criteria with the weight for each edge in E being the Euclidean distance between associated agent and target positions.

For theoretical analysis we model agents as point masses with zero width. Additionally, we assume no two agents and no two target positions occupy the same position, and that all agents move toward fixed target positions along a straight line at the same constant speed.

We call a role assignment *CM valid* (Collision-avoiding with Minimal-makespan) if it satisfies two properties:

1. *Minimizing longest distance* - M^* minimizes the longest distance from an agent to target, with respect to all possible mappings.
2. *Avoiding collisions* - agents do not collide as they move to their assigned positions.

A third desirable property, although not necessary for a role assignment function f to be *CM valid*, is the following:

3. *Dynamically consistent* - Given a *fixed* set of target positions, if f outputs a mapping M of agents to targets at time T, then f outputs M for every time $t > T$ as agents move to the targets specified by M.

Dynamic consistency is desirable as otherwise agents might unduly thrash between roles thus impeding progress.

[1]Videos of SCRAM in action, as well as C++ implementations of the role assignment algorithms and an online appendix, can be found at `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2013/html/scram.html`
[2]`http://www.robocup.org/`

3. ROLE ASSIGNMENT FUNCTIONS

We present two *CM valid* assignment functions. Polynomial time implementations of the functions with analysis of their time and space complexities are also given.

Minimum Maximal Distance Recursive (MMDR)

One potential role assignment function is to find a mapping of agents to target positions which recursively minimizes the maximum distance that any agent travels. We refer to this as this the Minimum Maximal Distance Recursive (MMDR) function. It is also known as the *lexicographic bottleneck assignment problem* [4]. In previous work we introduced MMDR and presented an exponential time dynamic programming implementation of MMDR [3]. We provide evidence that MMDR is both *CM valid* and dynamically consistent in an online appendix.[1]

We can compute the MMDR role assignment function in polynomial time by transforming MMDR into the *assignment problem* (finding a perfect matching in a bipartite graph that minimizes the sum of edge weights) which is solvable by the Hungarian algorithm [2] in $O(n^3)$ time.

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

To transform MMDR into the *assignment problem* we modify the weights of the edges of our bipartite graph to be a set of values such that the weight of any edge e is greater than the sum of weights of all edges with weight values less than that of e. A key insight into this transformation is expressed in Lemma 1. By sorting all edges in ascending order by distance, and then relabeling edge weights to be the value 2^i where i is the index of an edge in this sorted list, the sum of all edge weights of shorter distance edges will be less than any sum of edge weights with a longer edge. Solutions to the *assignment problem* return lowest cost MMDR mappings as the sum of modified weights of any mapping with a higher cost is greater than lower cost mappings.

Algorithm 1 gives a polynomial time solution for computing MMDR. Time complexity is dominated by the $O(n^3)$ Hungarian algorithm. Note that our transformed edge weights, represented as bit vectors with the *ith* bit of a 2^i value turned on, are of size n^2. The Hungarian algorithm must do comparisons of these weights and thus the time complexity of Algorithm 1 is $O(n^5)$. As our implementation of the Hungarian algorithm stores length n lists of size n^2 transformed weights, Algorithm 1 has a $O(n^3)$ space complexity.

Algorithm 1 MMDR $O(n^5)$ Polynomial Time Impl.

Input:
$Agents := \{a_1, ..., a_n\}$; $Positions := \{p_1, ..., p_n\}$
$Edges := \{\overline{a_1 p_1}, \overline{a_1 p_2}, ..., \overline{a_n p_n}\}$; $|\overline{a_i p_j}| :=$ `euclideanDist`(a_i, p_j)
$edgesSorted :=$ `sortAscendingDist`$(Edges)$
$lastDistance := -1$
$rank, currentIndex := 0$
for each $e \in edgesSorted$ **do**
  if $|e| > lastDistance$ **then**
    $rank := currentIndex$
  $lastDistance := |e|$
  $|e| := 2^{rank}$
  $currentIndex := currentIndex + 1$
return `hungarianAlg`$(edgesSorted)$

Another algorithm for computing MMDR with an $O(n^4)$ time complexity was discovered by Sokkalingam and Aneja [5]. We implemented this algorithm and analyze its performance with other algorithms in Section 4.

Min. Max. Dist. + Min. Sum Dist² (MMD+MSD²)

The Minimum Maximal Distance + Minimum Sum Distance² (MMD+MSD²) role assignment function minimizes the maximum distance any agent has to travel (but not recursively as done by MMDR), after which it minimizes the sum of distances squared that all agents travel. We show MMD+MSD² is a *CM valid* assignment, but not dynamically consistent, in an online appendix.[1]

Algorithm 2 implements MMD+MSD² by first finding a perfect matching with the smallest maximum edge (line 1) which is computed by adding edges to the graph in increasing order of length until a perfect matching is found through a breadth-first search of augmenting paths using the Ford-Fulkerson algorithm [1]. Then the set of all edges with length less than or equal to the longest edge in our perfect matching (line 2), and with edge weights equal to their distances squared, is used as input to the Hungarian algorithm (line 3).

Algorithm 2 MMD+MSD² $O(n^3)$ Polynomial Time Impl.

Input:
$Agents := \{a_1, ..., a_n\}$; $Positions := \{p_1, ..., p_n\}$
$Edges := \{\overrightarrow{a_1 p_1}, \overrightarrow{a_1 p_2}, ..., \overrightarrow{a_n p_n}\}$; $|\overrightarrow{a_i p_j}| :=$ `euclideanDist`$(a_i, p_j)^2$
1: $longestEdge :=$ `getMinimalMaxEdgeInPerfectMatching`$(Edges)$
2: $minimalEdges := e \in Edges$, s.t. $|e| \leq |longestEdge|$
3: **return** `hungarianAlg`$(minimalEdges)$

The $O(n^3)$ time complexities of both the Hungarian and Ford-Fulkerson algorithms dominate Algorithm 2 and thus its time complexity is $O(n^3)$. The breadth-first search of Ford-Fulkerson gives a space complexity of $O(n^2)$.

4. ALGORITHM ANALYSIS & SUMMARY

Table 1: Time and space complexities of role assignment algorithms and average running time in milliseconds for different values of n as measured on an Intel(R) Xeon(R) CPU E31270 @ 3.40GHz.

	Complexity		Value of n					
Algorithm	Time	Space	10	20	100	300	1000	10000
MMD+MSD²	n^3	n^2	0.016	0.062	1.82	21.2	351.3	115006
MMDR $O(n^4)$	n^4	n^2	0.049	0.262	17.95	403.0	14483	—
MMDR $O(n^5)$	n^5	n^3	0.022	0.214	306.4	40502	—	—
MMDR dyna	$n^2 2^{(n-1)}$	$n\binom{n}{n/2}$	0.555	2040	—	—	—	—
brute force	$n!n$	n	317.5	—	—	—	—	—

This abstract discusses SCRAM, a dynamic role assignment system for mobile agents. SCRAM minimizes the makespan for agents to reach target positions while avoiding collisions among agents.

To evaluate role assignment algorithms, we generated mapping scenarios for n agents and targets. Both agents and targets were assigned random integer value positions on a 2D grid with sides of length n^2. Table 1 shows the average runtime of the algorithms. As role assignment algorithms run in polynomial time, SCRAM scales to thousands of agents.

5. REFERENCES

[1] D. Ford and D. R. Fulkerson. *Flows in networks.* Princeton university press, 2010.

[2] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[3] P. MacAlpine, F. Barrera, and P. Stone. Positioning to win: A dynamic role assignment and formation positioning system. In *RoboCup-2012: Robot Soccer World Cup XVI*, Lec. Notes in Artificial Intel. 2013.

[4] D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774 – 793, 2007.

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