

A Decentralized Approach to the Art Gallery Problem

Robert N. Lass*, Michael J. Grauer*, Evan A. Sultanik* and William C. Regli*

Abstract

With the increasing interest in ubiquitous computing and mobile robotics, distributed algorithms for solving computational geometry problems in a decentralized fashion will grow in importance. This paper describes a mapping of discretized variants of the art gallery problem to a distributed constraint optimization problem. A testbed using several complete, optimal, distributed algorithms is presented for solving the problem in a robust and fault-tolerant manner, and without the need for centralization. Finally, some pruning techniques are analyzed for their effect on the size of the problem state space.

1 Introduction

In the traditional Art Gallery Problem (AGP), the goal is to find the minimum number of security officers or cameras that can guard a fixed number of paintings hung on the walls of a polygonal gallery. In the variants considered in this paper, the number of guards is fixed and the goal is to find positions for the guards within the polygon such that the number of guarded paintings is maximized. Specifically, the focus is on coordinating robotic guards that communicate with each other to find a globally optimal solution in a decentralized manner. This problem was proposed in [2], however their solution (and the traditional problem) makes assumptions that are not realistic for robotic guards with limited vision and angle of view.

A large class of problems that are inherently spatially or logically distributed can be modeled naturally as Distributed Constraint OPTimization (DCOP) problems. DCOP is an active area of the constraint programming community and a number of algorithms have been developed to solve DCOP problems [3, 4]. Formulating problems as DCOP affords the opportunity to take advantage of the variety of existing algorithms and techniques for reasoning about and solving DCOP problems. The robots can be placed in an arbitrary gallery and determine where to stand without any human intervention, supervising computing device or central point of failure. Additionally, by solving the problem as a DCOP, robots can dynamically re-solve for the (new) optimal solution if one or more of the robots fails.

In the remainder of this paper we first provide a scenario motivating the need to solve computational geometry problems in a robust, fault-tolerant and decentralized fashion. Next, we give background on DCOP and the mapping of the CG problems we examined to DCOP. We then describe a testbed that we created to compare different techniques for

solving these problems on live networks using the Adopt [3], DPOP [4], and other algorithms. All of these algorithms are guaranteed to produce an optimal solution. Finally, we present an analysis of different state-space pruning techniques and conclude with directions for future work.

2 Scenario

The problem domain is the well-known AGP, modified to have more realistic restrictions for actual agents (*e.g.* mobile robots). Each robot has a configurable angle of vision (360 degrees as in the traditional AGP, or some smaller value) and a limited range of vision (∞ as in the traditional AGP or some smaller value). The robots each have a map showing the layout of the gallery and the placement of paintings. A simplifying assumption is that the robots have error-free localization sensors that allow them to determine their position in the XY plane of the art gallery and their bearing, with 0 degrees pointing up. The interior space of the art gallery polygon is discretized to simplify the conversion to a DCOP. The number of guards and the number of paintings are configurable values. Each painting is located at a fixed position, and many paintings may be located in a disjoint position (non-overlapping) on any given wall. The art gallery itself is a randomly generated simple polygon [1].

Each robot has an agent that represents the robot in the DCOP algorithm, communicating with other agents to determine the optimal solution, with communication assumed to be error-free. A solution is a configuration value $\langle X, Y, \theta \rangle$ in $\mathbb{N} \times \mathbb{N} \times \mathbb{Z}_{360}$ for that robot, which is a position in the XY -plane (one of the grid points of the interior space of the art gallery), and a heading in degrees (with 0 facing up). The state space is the set of all possible configurations.

Scenario 1 The agents do not start with any particular configuration value for their robot, and the objective is for each agent to find a configuration value that minimizes the number of paintings not under observation. The state space is reduced by pruning any configurations from the state space from which no paintings are visible. Redundant configurations are also removed from the state space. Redundant configurations are those that contain either the same set or a strict subset of visible paintings as another. This leaves one representative configuration in the state space for each possible set of paintings.

Scenario 2 This is identical to scenario one, except that each robot starts in a unique discrete position, and the cost of the total distance traveled by all agents from their starting positions to their final positions is minimized. In this scenario, distance is only minimized for translations in the XY plane, rotations will not be minimized. As the interior space of the art gallery is discretized, the discrete grid

*Drexel University, Department of Computer Science, College of Engineering, 3141 Chestnut Street, Philadelphia, PA 19104, Email: {urlass, mjpg69, eas28, regli}@cs.drexel.edu

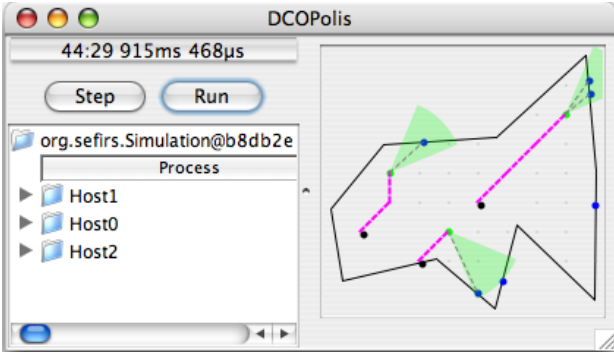


Figure 1: A solved AGP, in DCOPolis.

points are viewed as nodes of a graph, edges are created between neighboring (horizontal, vertical, or diagonal), visible nodes of weight equal to the Euclidean distance between the nodes, and the shortest path between all nodes is calculated using the Floyd-Warshall all pairs shortest paths algorithm. This scenario’s only state space reduction is to remove configurations from which no paintings are visible.

3 DCOP

A formal definition of a DCOP requires too much space for this format, and the reader is referred to [3], which fully and formally defines DCOP. Informally, the four main components of a DCOP are variables, domains, agents and constraints. Each *agent* has a set of *variables*, to which it must assign *values*. Each variable has an associated *domain*, which is the set of all possible value assignments to the variable. *Constraints* are a set of functions that specify the cost of any set of partial variable assignments. Finally, each agent is assigned one or more variables for which it is responsible for value assignment. DCOP algorithms work by exchanging messages between agents, who give each other just enough information to allow each agent to make a globally optimal variable assignment.

3.1 AGP to DCOP Mapping

Mapping the AGP to a DCOP is done as follows:

- **Agents:** Each robot has a DCOP agent, $a_i \in A$.
- **Variables:** Each agent has a single variable, $v_i \in V$, representing where the robot stands.
- **Agent to variable mapping:** The function α maps variables to an agent, $\alpha : V \rightarrow A$
- **Domain:** The domain for each variable, $d_i \in D$, is all the positions in which a robot can stand.
- **Constraints:** The cost of each variable assignment is the sum of all paintings that are not in view of a guard. In the variant of the problem taking distance into account, the distance traveled is also a cost. The function¹

$$f : \bigcup_{S \in \mathcal{P}(V)} \prod_{v_i \in S} (\{v_i\} \times D_i) \rightarrow \mathbb{N} \cup \{\infty\}$$

maps every possible variable assignment to a cost.

4 Results and Conclusions

We implemented both of the scenarios in section 2 in DCOPolis [5]. A screenshot from the program can be seen

¹“ $\mathcal{P}(V)$ ” denotes the power set of V ; “ \prod ” is the Cartesian product.

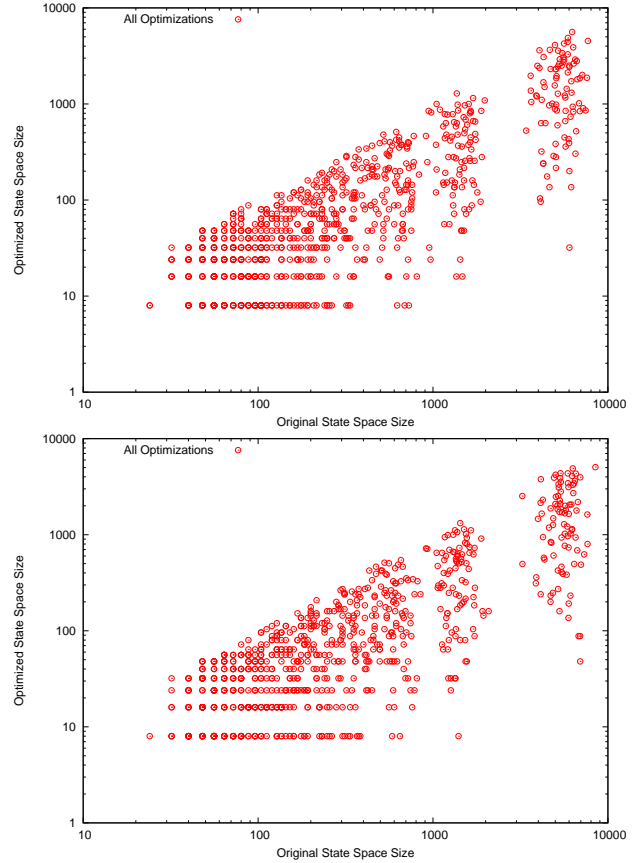


Figure 2: Reduction in scenarios 1 (above) and 2 (below). Please note: we are using a log scale!

in figure 1. As can be seen in the graphs in figures 2, the pruning techniques reduce the state space size by an average of approximately 30% in both scenarios.

References

- [1] T. Auer and M. Held. Heuristics for the generation of random polygons. In *Proceedings of the 8th Canadian Conference on Computer Geometry*, pages 38–44. Carleton University Press, August 1996.
- [2] A. Ganguli, J. Cortes, and F. Bullo. Distributed deployment of asynchronous guards in art galleries. In *American Control Conference*, pages 1416–1421, Minneapolis, MN, June 2006.
- [3] P.J. Modi, W.M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 161–168, New York, NY, USA, 2003. ACM Press.
- [4] A. Petcu and B. Faltings. A distributed, complete method for multi-agent constraint optimization. In *The Fifth International Workshop on Distributed Constraint Reasoning*, Toronto, Canada, September 2004.
- [5] E.A. Sultanik, R.N. Lass, and W.C. Regli. DCOPolis: A framework for simulating and deploying distributed constraint optimization algorithms. In *The Ninth International Workshop on Distributed Constraint Reasoning Workshop*, Providence, RI, September 2007.