Swarm-Based Path Creation in Dynamic Environments for Search and Rescue

William K. Richard
BBN Technologies
10 Moulton St., Cambridge, MA 02138
william.richard.no.s@gmail.com

Stephen M. Majercik
Bowdoin College
8650 College Station, Brunswick, ME 04011
smajerci@bowdoin.edu

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems, coherence and coordination*

Keywords

Swarm intelligence algorithms, search and rescue

1. INTRODUCTION

Swarm intelligence is a natural phenomenon in which complex behavior emerges from the collective activities of a large number of simple individuals. Swarms are adaptable to changes in their environments, robust to the loss of swarm members, and scalable. Swarm algorithms attempt to capture these desirable qualities by distributing computation among simple agents that interact and sense only locally, the desired solution or behavior emerging from their actions. We introduce Swarm Search-and-Rescue (SWARM-SR), a swarm algorithm that locates survivors in a dynamic, hazardous environment, finds high quality paths from an arbitrary starting location to these survivors, physically marks the paths so that they can be traversed by human rescuers, and updates the paths, creating new paths if necessary, to reflect changing conditions. We present results of successful initial tests of SWARM-SR in a simulated disaster scenario.

2. THE SWARM-SR ALGORITHM

The agents in our simulations are partially based on the AscTec Pelican, a small UAV that can hover and has sufficient processing power (1.6 GHz) and memory (1 GB RAM) for complex sensor analysis and movement control. We assume they can 1) communicate with each other over a limited range, but with no limit on bandwidth, 2) locate themselves, 3) identify hazards and survivors, and 4) identify the direction of a survivor shout within a specified range.

Our search and rescue environment is partitioned into four types of *zones*: a single *Base Zone* from which all agents start and in which all survivor paths must end, *Safe Zones*, which are safe for agents and rescuers to travel through, *Danger Zones*, which agents avoid, but through which rescuers can pass, and *Fire Zones*, which agents strongly avoid and through which rescuers cannot pass. Conditions in a given zone may change during the simulation; zone boundaries do not. Survivors, distributed randomly, do not move.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

There are three types of agents: *Normal Explorers* (NE), *Bold Explorers* (BE), which move more aggressively into Danger and Fire Zones, and *Path Markers* (PM). All agents leave the Base Zone as NEs, searching for survivors. When an agent senses a survivor, it claims that survivor, notifies the swarm of the survivor's location, and initiates a path creation process. Agents near the path become PMs, physically marking the path by evenly distributing themselves along the path. Finally, agents that have claimed survivors periodically initiate a path reconstruction process to search for better paths. During each time step, an agent's actions proceed in four phases for an Explorer: *movement*, *survivor management*, *messaging*, and *role switching*, and in three phases for a PM: *movement*, *messaging*, and *role switching*.

Movement Phase: An Explorer that has claimed a survivor moves toward that survivor. An Explorer that has not claimed a survivor moves based on the sum of a cohesion force with each agent in its neighborhood (a circle of a specified radius centered on the Explorer) and separation forces from each agent in its neighborhood and from obstacles and Danger and Fire Zones. The cohesion force is a scaled vector to the average location of all neighbors. The separation force is similar to the force used for obstacle avoidance in [3] and is specified by four parameters, which differ depending on the entity being separated from: C, a scaling constant, r_{min} and r_{max}, which specify the distance interval over which the force is not a constant, and s, an exponent that dictates the shape of the force between r_{min} and r_{max}. $F_{sep}(r)$, the magnitude of the separation force at a distance of r, is 0.0 if $r > r_{max}$, C if $r < r_{min}$, and computed as follows otherwise:

$$F_{sep}(r) = C * \frac{r^s - r^s_{max}}{r^s_{min} - r^s_{max}} \quad (1)$$

A PM moves toward a path or, if already on a path, calculates its motion based on the separation forces exerted by its two closest PM neighbors. These forces are calculated using Equation 1 with parameters that, empirically, result in approximately even spacing between PMs.

Survivor Management Phase: If an Explorer that has not yet claimed a survivor finds a survivor, it claims that survivor by recording its location, adding it to its known survivor list, and broadcasting a claim message. An Explorer that has claimed a survivor periodically initiates a path creation process by broadcasting a path creation request.

Messaging Phase: There are four types of messages: 1) Location, 2) Found-Survivor, 3) Claim-Survivor, and 4) Path-Creation. Each agent broadcasts a Location message when it moves to a new location. An Explorer broadcasts a Found-Survivor message when it finds a survivor, and stores

and rebroadcasts new Found-Survivor messages it receives. An Explorer broadcasts a Claim-Survivor message when it claims a survivor (a leader election algorithm ensures that only one Explorer claims each survivor) and stores and rebroadcasts new Claim-Survivor messages it receives.

A Path-Creation message consists of a list of distinct paths, each of which is a collection of points starting at a survivor's location, each point annotated with the ID of the agent that added that point and the condition of that point's zone. A Path-Creation message initiating the path creation process consists of a single path containing the survivor's location as its only point. A path is *complete* if the final point of the path is in the Base Zone; otherwise, it is *partial*. Each type of zone is assigned a weight—dangerous zones have higher weights—and $L(P)$, the length of a path P is:

$$L(P) = \Big[\sum_{i,\ j \text{ adjacent in } P} d(i,j) \max(w(i), w(j)) \Big] + h(P)$$

where $d(i, j)$ is the distance between points i and j, $w(i)$ is the weight of the zone containing i, and $h(P)$ (similar to the heuristic value in A* search) is the unweighted distance from the last point in the path to a fixed Base Zone point for partial paths, and 0 for complete paths.

An agent maintains a list of the best paths it knows, and processes the paths it receives in Path-Creation messages with reference to this list in order to create a list of paths to rebroadcast (R-List). It processes complete paths first, then partial paths in order of ascending length. A new complete path P (there is no complete path to the same survivor stored in the agent's list) is stored and added to the R-List. If a complete path P has the same survivor as a stored path P', the shorter path replaces P' and is added to the R-List.

A partial path is discarded if the agent 1) is too close to the last point in the path (to avoid extremely short path segments), or 2) has already contributed to this path (to avoid loops), or 3) has a complete path to that survivor that is shorter than the partial path. In the last case, it adds the stored complete path to the R-List. If the partial path is not discarded, the agent adds its own location to the partial path and adds it to the R-List. If the added location is in the Base Zone, it marks the path as complete and reprocesses it as a complete path.

Role Switching Phase: Each agent maintains a probability that it will become a PM (p_{PM}), a probability that it will become an NE (p_{NE}), and a probability that it will become a BE (p_{BE}). These are updated at each time step based on the agent's current role and observed conditions, and the agent uses these to stochastically decide which role to assume at the next time step. For an Explorer, the absence of nearby paths decreases p_{PM}; poorly covered nearby paths increase p_{PM}. In the absence of nearby dangerous zones, p_{NE} and p_{BE} do not change; otherwise, these probabilities are updated to make it more likely that the agent will become the type of Explorer in the minority in the dangerous zone. For a PM, the probabilities are updated so that it is more likely to remain a PM if local path coverage is poor; otherwise, so that it is more likely to become some type of Explorer.

3. EXPERIMENTS

We ran experiments for 30 minutes of simulated time, on a 600×600 meter^2 area with 200 zones, each zone changing its safety status stochastically (every five seconds, on average). We varied the number of survivors from one to five (placed randomly) and the number of agents from 20 to 200 (at intervals of 20), and ran five experiments for each survivor-agent combination. We were not able to run tests with more than five survivors due to insufficient memory; actual UAVs, however, would have ample memory for a single agent's computations, so the number of survivors SWARM-SR can handle would not be constrained by lack of memory. Details of parameter settings are available in the full paper.

The overall performance measure (OPM) was defined as FSF/(ROPL$\times$SPCS), where FSF is the fraction of survivors found, ROPL is the ratio of the length of the best path found to the optimal length, and SPCS is a scaled path coverage score based on the average number of PMs that maintain an acceptable distance between themselves and their PM neighbors. OPM ranges from 0.0 (no survivors found) to 1.0 (optimal, well-covered paths to all survivors).

In the 5 survivors, 200 agents scenario, the swarm achieved an average OPM of approximately 0.8, resulting from an FSF of 1.0 (all survivors found), an average ROPL of approximately 1.25 (average path length 25% higher than optimal), and an SPCS of 1.0 (all paths well-covered). This state was achieved in approximately three minutes, which would allow human rescuers to respond very quickly, and the swarm was able to maintain this high performance level in the face of changing conditions for the entire simulation. Note that reducing the number of agents by 50% to 100 decreased the overall performance only by about 0.1 to approximately 0.7.

4. RELATED AND FUTURE WORK

Atyabi et al. describe *Area Extended PSO* for search and rescue in dynamic environments [1]. Their scenario is more realistic (survivors can move and die), but there are no hazards, agents know the number of survivors, and no attempt is made to find paths. Ducatelle et al. describe a system that uses swarms of foot-bots and eye-bots, which have fixed roles, to navigate between source and target locations [2]. Their system is intended for static, indoor environments; ours is intended for dynamic, outdoor environments. More related work is described in the full paper.

With regard to future work, changes in agent behaviors could improve performance, e.g. an agent marking a path could broadcast updates about the current path's quality, making future path re-creation faster. SWARM-SR should be tested in more realistic scenarios: e.g. more survivors, survivors that can move and die, and agents that can be damaged or destroyed. Finally, the performance of SWARM-SR needs to be evaluated under a broader range of assumptions and parameters in order to better assess its robustness.

5. REFERENCES

[1] A. Atyabi, S. Phon-Amnuaisuk, and C. K. Ho. Applying area extension PSO in robotic swarm. *Journal of Intelligent and Robotic Systems*, 58(3-4):253–285, Oct. 2009.

[2] F. Ducatelle, G. A. D. Caro, C. Pinciroli, and L. M. Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96, 2011.

[3] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, and A. Oustaloup. Consideration of obstacle danger level in path planning using A* and Fast-Marching optimisation : comparative study. *Signal Processing*, 83:2387–2396, 2003.