Algorithms for Distributed Exploration

Thomas Walker, Daniel Kudenko
Department of Computer Science
University of York
United Kingdom

Malcolm Strens
Future Systems Technology Division
QinetiQ
United Kingdom

Abstract.
In this paper we propose algorithms for a set of problems where a distributed team of agents tries to compile a global map of the environment from local observations. We focus on two approaches: one based on behavioural agent technology where agents are pulled (or repelled) by various forces, and another where agents follow a approximate planning approach that is based on dynamic programming. We study these approaches under different conditions, such as different types of environments, varying sensor and communication ranges, and the availability of prior knowledge of the map. The results show that in most cases the simpler behavioural agent teams perform at least as well, if not better, than the teams based on approximate planning and dynamic programming.

The research has not only practical implications for distributed exploration tasks, but also for analogous distributed search or optimisation problems.

1 Introduction

Consider the following scenario. A number of agents (e.g., robots) have the task to create a map of a region, with limited sensory range, and possibly limited communication abilities. Such scenarios are of practical importance especially in multi-robot exploration domains, such as search and rescue.

Clearly, the exploration task can be performed more efficiently when more than one agent is available. But what is the best distributed exploration algorithm to employ? Specifically, how can the availability of a *team* of agents be best exploited to reduce exploration time? Search algorithms that generate optimal solutions, such as A^* (and its distributed variants [8]), are only feasible for the exploration of relatively small regions. In the case of larger regions different approaches are necessary.

To understand the complexity of this problem, we can relate it to the formal models used in reinforcement learning and planning. Even in the case that the agents have full communication, it is an intractable planning problem: a partially observable Markov Decision problem (POMDP) in which the *observable* state consists of the explored region of the map and the *hidden* state is the unexplored area. The optimal exploration policy can, in theory, be obtained by forming a large MDP over the belief states of the POMDP and solving it by dynamic programming [5]. Unfortunately the number of belief states is infinite and so approximate methods are needed. When agents have limited communication, each one then has an even more difficult decision problem: the hidden state consists not only of the unexplored map area, but also the beliefs and plans and other agents. While parts of this hidden state could be ignored in the planning process, this may severely limit performance. For example, if agents do not account for each other's plans, effective cooperative behaviour will not result.

Many related multi-agent approaches, such as [7] and [3], are assuming that agents are always in communication range. In contrast, we consider algorithms that do not make such an assumption. In other related work, Balch and Arkin [1] studied the impact of communationon distributed exploration, but the exploration technique used was very basic.

In this paper we investigate different distributed exploration algorithms and evaluate their performance under different conditions such as varying sensor and communication ranges, and the availability of prior knowledge of the map. We focus our study on two types of approaches: one based on behavioural agent technology where agents are pulled (or repelled) by various forces, and another where agents follow a approximate planning approach that is based on dynamic programming [2].

Our results show that heuristic behavioural methods can do as well if not better than approximate planning approaches, unless the planning uses a very sophisticated model, i.e., avoids assumptions or approximations.

The paper is structured as follows. We first describe the two-dimensional maze environment that we used to evaluate our approaches. We then discuss the individual algorithms and variations. This is followed by a presentation of the empirical evaluation and results. Finally, we conclude the paper with a summary and an outlook to future work.

2 The Simulator

We chose a two-dimensional grid world of size 50x50 as the environment in which to evaluate the various exploration approaches (see Figure 1 for a small example). Each cell in the grid world is either clear or filled (i.e., a wall cell). The agents are able to move horizontally or vertically to an adjacent clear cell, but are not permitted to enter wall cells.

The agents act simultaneously, but in a synchronized fashion. Each turn consists of three phases during which each agent moves, scans the surroundings, and possibly communicates with other agents. When agents scan the surroundings, they receive information on the state of the four cells adjacent to them. Communication is mostly used to exchange information on the maze (i.e., exchange partial maps of the maze), but can also be used to communicate move intentions. Agents are able to communicate virtually simultaneously.

An agent's beliefs are represented by a grid representing the map where each cell could take one of three values (unknown, clear or filled). An agent also has a list containing the positions of other agents (if known) and possibly their plans.

3 Exploration Algorithms

As stated before, precise search and planning approaches break down in the distributed exploration task as soon as the maze reaches a reasonable size. This section presents various heuristic algorithms for distributed exploration that are feasible also for larger mazes: a very straight-forward greedy technique, an approach based on behavioural agents, and an approach based on approximate dynamic programming.

3.1 Simple Agent

This agent is very simple in that it moves towards the closest cell it does not yet know about. When it finds out about this cell (either by scanning it or another agent communicating with it) it will change to move towards the next closest unknown cell.

3.2 Behavioural Agent

The behavioural agent is based on a schema architecture consisting of two behaviours. One attracts the agent towards unexplored areas while the other tries to keep the agents apart, which causes them to scan different areas. These behaviours are determined by assigning heuristic values to every cell (indexed by n) based on whether it is known or unknown, and (if known) whether it is occupied by another agent. Known cells are assigned a large negative value (penalty) if occupied by another agent, and zero otherwise; we will call this value $p(n)$. An *exploration bonus* [6] term $b(n)$ is also added, taking a small positive value for unknown cells (and zero for known cells). These values are propagated back towards the agent to obtain a heuristic exploration value $V(n)$ of each cell n.

$$V(n) \leftarrow b(n) + p(n) + \gamma * \sum_{\{n' \in \text{neighbours of } n\}} V(n') \tag{1}$$

where γ is a set decay rate. The computation of this heuristic exploration value function starts with the cells furthest away and only one iteration is carried out.

This results in a system where the agents attempt to spread themselves across the map while also moving towards unexplored areas. Unfortunately this system can result in deadlock since the repulsion between agents can cause them to become trapped in dead ends they have already scanned. To get around this dead ends are marked with a value of $-\infty$ after the above propagation function has been applied but before the agent moves. This prevents an agent from moving into areas where it is not possible to acquire new data. This fix removed most of the problems resulting in deadlock situations. However, it was still possible in very specific circumstances for the agents to get themselves locked into loops where they moved in circles relative to each other. To prevent this situation from occurring one of the agents was set to ignore other agents. This results in a system where repulsion cannot cause deadlock, since at least one agent will always be unaffected. Also, since this agent will continue moving its repulsion force should influence other agents stuck in loops and cause them to move on.

3.3 Approximate planning (AP)

The approximate planning (AP) agent propagates values of squares using a more theoretically justifiable approach (dynamic programming) that is known to yield optimal polices *if all assumptions are met*. For each cell it associates an *immediate reward* $r(n)$ equal to the number of additional cells, that would become known if the agent could move instantly to the new location[1] In calculating this reward we assume that all unknown cells will be empty (so the sensors will scan to their maximum range except when obstructed by a known filled cell). These rewards are then propagated by repeated application of the Bellman backup operator:

$$V(n) \leftarrow r(n) + \gamma \max_{\{n' \in \text{neighbours of } n\}} V(n') \tag{2}$$

where γ is a set decay rate. Regardless of initialisation of $V(n)$, this *dynamic programming value iteration* process converges to the expected discounted return for each cell, where the exploration bonus is interpreted as the immediate reward in each cell.

Each agent plans its next move by greedily choosing a path through cells with greatest value. The agents plan ahead for 10 moves.

The AP agent can be extended to take account of other agents' intentions by using *their* plans to affect the values *it* assigns to cells. When one agent communicates its intention to scan a cell, the value to a different agent of scanning that cell is decreased. (It isn't typically set to zero because the agents' plans may change or the cell may actually be obscured, by an unseen wall.) By using this system, agents within communication range should be able to form joint plans that avoid doing duplicate work.

4 Evaluation Scenarios

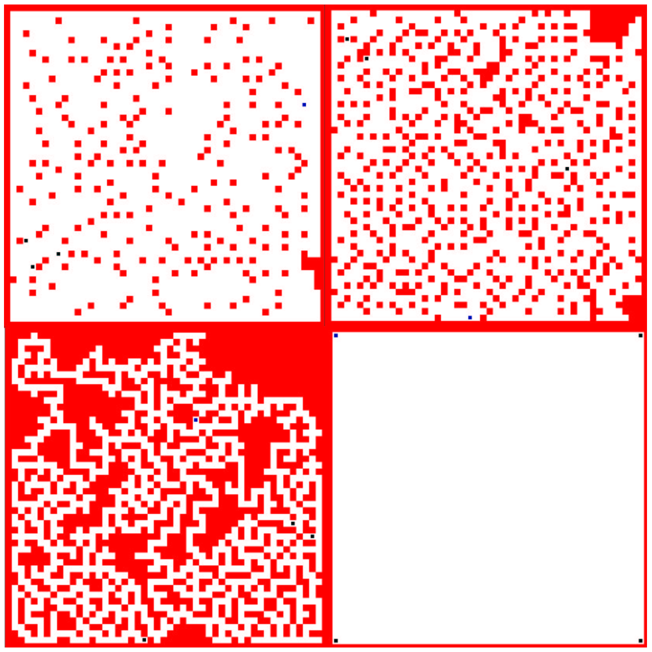

Figure 1. Different types of maps from left to right, Spacious, Medium, Narrow, Empty

We have evaluated the above exploration algorithms on 3 types of maze that have been randomly generated: spacious, medium, and

[1] Note that this may be an overestimate of the real "value of information" for visiting that cell, because some of the additional cells would have become known in moving to the new location.

narrow. The main difference between these mazes is the ratio of clear versus wall cells. We also chose a number of manually created mazes to cover some extreme cases.

The first type of map ("spacious") has relatively few walls and plenty of empty space. The second type ("medium") has more walls but still quite a lot of empty space. The third type ("narrow") are maps with long winding walls and corridors. There are also no loops in the map so agents must backtrack to go down different branches. The extreme cases consisted of a map with nothing but a wall surrounding the edges, a grid shaped map and a map with a single circuit around the edge.

In order to perform the testing 2 sets of maps were generated. Each set consisted of ten 50x50 maps of each type with starting positions of agents fixed. The first set was used to determine how the parameters of the agents should be set. The second set was used for testing. The parameters were set experimentally by using the ones that minimised the average number of turns taken per map. The finishing criterion for each run was set as the point where each cell on the map that could be scanned had been scanned by at least one agent.

Each maze was explored by teams of four agents. Each team was homogeneous, i.e., consisted of agents of one type. Each agent started the exploration in one of the four corners on the maze. The performance of the agent teams was measured in terms of the average proportion of the mazes covered in a given number of steps.

The agents were tested over a number of different communication and sensor ranges. Also, both the behavioural agent and the approximate planner were tested with and without prior knowledge of the maze structure (the agents still had to explore all cells in both cases). In both agents the propagation functions were adapted so that values would not propagate through wall cells that the agent was aware of due to prior knowledge. In the case of the approximate planning agent the evaluation function was adapted to take account of the fact that the agent will now know exactly how many cells will be scanned when an agent is in a particular space since it knows when a sensor is going to hit a wall.

5 Evaluation Results

5.1 Results on Spacious Mazes

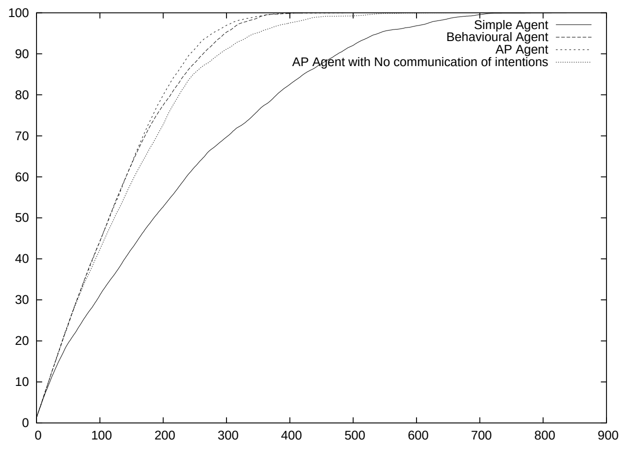

Figure 2. Average proportion of spacious mazes covered as a function of moves

In the case of the spacious mazes there was no statistically significant difference (using standard error) in the performance of behavioural agent teams and the AP agent teams in terms of the time taken to scan the complete map. Also there was no significant difference in performance between agents using prior knowledge of the maze and agents having no such knowledge. As expected, the simple agent performs drastically worse from very early on. Its performance also varies significantly due to a "sticking" effect where two agents on the same cell that can communicate will effectively become stuck together since they will both have the same belief about the world and hence both make the same decisions.

The AP agent without communication of intentions performs significantly worse for communication ranges of about 5 upwards, partly because it also suffers from this sticking effect.

5.2 Results on Medium Mazes

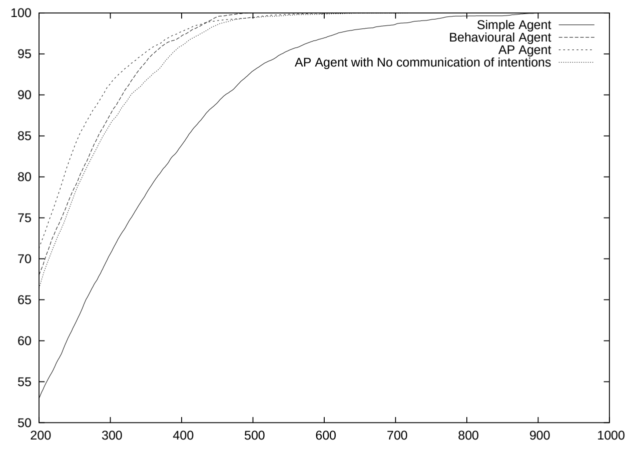

Figure 3. Average proportion of medium mazes covered as a function of moves from move 200 (below this point they are roughly the same except for the Simple agent which is noticeably worse from about move 50 onwards)

In the case of the medium mazes the team of simple agents was clearly outperformed across all communication ranges by all other teams. Again there was no significant difference between the performance of the behavioural agents and the AP agents in the time taken to scan the complete maze. Also, prior knowledge of the maze does not seem to play a significant role, except for the one AP agent case above.

5.3 Results on Narrow Mazes

Interestingly, for the Narrow Mazes the behavioural agent team performed significantly better than the AP agents. From Figure 4 it can be seen that at times the behavioural team has as much as a five percent lead over the other teams in terms of the amount of map covered as a function of time. Also, there is no significant difference between the performance of the simple agent team and the AP agent team with no communication of intentions. In this case the use of prior knowledge has a significant impact on the performance of the AP agent team. This is probably due to the fact that normally the AP agents assume that all unscanned cells are empty which in the case of this type of maze is probably not true. By using prior knowledge the agent is

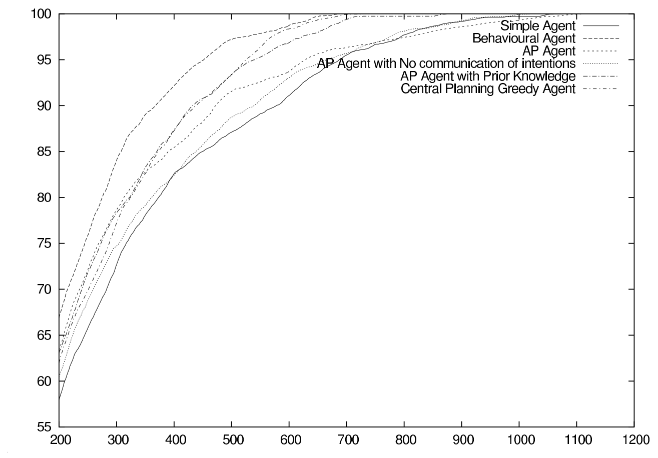

Figure 4. Average proportion of narrow mazes covered as a function of moves starting from move 200 (below this point the graphs are roughly the same)

able to accurately calculate the exploration bonuses for these sorts of maps.

5.4 Results on Extreme Cases

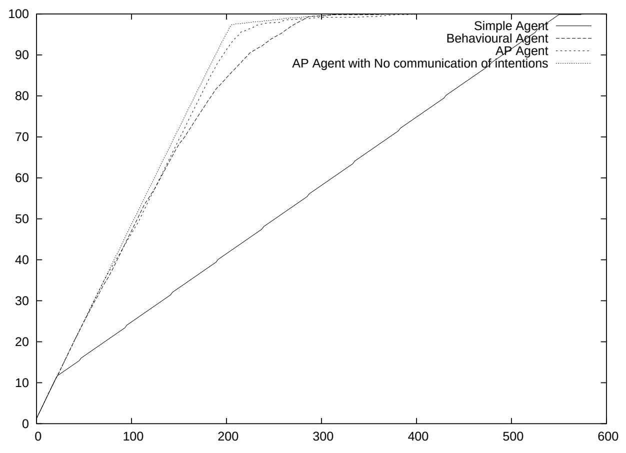

Figure 5. Proportion of Empty mazes covered as a function of moves

In the case of a completely empty map (Figure 5) the AP team clearly performed better than the other teams of agents. In fact the AP's performance was not much behind the lower bound for the map (calculated by the fact that it is not possible for a team of agents to scan more than 12 cells per turn after the first) up until about time 200 where its performance started to tail off. The reason the behavioural team falls behind may be due to the fact that since one of the agents ignores other agents symmetry is broken resulting in lower performance. The AP team also has the advantage that in nearly all cases their assumption about their sensors will hold since the only filled cells are around the edge.

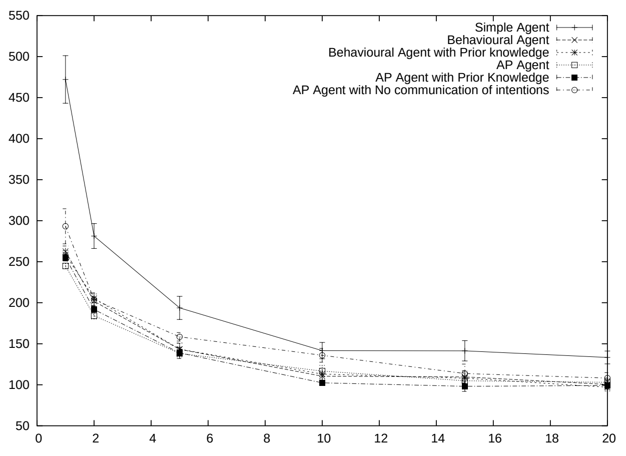

Figure 6. Average time taken to cover 90% of a Spacious map as a function of Sensor Range

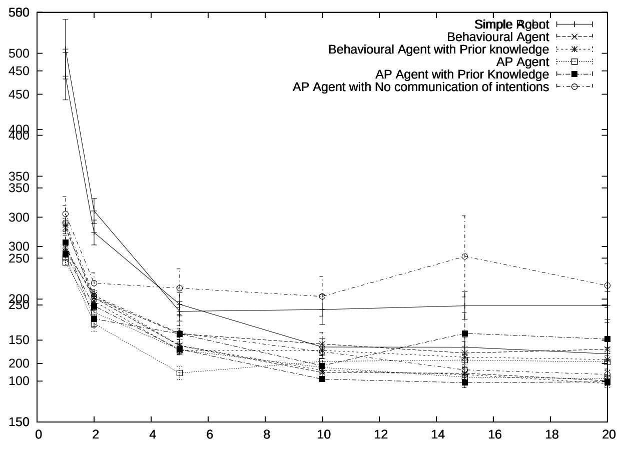

Figure 7. Average time taken to cover 90% of a Medium map as a function of Sensor Range

5.5 The Effect of Sensor Range

As could be expected, increasing sensor range improved performance for all agents as shown in figures 6 and 7. In the cases of the Medium Mazes and the Narrow Mazes typically the agents performance failed to improve when the sensor range was above about 5 since there will be so few points on the map where the agents will use their full sensor range.

6 The Effect of Communication Range

From Figure 8 it can be seen that Communication range has a significant impact up to 10 cells at which point it quickly decreases. This trend holds for the other two types of maze as well. This is probably due to the agents acting as "relays" between each other to allow one agents belief to be communicated to other agents that are further away than the communications range. Once this reaches about $1/5$ of the width of the map the agents will be within communications range of each other regularly so each agents belief about the world will be almost the same. By the time the range reaches $1/2$ of the

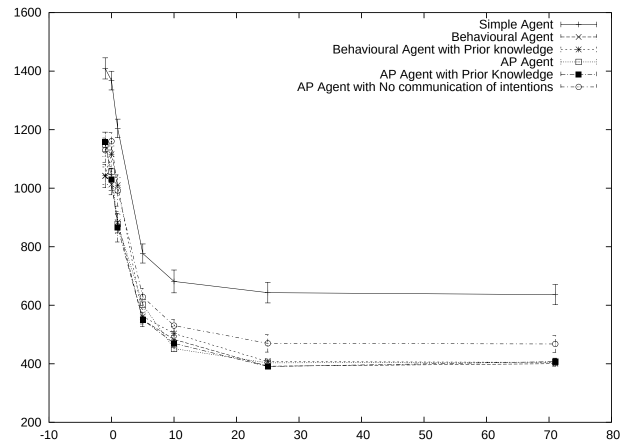

Figure 8. Average time taken to scan 100% of a Spacious map as a function of Communication Range

map it is almost the same as if it were infinite.

7 Discussion

Distributed exploration is an example of a Partially Observable Markov Decision Problem (POMDP). The partially observability arises because certain state information (the map data and other agents' states) is hidden from each decision-making agent. It is, in theory, possible to obtain the optimal policy for a POMDP [5] by transforming it into an MDP over *belief* states, then solving the MDP by dynamic programming [2]. In practice, the size of the belief state-space is very large or infinite, and so approximate solution methods are required, even if there is no multi-agent aspect to the problem.

Therefore *approximate* planning in which we make various simplifying assumptions, to make the planning process tractable, must be considered. In this case, we have made several simplifying assumptions: we have not planned over every possible future observation, instead assuming that the planned path will be passable. Similarly, our planning ignores possible interference from other agents that are beyond communication range (although it does form joint plans with any agents that are in range, through communication of intentions). Furthermore, the planning mechanism is greedy (it does not consider all options) and the time horizon is limited (10 steps). Together these assumptions (and several others) led to acceptable computation time for the planning mechanism, but affect optimality of the resulting behaviour. Hence the possibility arises that a heuristic method could outperform approximate planning.

The experimental results did indeed show better performance from the simpler (and faster) heuristic approach. While the ideal solution remains the exact optimal policy that would be obtained by planning over belief states, it is worth asking the questions: (i) How many of the planning assumptions/approximations would have to be reversed in order to exceed the heuristic performance? (ii) If optimal planning is intractable for a "toy-problem" like this, how feasible it is to obtain near-optimal approximate planning in a larger domain? The answers to these questions will be application-specific, but it may be that in many complex, partially observable domains, reactive behaviours (whether heuristic or learnt) are the *only* feasible approach for multi-agent cooperation.

It is also worth noting that the relative performance of AP varied significantly across the different types of scenario; this may indicate that some of the approximations made are less erroneous in some scenarios than in others.

From the results over communication ranges it can be seen that while to start with the increased communication range had a very significant impact in the case of all agents. However, the impact of this quickly tailed off to the point where being able to communicate over a fraction of the map was almost as good as being able to communicate over all of it.

8 Conclusion

In summary, we presented a number of heuristic techniques for distributed exploration and evaluated them on a wide range of scenarios and under varying conditions. The results show that in most cases the simpler behavioural agent teams perform at least as well, if not better, than the teams based on approximate planning and dynamic programming. The results can also be applied to distributed search tasks or optimisation problems that are analogous to exploration.

For multi-agent problems that have a large element of partial observability (about the environment state or other agents' beliefs) it may be impractical to implement an approximate planning solution: heuristics or learnt/optimised behaviours may be more appropriate. Nevertheless it is useful to approach the problem from the perspective of sequential decision theory (and the POMDP model) in order to reason about effective behaviour: for example even our heuristic behaviours keep track of a belief distribution (indicating known/unknown areas of the map).

The effective multi-agent exploration obtained with the heuristic method makes use of repulsion between nearby agents within sensor/communications range (this force could also be seen as defining a potential field). In non-physical environments (e.g. data mining) the same approach could be used if an appropriate distance measure is available. Therefore our exploration algorithm has properties in common with Tabu search [4].

REFERENCES

[1] Tucker Balch and Ronald C. Arkin, 'Communication in reactive multiagent robotic systems', *Auton. Robots*, **1**(1), 27–52, (1994).

[2] Richard E. Bellman, *Dynamic Programming.*, Princeton University Press., 1957.

[3] Wolfram Burgard, Mark Moors, and Frank Schneider. Collaborative exploration of unknown environments with teams of mobile robots.

[4] F. Glover and M. Laguna, *Tabu Search*, Kluwer, 1997.

[5] J. J. Martin, *Bayesian Decision problems and Markov Chains*, John Wiley, New York, 1967.

[6] R. Sutton, 'Integrated architectures for learning, planning and reacting based on approximating dynamic programming', in *Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA, (1990).

[7] Brian Yamauchi, 'Frontier-based exploration using multiple robots', in *Proceedings of the second international conference on Autonomous agents*, pp. 47–53. ACM Press, (1998).

[8] Makoto Yokoo and Toru Ishida, 'Search algorithms for agents', in *Multi–Agent Systems*, ed., Gerhard Weiß, 165–199, MIT Press, Cambridge, Massachusetts, (1999).