1. Introduction

We discuss general issues pertaining to theSprite Planning Research Challenge (SPRC) hosted by the University of York and QinetiQ in which a number of Unmanned Aerial Vehicles (UAVs) search for survivors of an earthquake. We also discuss our proposed solution.

2. General Discussion

2.1. One Scenario is a Class of Problems

The optimal solution (possibly even type of solution) to the SPRC appears to be very sensitive to exactly how it is parameterised. For example, when there are many survivors the importance of sending a UAV to every observed trace is diminished and a simple non-revisiting search may be hard to improve upon. Conversely, if there are few survivors, each trace is of comparatively greater value. As the expected number of traces decreases the penalty for ignoring one in the hope of future success grows and coordination and planning may yield far greater improvements. Other important parameters include the duration of simulation, relative speed to the area and sensor range and the length of a timestep.

Because of this variability many solution methods might be successfully applied to different SPRC instances. For some parameterisations we might find an evolutionary approach is best, whereas others may best be tackled using reinforcement learning, and so on. Accordingly we argue it is appropriate to think of the un-parameterised SPRC as describing a class of problems, and consequently finding a high-quality solution procedure entails that the simulation’s parameters must be specified fully and precisely ahead of time. If not, the best we can do is to provide a range of solutions, and perhaps a meta-learning mechanism for selecting among them.

In a real-world sensor planning problem there are likely many parameters which cannot be known in advance. The best approach is to estimate parameters as well as possible, and have a range of solution methods applicable to different scenarios available. Any single method which provides reasonable performance for a wide range of parameters can be expected to be overtaken by highly specialised alternatives designed for a narrower parameter selection. Large regions of the parameter space may have very little structure for any algorithm to exploit and as a result are best suited to simple, adaptation-free algorithms.

2.2. Exploration and Exploitation

The lack of knowledge about the problem at hand introduces an explore/exploit problem. For example, it may be worth spending time working out how many survivors there are, then behaving accordingly (perhaps by selecting among different learning mechanisms to use). The exploration phase can be seen as determining which of a range of possible problems one is actually facing. However, optimum behaviour will probably involve balancing exploration and exploitation at the same time, which is a challenging requirement.

3. Solution Method

We have chosen to take a model-free learning approach to solving this problem. The first observation to make is that the primitive state and action spaces are much too large to learn in directly, so we need ways of reducing them. Another observation is that multiple UAVs generally need to be coordinated in order to maximise their coverage of the map and minimise their overlaps. Specifically, the SPRC encourages multiple sensor-passes over an area while making the time to return to a single point relatively large, so that coordination is required in order to keep the time between multiple views to a minimum. A simple approach is to have all UAVs fly in some kind of formation, the exact
shape of which could be dependent on the density of survivors and number of other UAVs available. Figure 1 shows an example.

![Figure 1](exploration_exploitation.png)

**Figure 1.** The upper two UAVs explore the search space whilst the lower three are configured to follow them, using the new information to scan survivors.

A way to abstract from primitive actions is to combine a sequence of actions into a behaviour that achieves some goal (such as sending a UAV to a destination) or maintains a condition (stays at the centroid of the other UAVs).

This presents two learning/optimisation problems: 1) determining which behaviour each UAV will have at any time and 2) tuning individual behaviours for global optimality (thresholds, turning decisions etc.).

To allow for collaboration, a behaviour might cover multiple UAVs (three UAVs concentrate on one area while others concentrate on another). Unfortunately allowing joint behaviours of any size creates a combinatorial explosion of possible behaviour allocations – there are \( B_k = \sum_{k=0}^{a-1} \binom{a-1}{k} \cdot B_k \) ways to partition \( a \) UAVs between joint behaviours.

There are various methods to reduce this vast range of options. Our approach is to consider only individual behaviours which can still be parametrised by another UAV, such as follow(UAV2), allowing for collaboration whilst reducing the search space. As coordination of efforts as an emergent property is by no means guaranteed (consider two UAVs that independently decide to follow each other - this is unlikely to be fruitful), we also allow for conditions to be added that may disallow certain patterns of behaviour. This also has the practical advantage of simplifying migration to a decentralised system.

### 3.1. Learning Behaviour Assignment

Assigning behaviours to UAVs is now a case of learning a policy that assigns a behaviour for each UAV, based on the current state. To reduce the size of the state space, we use a feature space combining some encoding of the global situation (density map, UAV positions, etc.) and the current behaviours\(^1\).

As we have a reward signal, reinforcement learning or a genetic algorithm could be used to learn policies. Rewards are available throughout a trial, suggesting a temporal difference method if reinforcement learning is used. We choose reinforcement learning in our approach because it gives us a value function that allows us to order behaviours in a given state. Thus, if a condition makes one behaviour impossible then the next best can be chosen. Direct policy search approaches, such as genetic algorithms, do not provide such a function.

#### 3.1.1. Optimality

With each assumption such as restricting ourselves to behaviours and a set of features we are possibly increasing our minimum distance from a globally optimal solution. For learning a behaviour assignment policy we assume a hierarchical optimality criterion which is optimal given the set of behaviours. In tuning behaviours we may make the current policy sub-optimal; therefore tuning and policy improvement are best performed alternately.

We can consider a policy to be a behaviour for solving a particular scenario. Although the challenge will test our solution for a single scenario, one could consider a broader definition of optimality as how robust a solution is, i.e. what area of the scenario parameter space does this solution perform well in? More robust solutions require less classes of approach to cover all reasonable initialisations of the SPRC problem.

### 4. Final Remark on Necessity of Learning

One question yet to be answered is whether learning is needed for this type of problem. In creating a set of behaviours, state-space features that are informative for learning assignment of behaviours, and possibly a hierarchy of behaviours, we are reducing the complexity of the learning problem by reducing the search space. At some point we may provide so much information that we have hand-coded the solution, leaving little or no need for learning. If this solution turns out to be reasonably optimal/robust over a range of scenarios, then we would still consider this a positive outcome.

\(^1\)Without considering the current behaviour of the UAV (and possibly those of the other UAVs), it is possible to flip-flop between one behaviour and another.