Asynchronous Reliability-Aware Multi-UAV Coverage Path Planning

Mickey Li
Bristol Robotics Laboratory
University of Bristol
Bristol, UK
mickey.li@bristol.ac.uk

Arthur Richards
Bristol Robotics Laboratory
Unviersity of Bristol
Bristol, UK
arthur.richards@bristol.ac.uk

Mahesh Sooriyabandara
Bristol Research and Innovation Laboratory
Toshiba Research Europe Ltd
Bristol, UK
mahesh@toshiba-trel.com

***Abstract*—Graceful degradation is a potential advantage of Multi-Robot Systems over Single-Robot Systems. In aerial robotics applications, such as infrastructure inspection, this trait is desirable as it would improve mission reliability despite the use of failure-prone low-cost drones. The Reliability-Aware Multi-Agent Coverage Path Planning (RA-MCPP) problem finds path plans for each robot to maximise the probability of mission completion by a given deadline. This paper proposes a path planner for RA-MCPP formulated in continuous time, enabling more complex realistic environments to be considered. The proposed method (i) extends a reliability evaluation framework to evaluate the Probability of Completion metric on asynchronous strategies on non-unit lattice graph environments, and (ii) introduces a greedy-genetic meta-heuristic optimisation method as a scalable and accurate RA-MCPP solver. This method is shown to provide plans with higher reliability when compared with existing approaches in three real inspection scenarios.**

***Index Terms*—Multi-Robot Systems, Coverage Path Planning, Reliability Analysis**

I. Introduction

Ensuring performance with respect to time constraints and external risks is a common requirement in real-life planning. This is especially true in aerial robotics applications, as small unmanned aerial vehicles (UAVs) are prone to failures [1, 2]. Multi-UAV solutions are attractive for their flexibility, scalability, and tolerance to individual failure in comparison to single-UAV solutions [3]. This comes at the cost of more challenging coordination, for which task allocation methods typically seek efficient division of work to prioritise early completion [4]. However, consider the example of a railway bridge inspection: the priority is to avoid disruption to services, requiring reliable completion before the passage of the next train. This may demand a different distribution of work to that of prioritising an early finish. A reliablity-informed Multi-UAV approach has the potential to achieve these requirements where Single UAV solutions may struggle [5].

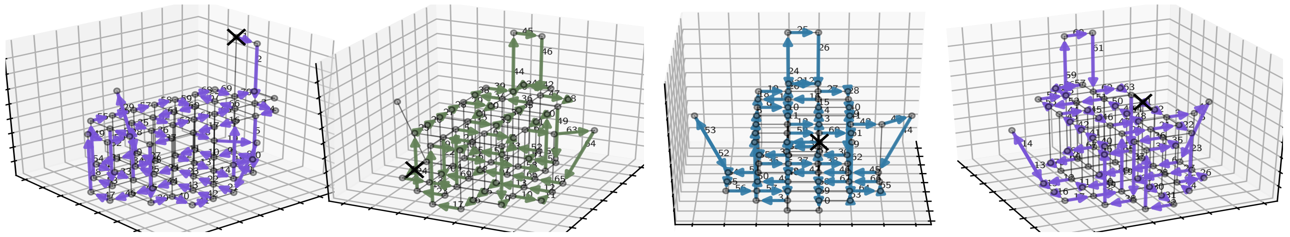
The Reliability-Aware Multi-Agent Coverage Path Planning (RA-MCPP) problem is to find paths for each robot which will maximise the probability that every part of the environment will be covered by a deadline. It extends Multi-Agent Coverage Path Planning (MCPP) which solely seeks to find minimum cost paths such that the whole environment is visited [6]. In MCPP, to minimise the cost, often travel time, each node is planned to be visited by only a single robot. However, this is a poor solution for RA-MCPP, as a single robot failure would require another robot to complete the missed tasks in addition to its own, likely exceeding the allotted time and incurring additional costs. RA-MCPP uses explicit models of agent failure-rates to optimise Probability of Completion (PoC). Hence it promotes overlapping robot-task allocations, offering higher mission reliability at the expense of nominal-case time.

Previous formulations of RA-MCPP [7, 8] assumed synchronous motion of agents across a uniform lattice of tasks. This enabled a Markov model of PoC to be embedded in optimisers such as MILP or GA. The contribution of this paper is to extend RA-MCPP to less constrained scenarios. This method consists of two parts (i) a formulation of the PoC model including asynchronous moves and tasks of varying duration; and (ii) a meta-heuristic optimisation method referred to as the Greedy Genetic Algorithm for solving the extended RA-MCPP. The solution is constructed iteratively by using a genetic algorithm to greedily choose additional trajectories which maximise PoC. The new optimisation approach, along with existing evolutionary and heuristic methods [8], is then evaluated on two variants of an aeroplane inspection scenario and a laboratory tour scenario (Figure 1 and 3). The results demonstrate that our new method can produce plans with higher reliability than previous methods in more realistic conditions.

II. Related Work

Resilience to failures for multi-robot coverage is currently an active area of research. Early work [10, 11] defined a given plan as *robust* if the mission will eventually complete as long as at least one robot remains alive. This, however, is often the worst case, with no graceful degradation, likely providing conservative strategies in practice. More recently, [12] considers MCPP in an unknown environment with failure-prone robots by applying distributed game theoretic decision methods in order to cooperatively decide task re-

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) iCASE with Toshiba Research Europe Ltd and FARSCOPE Centre of Doctoral Training at the Bristol Robotics Laboratory


(a) Aeroplane Inspection Environment 1 - 4 drones achieves $PoC = 0.9978$ at $t = 597s$ with *bathtub1500* model.

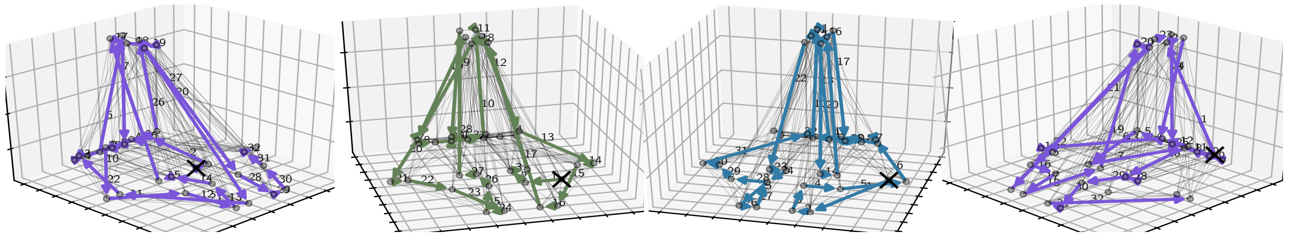

(b) Aeroplane Inspection Environment 2 - 4 drones achieves $PoC = 0.9636$ at $t = 304s$ with *bathtub800* model.

Fig. 1: Inspection paths found by the Greedy Genetic method for 4 agents covering a Boeing 747 model. The waypoints found using our implementation of [9]. Cross indicates starting location, traversable edges in grey, path marked by arrows

allocations on failure. Similar to our work, task allocations are evaluated by their probabilities of success given a model of battery reliability. [13] combines a reliability constraint on the optimisation of the number of drones required to complete an area coverage task, with a local re-planner to actively compensate if a robot is lost. Both of these methods are online reactive methods which seek to find actions which minimise the effect of robot failure on the overall objective. In contrast, our work considers the fundamental effect of planning the initial paths with respect to a reliability-aware objective function through a-priori analysis of known environments. This is similar to [14–16], where the authors' a-priori design strategies for non-coverage tasks are robust to a fixed number of failures during execution. However, our strategy optimisation also takes into account the failure of any number of robots. This work is also informed by Reliability Optimisation, applied to task allocation on distributed computer systems by [17, 18]. Our method extends theirs by both requiring spatial task orderings, and supporting varying task lengths with asynchronous execution.

In previous work, [7] presented a brief introduction of the reliability evaluation framework along with a Linear Program solver for discrete time RA-MCPP. [8] then formally introduced the RA-MCPP problem, the complete mathematical formulation of the evaluation framework, the *Probability of Completion* (PoC) metric, and presents a genetic algorithm (GA) for finding RA-MCPP path plans. However, in this previous work it is assumed that the environment can be discretised into a unit lattice with all moves occurring synchronously. In this work, these two assumptions are relaxed, in order to apply RA-MCPP to larger, more complex and realistic environments. Note that it is still assumed that robots move at a constant speed of one unit per second, only stopping when failed, and multiple robots can exist on the same location (no collision).

III. Reliability-Aware Multi-Agent Coverage Path Planning

A. Preliminaries

Let the state of the system of n agents at a time t with deadline $\bar{t}$ to be $x = (t_1, \ldots, t_n) \in [0, \bar{t}]^n = \mathbf{S}$, where t_i^t is the length of time an agent has survived. The environment graph $G(\mathbf{J}, E)$ defines a set of m individual tasks, described by the nodes $\mathbf{J} = (j_1, \ldots, j_m)$, with the edges, E describing valid traversable paths between two tasks. For an agent i, the failure probability density is denoted $f_i(t)$, cumulative density $F_i(t)$, The probability of agent i surviving at time t is the *reliability* $R_i(t) = 1 - F_i(t)$.

B. Problem Formulation

Given a set of n agents, each following failure distributions $f_i(t)$ and environment graph $G(\mathbf{J}, E)$ containing a set of m task nodes and a set of traversable edges E between them. Let $\psi_i \in \Psi$ be a finite ordered subset of connected tasks $j \in \mathbf{J}$ (*i.e.* a path through G). A path for each agent forms a strategy $\psi = \{\psi_1, ..., \psi_n\} \in \Psi^n$. The objective is to find the n paths in the strategy $\psi_1, ..., \psi_n$ which maximises the reliability metric of probability $PoC(\psi)$ by a deadline $\bar{t}$, whilst ensuring all tasks have been visited. More formally,

$$\max_{\psi \in \Psi^n} PoC(\psi) \tag{1}$$

subject to,

$$\bigcup_{i \in 1..n} \psi_i = \mathbf{J} \tag{2}$$

$$(\psi_{i,k}, \psi_{i,k+1}) \in E \quad \forall k < |\psi_i|, \forall i \in 1...n \tag{3}$$

C. Probability of Completion Metric

Given a particular strategy ψ, robots will move around and complete tasks with the possibility of failing. At each state of the system x, either all tasks have been visited and the mission is completed, or not. Therefore given ψ, the state space $\mathbf{S}$ can be partitioned into two non-intersection regions of Completion $\mathbf{C}_\psi$, and Non-Completion $\overline{\mathbf{C}_\psi}$.

For any strategy given as a set of paths ψ, define an allocation matrix $T^\psi \in \mathbb{R}^{n\times m}$, where the elements T^ψ_{ij} are the *first time* at which agent i is scheduled to complete the task j. This is valid, as completion of task j is dependent only on each agents first scheduled visit to j (the only reason j would be missed is due to failure). Therefore a task j is considered completed by agent i if $t_i \geq T^\psi_{ij}$, and the **Completion Region** $\mathbf{C}_\psi$ can be defined as follows:

$$\mathbf{C}_\psi = \{x \in \mathbf{S} \mid \forall j\ \exists i\ t_i \geq T^\psi_{ij}\} \tag{4}$$

This is illustrated in Figure 2 showing the annotated state space of two agents covering a small set of 5 tasks following an example strategy. The two axes describe the state (time alive) for each agent t_i, with the deadline $\bar{t}$ clearly denoted. The times at which each agent's tasks have been scheduled are marked on the axis and grid lines. Then, for this particular strategy, the red boundary denotes the earliest set of points where completion has been achieved, *i.e* $t_i = T^\psi_{ij}$. The blue region, including the red boundary, thus graphically define the completion region $\mathbf{C}_\psi$. This completion region stretches onwards to infinite time.

The *Probability of Completion* is then defined as the probability that any realisation of the system will reach a completion state and fall within the completion region. In general, this is equivalent to computing the total probability mass within the completion region. Note that in Figure 2, the completion region is comprised of rectangles $k \subseteq \mathbf{C}_\psi$ formed by the task orderings. For instance, the highlighted region $k_{2,3}$ is a result of the case where agent 1 and agent 2 fail between their 2nd and 3rd, and 3rd and 4th assigned tasks respectively. Therefore, the probability density of each region k is defined by the scheduled times $\underline{t_i^k}$ and $\overline{t_i^k}$ for consecutive tasks of each agent i. Then the PoC is the sum of the integral over the probability densities of each $k \subseteq \mathbf{C}_\psi$. With more agents, the dimension of the state space increases, and the regions k now define hyperrectangles in n-d space with n agents. This can be formally defined:

Definition 1. *Given independent agents, each with failure CDF $F_i(t)$, the* ***probability of completion*** *(PoC) for a given path plan ψ is the integral of the probabilities of reaching any state within the completion region $\mathbf{C}_\psi$.*

$$PoC(\psi) = \int_{x\in\mathbf{C}_\psi} p(x)\ dx \tag{5}$$

$$= \sum_{k\subseteq\mathbf{C}_\psi} \prod_{i\in[1..n]} F_i(\overline{t_i^k}) - F_i(\underline{t_i^k}) \tag{6}$$

where $\underline{t_i^k}$ and $\overline{t_i^k}$ refers to the lower and upper time bounds of the region k for each agent i, corresponding to the scheduled times of consecutive tasks of an agents. Note for a region k corresponding to an agent's last scheduled task, $\overline{t_i^k} = \infty \implies F_i(\overline{t_i^k}) - F_i(\underline{t_i^k}) = 1 - F_i(\underline{t_i^k}) = R_i(\underline{t_i^k})$, the Reliability.

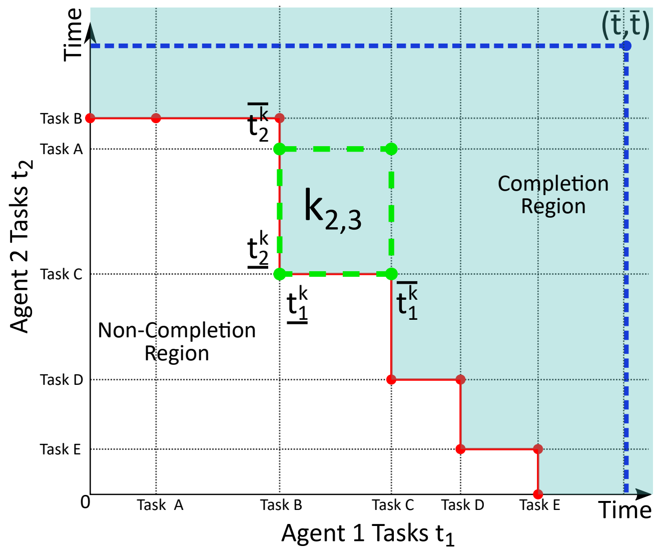


Fig. 2: Annotated state space for 2 agents $x = (t_1, t_2)$, with deadline $\bar{t}$, covering a set of 5 tasks by following an example strategy. The authors encourage the reader to reference the supplementary video for an animated explanation.

D. PoC Implementation

In order to compute PoC, the solver must then enumerate all the boundaries of each region $k \subseteq \mathbf{C}_\psi$. This is equivalent to enumerating all combinations of tasks for each agent, and computing the density of the corresponding k regions, emitting $O(m^n)$ complexity for n agents and m tasks. Observe that $\mathbf{C}$ is divided into two continuous regions, separated by a single continuous border. By taking advantage of this topology, the method can be improved by only considering all possible combinations of tasks for $n-1$ agents. For each combination, a binary search can be applied over the tasks of the nth agent to find the earliest task for which the corresponding region is within $\mathbf{C}_\psi$, *i.e.* the earliest task lying on the red boundary. In Figure 2, this is equivalent to, for each row, performing a binary search for the boundary, and calculating the density of the strip to the right of that boundary. This improves the complexity to $O(m^{n-1}\log m)$. Especially when used as an optimisation objective, this improvement is significant for the evaluations of strategies with large numbers of tasks and agents.

IV. Greedy-Genetic Algorithm Approach To Solving RA-MCPP

This section argues that PoC is a monotone submodular function so RA-MCPP can then be reformulated as a submodular maximisation problem subject to cardinality constraints, and thus solvable by a Greedy algorithm. However, practical applications of this technique may often be intractable. Therefore a Greedy-Genetic Algorithm is proposed for use in practice.

A. *Submodular Formulation of PoC*

Existing approaches all attempt solutions by simultaneously optimising all n trajectories of ψ. In many recent works, researchers have exploited the diminishing returns property of submodular functions in order to obtain at most $1 - \frac{1}{e}$ error bounded solutions through greedy strategies [19, 20].

Proposition 1. *$PoC(\psi)$, $\psi \subset \Omega$ is a monotone submodular function, where Ω is the ground set containing all walks through G up to a finite length.*

Set the maximum walk length to be longer than any reasonable path. Consider constructing a strategy ψ by adding one agent at a time. Each added agent i appends an additional path ψ_i to the existing paths, creating a new strategy $\psi \cup \{\psi_i\}$. As each path ψ_i is added, either (i) ψ_i will not reach any task earlier than the existing paths, with no improvement in $PoC(\psi)$, or (ii) ψ_i will reach a task earlier and possibly improve $PoC(\psi)$. Therefore PoC must be monotonic w.r.t additional paths.

Submodularity is then shown if adding the same agent path ψ_i to ψ gets diminishing returns as ψ incorporates more agents (*i.e.* grows in size) [20], if $\psi_A \subseteq \psi_B \subseteq \Omega$:

$$PoC(\psi_B) - PoC(\psi_A) \geq PoC(\psi_B \cup \{\psi_i\}) - PoC(\psi_A \cup \{\psi_i\}) \tag{7}$$

From monotonicity, each additional path potentially improves PoC by improving the probability of early task completion. For the LHS, comparing two sets of paths $\psi_A \subseteq \psi_B$, there exists tasks which are improved only by the new trajectories ψ_B/ψ_A. These new paths contribute an amount $p_d \geq 0$ to $Poc(\psi_B)$. For the RHS, monotonicity demonstrates that $PoC(\psi_B \cup \{\psi_i\}) \geq PoC(\psi_B)$ and $PoC(\psi_A \cup \{\psi_i\}) \geq PoC(\psi_A)$ due to ψ_i possibly getting to tasks earlier. Again comparing $\psi_B \cup \{\psi_i\}$ with $\psi_A \cup \{\psi_i\}$ the difference is also only due to the trajectories ψ_B/ψ_A as ψ_i is a member of both. However this must contribute an amount $p'_d \leq p_d$ as ψ_B/ψ_A may repeat improvements already made by ψ_i. Thus the difference in LHS $p_d \geq p'_d$, the difference of the RHS, and PoC is submodular. A formal proof is omitted for brevity.

Therefore, RA-MCPP can be restated as choosing a set of trajectories from the ground set $\psi \subset \Omega$ (in addition to existing constraints (2) and (3)):

$$\max_{\psi \subset \Omega} PoC(\psi, \bar{t}) \quad \textit{subject to} \quad |\psi| \leq n \tag{8}$$

The maximisation of submodular functions in general is NP hard [20], However the maximisation of monotone submodular function with cardinality constraint $|\psi| \leq n$ allows for a simple greedy algorithm which selects an element with the maximal marginal gain on the submodular function each iteration, giving the $1 - \frac{1}{e}$ approximation of the optimal [20].

B. *Greedy-Genetic Algorithm*

Unfortunately, in all but the smallest of graph environments, it is intractable to enumerate the set of all trajectories Ω in order to find the greedy choice. Therefore this work

Algorithm 1 Greedy Genetic Solver

```
Input: number of agents $n$, environment graph $G$, agent
    failure models $f$
Output: Strategy $\Psi$, Probability of Completion $PoC_{best}$
1: $\Psi \leftarrow \emptyset$, $PoC_{best} = 0.0$
2: for $i = 1$ to $n$ do
3:     // Input is the current set of trajectories, number of
       agents, environment graph and failure models
4:     $PoC_{new}, \psi_i = geneticFindTrajectory(\Psi, i, G, f)$
5:     $PoC_{best} = PoC_{new}$
6:     $\Psi = \Psi \cup \{\psi_i\}$
7: end for
8: return $\Psi, PoC_{best}$
```

utilises a genetic algorithm to generate single candidate trajectories ψ_i which maximise PoC when combined with trajectories $\psi_1, ..., \psi_{i-1}$ found in previous iterations (Algorithm 1). Whilst the submodular approximation bounding no longer holds, the method still provides good strategies in practice.

A genetic algorithm is a meta-heuristic optimisation approach characterised by modelling the state as a chromosome, and randomly applying operators known as mutation and crossover in order to find optimal chromosomes scored by fitness [21]. The *geneticFindTrajectory* method in Algorithm 1 uses an adaptation of the previous genetic algorithm in [8]. A *chromosome* is constructed as a single ordered set of tasks $\psi_i \in \Omega$. The population $P := \{\psi_1, .., \psi_\mu\}$ for a population size μ. The chromosome fitness is defined to be the PoC of $\Psi \cup \{\psi_i\}$ where Ψ are the trajectories found in previous iterations. The strategy is constructed by accumulating the minimum time taken for each agent to travel between its tasks via the shortest path with respect to the environment graph, thus also allowing backtracking routes can also be found for environments with no Hamiltonian Cycles. The optimisation goal is then to maximise the fitness function of PoC.

In training, the algorithm is initialised with a valid tour of the environment. The following 5 *Mutation* and *Crossover* operators are implemented for reproduction: (1) *swap-mutation* randomly swaps consecutive tasks, (2) *add-mutation*, (3) *delete-mutation* are used to add or remove tasks, (4) *roll-mutation* randomly cycles the starting task by a random amount, and (5) *sequence-crossover* which takes a random pair of chromosomes and chooses a random split point on each in order to splice the paths together - start of one to the end of the other and vice versa - to output two new ones. Finally, for *selection* operators, *random selection* is used for reproduction, and *tournament selection* with *elitism* (top k chromosomes are kept) is used for constructing the next generation. For evaluation in this paper, the Greedy-GA was implemented using DEAP Python library [22] and run for 2000 generations with an initial population of 100 chromosomes. The crossover and mutation probabilities were set high to 0.5 and 0.3 respectively in order to explore more of the state space.

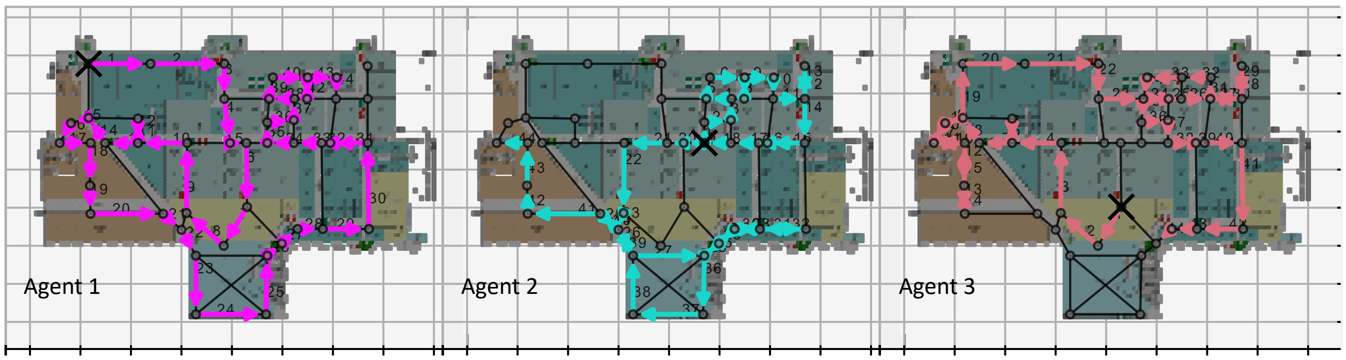


Fig. 3: Inspection paths found by the Greedy Genetic Algorithm, with 3 agents covering the Bristol Robotics Laboratory Tour Scenario. $PoC = 0.8494$ at $t = 1263s$ each using the *bathtub1500* failure model.

V. Evaluation

The presented Greedy-Genetic Algorithm is compared with existing methods: Two variants of a full path-based genetic solver [8], which attempts to simultaneously optimise all agent's trajectories. The variants maximise (i) PoC only, and (ii) the weighted sum of PoC and earliest completion time respectively. (iii) Partitioning methods representing existing solutions to MCPP (iv) A simple extension to partitioning where the agents' paths overlap with a neighbour, and (v) Random walk.

These methods are then evaluated on three sample environments: (i) Aircraft Inspection Scenario 1 in Figure 1a, an environment with many tasks but fewer connections, (ii) Aircraft Inspection Scenario 2 in Figure 1b, an environment with fewer tasks but greater interconnectivity, and (iii) A Laboratory Tour Scenario in Figure 3 to evaluate our method on a real location. For the aircraft inspection, both graph-based environments were generated from a 3D aircraft mesh by applying our implementation of Adaptive Viewpoint Sampling [9] at different resolutions and adaptation levels. The edges were generated by choosing in (i) closest nodes, (ii) all possible nodes, removing edges which would intersect the aircraft. The graph for the laboratory tour was generated by hand considering major junctions, doorways and thoroughfares.

In addition, a Bathtub failure model [8] has been chosen for each Robot, giving a more realistic failure distribution. The Bathtub model is essentially a mixture of 3 Weibull Distributions [23] representing early death, useful life, and wear out periods respectively. The Bathtub curve we use is defined by the resultant distribution generated by a weighted sum of the respective failure rates, with PDF and CDF respectively:

$$\begin{aligned} f_b(t) &= a_1 f_1(t) R_2(t) R_3(t) + a_2 f_2(t) R_1(t) R_3(t) \\ &\quad + a_3 f_3(t) R_1(t) R_2(t) \\ F_b(t) &= 1 - R_1(t) R_2(t) R_3(t) \end{aligned}$$

Refer to [8] for further details. Weights $a_i = 1/3$. In this work, all agents are assumed homogeneous and following one of the two particular bathtub distributions shown in Table I, chosen in order to best demonstrate results in the

Name	Weibull 1		Weibull 2		Weibull 3	
	λ	μ	λ	μ	λ	μ
bathtub800	0.39	2000	1.00	1000	5.80	600
bathtub1500	0.76	5000	1.00	5000	11.10	1100

TABLE I: The shape λ and scale μ parameters of the Bathtub distributions used where name indicates approximate lifespan

environments. In practice, these parameters would be found through fitting this model to failure data collected from the drones.

Figure 3 visualises a solution found by our method where interestingly, the 3rd robot overlaps the others, presumably in case of early failure. Quantitatively, Figures 4, 5 and 6, each plot the PoC over time of the strategies found by each method on each scenarios. The values of interest are the PoCs at the deadline, which is set to the time taken for an agent to traverse a tour of the environment. All of the figures show the significant difference in reliability between the strategies found by failure-informed methods compared to standard methods. In comparison to the existing GA methods, the Greedy-Genetic method is shown to find better solutions on all scenarios. In particular, in both the 1st Scenario and the lab tour (Figures 4 and 6), our method finds solutions with a visible increase in reliability. In fact, completion is almost guaranteed in the 1st Scenario with our method.

Finally, Figure 7 compares the results of the Greedy-Genetic Algorithm applied to the 2nd Aircraft Inspection Scenario with an increasing number of agents. With each additional agent, the optimal paths for the previous agents are kept. The increase in reliability as more agents are added follows intuition. Crucially, the diminishing returns property is clearly shown as agents are added, experimentally validating the submodularity of PoC, and the reformulation of RA-MCPP as a submodular maximisation (Equation 8).

VI. Conclusion

This work proposes a path planner capable of solving the Reliability-Aware Multi-Agent Coverage Path Planning (RA-MCPP) problem in continuous time, enabling the consideration of more complex and realistic environments. The RA-MCPP problem is formalised as the optimisation of the Probability of Completion (PoC) metric subject to connectivity

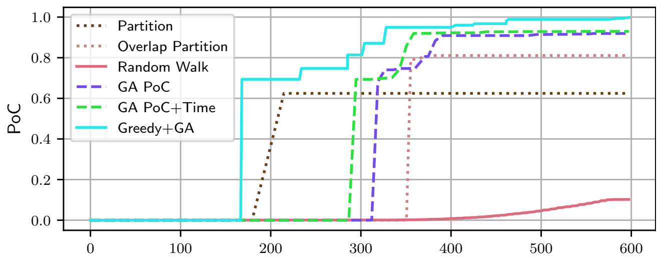


Fig. 4: PoC over time for Aeroplane Scenario 1 in figure 1a.

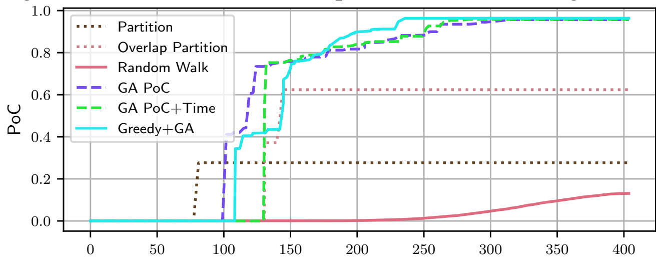


Fig. 5: PoC over time for Aeroplane Scenario 2 in figure 1b

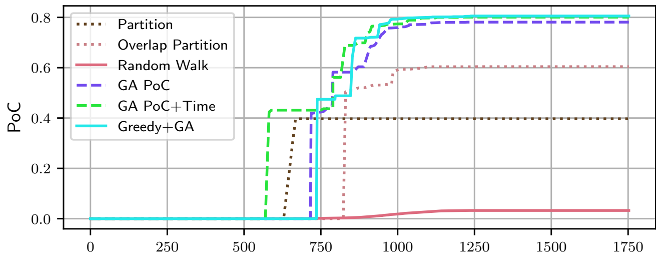


Fig. 6: PoC over time for Laboratory Scenario in figure 3

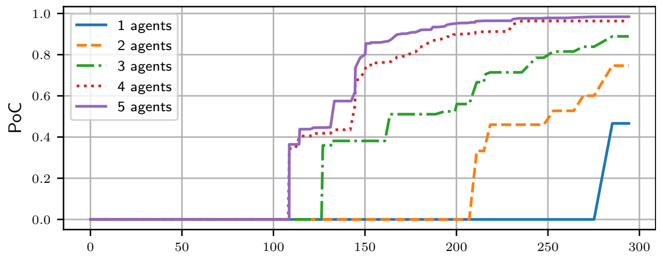


Fig. 7: A comparison of PoC of different numbers of agents covering the Aeroplane Scenario 2 in figure 1b.

and failure models. It is argued that the PoC function is a monotone submodular function and RA-MCPP can also be formalised as a submodular maximisation problem subject to cardinality constraints over the set of all paths which emits a Greedy approximation. From this reformulation, the Greedy-Genetic Algorithm is proposed where, a genetic algorithm is applied iteratively to successively find agent paths which maximise PoC. Our method is shown to provide plans with higher reliability when compared with existing approaches in three real inspection scenarios. Future directions of this work will be to validate this method on real drones, and also investigate the effect of reliability awareness in on-line dynamic planning and during reactive reorganisation when failures occur.

References

[1] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor Unmanned Aerial Vehicles," in *2012 Proceedings of IEEE Southeastcon*. IEEE, 3 2012, pp. 1–6.

[2] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 8 2018.

[3] G. Weiss, *Multiagent systems, Second Edition*. MIT Press, 2013.

[4] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 10 2013.

[5] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "A survey on inspecting structures using robotic systems," *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, p. 172988141666366, 12 2016.

[6] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 12 2013.

[7] M. Li, A. Richards, and M. Sooriyabandara, "Reliability-Aware Multi-UAV Coverage Path Planning Using Integer Linear Programming," in *UKRAS20 Conference: "Robots into the real world" Proceedings*. EPSRC UK-RAS Network, 5 2020, pp. 15–17.

[8] M. Li, A. Richards, and M. Sooriyabandara, "Reliability-Aware Multi-UAV Coverage Path Planning using a Genetic Algorithm," *Accepted for 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 5 2021.

[9] R. Almadhoun, T. Taha, D. Gan, J. Dias, Y. Zweiri, and L. Seneviratne, "Coverage Path Planning with Adaptive Viewpoint Sampling to Construct 3D Models of Complex Structures for the Purpose of Inspection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10 2018, pp. 7047–7054.

[10] N. Hazon and G. A. Kaminka, "On redundancy, efficiency, and robustness in coverage for multiple robots," *Robotics and Autonomous Systems*, vol. 56, no. 12, pp. 1102–1114, 12 2008.

[11] P. Fazli, A. Davoodi, P. Pasquier, and A. K. Mackworth, "Complete and robust cooperative robot area coverage with limited range," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 10 2010, pp. 5577–5582.

[12] J. Song and S. Gupta, "CARE: Cooperative Autonomy for Resilience and Efficiency of robot teams for complete coverage of unknown environments under robot failures," *Autonomous Robots*, vol. 44, no. 3-4, pp. 647–671, 3 2020.

[13] R. K. Ramachandran, L. Z. J. A. Preiss, and G. S. Sukhatme, "Resilient Coverage: Exploring the Local-

to-Global Trade-off," 10 2019.

[14] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 1 2019.

[15] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient Flocking for Mobile Robot Teams," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1039–1046, 4 2017.

[16] L. Guerrero-Bonilla, A. Prorok, and V. Kumar, "Formations for Resilient Robot Teams," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 841–848, 4 2017.

[17] S. Shatz, J.-P. Wang, and M. Goto, "Task allocation for maximizing reliability of distributed computer systems," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1156–1168, 1992.

[18] H. R. Faragardi, R. Shojaee, and N. Yazdani, "Reliability-Aware Task Allocation in Distributed Computing Systems using Hybrid Simulated Annealing and Tabu Search," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*. IEEE, 6 2012, pp. 1088–1095.

[19] J. Liu and R. K. Williams, "Monitoring Over the Long Term: Intermittent Deployment and Sensing Strategies for Multi-Robot Teams." Institute of Electrical and Electronics Engineers (IEEE), 9 2020, pp. 7733–7739.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 12 1978.

[21] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 6 2006.

[22] F.-A. Fortin, U. Marc-André Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," Tech. Rep., 2012.

[23] P. D. T. O'Connor and A. Kleyner, *Practical reliability engineering*. Wiley, 2012.