\textit{k}-survivability: Diversity and survival of expendable robots

Yu-Han Lyu, Yining Chen, and Devin Balkcom\textsuperscript{1}

\textbf{Abstract}—We define the \textit{k}-survivability of a set of \(n\) paths as the probability that at least \(k\) out of \(n\) robots following those paths through a stochastic threat environment reach goals. High \(k\)-survivability sets tend to contain short and diverse paths.

Finding sets of paths with maximum \(k\)-survivability is \textit{NP}-hard. We design two algorithms: a complete algorithm that finds an optimal list of paths, and a heuristic method that finds paths with high \(k\)-survivability. Although computing \(k\)-survivability is expensive and this work is still preliminary, we believe that understanding the relationship between diversity and survival will yield new insights into multi-robot motion planning.

I. INTRODUCTION

How should a set of robots move through a dangerous environment to accomplish objectives? Is it better for the robots to travel together, or should the robots split up? What is the relationship between survival and diversity of actions?

As an example, consider the following whimsical planning problem: \(n\) ants must migrate from one nest to another through a field containing both obstacles and antlions, which make disc-shaped traps. If we assume a uniform distribution of trap locations, which \(n\) paths should the ants follow, if the ants must decide their paths before moving and cannot reroute during movement?

One idea might be to maximize the expected number of surviving ants. However, the best strategy for this problem turns out to be uninteresting and unwise: find the safest path for a single ant (for simplicity, assume there is a unique safest path), and have all ants follow that path. This solution is not robust — a single trap could destroy the entire colony. Therefore, we consider a problem that is more suitable if ants are expendable: maximize the probability that at least some \(k\) (with \(k \leq n\)) ants survive. If the number of traps is unknown, the solution may contain up to \(n\) unique paths.

Figure 1 shows an example problem for which paths have been selected to achieve high survivability of routes across a college campus. The paths are short, interestingly diverse, and may be of practical interest if there is actual danger, traffic congestion, or surveillance to be avoided.

We believe this to be the first work that explicitly studies the theoretical implications of robot expendability. Path diversity has been explored in several settings, with applications including motion planning [1]–[7], robust routing in computer networks [8], and dissimilar paths in transportation [9]. Approaches to finding diversity typically involve defining an arbitrary distance metric that describes separation of paths, and finding solutions that balance distance between paths against length of paths, using linear or non-linear weights, constrained optimization, or by analyzing the Pareto frontier.

Instead of defining an arbitrary pairwise path diversity metric or choosing arbitrary tradeoffs between path lengths and diversity metrics, our approach proceeds directly from the threat model, since we believe that diversity should be considered as a means rather than an end.

We define \textit{k}-survivability to measure the quality of paths in a stochastic threat environment. Sets of paths with high \(k\)-survivability naturally balance length and diversity. Although choosing sets of paths to maximize \(k\)-survivability is \textit{NP}-hard, we design a complete algorithm. Since the maximization algorithm is computationally infeasible except for \(k = 1\) and \(n = 2\), we also design a practically faster heuristic method that finds paths with high \(k\)-survivability.

\textsuperscript{1}THE author is with the Department of Computer Science, Dartmouth College. Contact: devin@cs.dartmouth.edu

\textsuperscript{2}THE campus map is from the Dartmouth College website and the street data is from OpenStreetMap. Since data from OpenStreetMap is inconsistent with the campus map, some vertices are slightly inside buildings.
is in effect under the fixed trap model, a different trap shape (or even unknown but drawn from some known distribution. Even known shapes, and variable traps of traps will be discussed in the next section.

Unknown stochastic dangers are called $G$ is represented by a graph $D$ is represented by a point set $A$. Model diamonds are example fixed traps. $G$ circles are vertices of $D$ and the free space. See Figure 2. (a) Paths with the highest 1-survivability under the fixed 1-disc trap model.

The diversity of trajectories in motion planning has been studied by several researchers [1]–[7]. Our work is most related to Erickson’s and LaValle’s work [14]. They propose a definition of survivability that measures the correlation of all probabilities are equal, then the model is a uniform fixed trap model.

For example, under a uniform fixed $r$-disc trap model, each vertex has equal and independent probability to be the center of a disc trap of radius $r$. Figures 3a and 3b show examples.

A variable trap $A$ is represented by a distribution over a set of fixed traps. A variable trap model $M$ is represented by a collection of variable traps and corresponding probabilities: $M = \{(A_i, p_i)\}_{i=1}^{|M|}$.

For example, under a variable $r$-disc trap model, each vertex has identical and independent probability to be the center of a disc trap, whose radius follows a geometric distribution with mean $r$. Two paths with high 1-survivability under the variable 5-disc trap model are shown in Figure 3c.

II. RELATIVE WORK

Diversity has been studied in location theory, motion planning, graph theory, computer networks, and transportation.

Location theory: In location theory, the maximum diversity problem is to find $m$ points maximizing diversity among given points in a metric space. Although location theory focuses on finding diverse points, methods can be adapted to find diverse paths as long as a metric space on paths can be defined. Formulations include [10]:

1) remote-edge problem: find a set of points maximizing the minimum mutual distance (also called the $p$-dispersion problem [11]).
2) remote-pseudoforest problem: find a set of points maximizing the sum of the distance to the nearest neighbors (also called the $p$-defense problem [12]).
3) remote-clique problem: find a set of points maximizing the sum of mutual distances (also called the max-avg facility dispersion problem [13], or the maximum dispersion problem [11]).

Fig. 3: Example paths for two robots in different environments and parameters. In Figure 3a, since $r = 1$, the optimal solution has parallel subpaths with distance two to avoid being destroyed by one 1-disc easily.

A. Model

We focus on the discrete problem in which the environment is represented by a point set $D$ and the free space is represented by a graph $G = (V, E)$, where $V \subseteq D$. Unknown stochastic dangers are called traps. Several models of traps will be discussed in the next section.

Our problem is to find paths for $n$ point robots such that the $i$-th path connects the designated start vertex $s_i \in V$ and the designated goal vertex $g_i \in V$. See Figure 2. Robots cannot communicate, do not have sensors, and cannot reroute; both obstacles and traps are time-independent.

We define $k$-survivability to be the probability that at least $k$ paths successfully connect their (perhaps different) starts to goals. The $k$-survivability problem ($k$SP) is formalized as:

**Input** = $(G, M, \{(s_i, g_i)\}_{i=1}^n, k)$, where
1) $G = (V, E)$ denotes the free space.
2) $M$ is a trap model (see next section).
3) $n$ point robots have start locations $s_i \in V$ and goal locations $g_i \in V$ for all $1 \leq i \leq n$.
4) survivability parameter $k$, with $1 \leq k \leq n$.

**Output** = $P$, a list of $n$ paths maximizing $k$-survivability such that for all $1 \leq i \leq n$, $P_i \in P$ connects $s_i$ and $g_i$.

We now discuss two trap models: fixed traps, which have known shapes, and variable traps, for which the shape is unknown but drawn from some known distribution. Even under the fixed trap model, a different trap shape (or even multiple traps) may be placed at each vertex.

A fixed trap $F$ is a subset of $D$. When a fixed trap $F$ is in effect, all paths passing through $F$ are blocked. A fixed trap model $M = \{(F_i, p_i)\}_{i=1}^{|M|}$ is a collection of fixed traps and their corresponding, independent probabilities. If

![Figure 2: Eight paths with high 1-survivability. Small gray circles are vertices of $G$ (4-connected). Squares are obstacles; diamonds are example fixed traps.](image)

![Figure 3: Example paths for two robots in different environments and parameters.](image)
damage on paths when a random disc obstacle is placed on a path. Whereas survivability favors separated paths, $k$-
survivability is a direct probabilistic measure of survival that
in some cases can be maximized by allowing robots to follow
overlapping short paths.

Finding trajectories in a threat environment has been
studied for aircrafts [15], UAVs [16], vehicles [17], and
ships [18]. Our work differs in that the threat model is
probabilistic, and in the search for multiple trajectories.

In the Euclidean plane, finding a path connecting two
points among polygonal obstacles can be solved efficiently [19]. One possible definition for the diversity of paths
in the Euclidean plane is the number of distinct homotopy
classes of paths [20]. Eriksson-Bique et al. studied the
problem of finding $k$ shortest paths with distinct homotopy
classes [21].

Path diversity on graphs: The problem of finding $k$
shortest paths on a graph has been studied since the ’70s [22],
as the problem of finding vertex or edge disjoint paths [23].
When each vertex/edge is associated with a failure probability,
short and reliable paths are desirable. Finding short paths
subject to reliability constraints can be considered as resource-constrained shortest-path problems [24]. These models
only disfavor paths sharing edges or vertices, while
survivability disfavors paths passing through the same
traps, which is more general.

Robust routing in computer network: One way to improve
the robustness of a network is to increase the path diversity
between end-points [25]. Diverse routing problems have
been studied for more than a decade using graph theory
methods [8]. Rohrer et al. define the diversity of paths based
on the distance on graphs and geographic distances [26],
which is similar to the idea of path space [5].

Dissimilar paths in transportation: The problem of finding
dissimilar paths has been studied in transportation, since
dissimilar paths avoid bottlenecks and are beneficial for e.g.
hazardous waste transportation [9], [27].

III. COMPUTING $k$-
survivability

Since $k$-
survivability is independent of the order of vertices
along paths, paths are represented as sets of vertices.

A. Computing $k$-
survivability under the fixed trap model

Given a fixed trap model $M = \{ (F_i, p_i) \}_{i=1}^{M}$ and a path $P$
on a graph $G$, the forbidden index set of $P$ is $\text{Forbid}(P) = \{ i \mid P \cap F_i \neq \emptyset \}$. The probability that $P$ is not blocked
equals $\Pr(P) = \prod_{i \in \text{Forbid}(P)} (1 - p_i)$. Similarly, for a set of
paths $\mathcal{P} = \{ P_1, \ldots, P_k \}$, the forbidden index set of $\mathcal{P}$ is
$\text{Forbid}(\mathcal{P}) = \bigcup_{P \in \mathcal{P}} \text{Forbid}(P)$. The probability that all
paths in $\mathcal{P}$ are not blocked equals $\Pr(\mathcal{P}) = \prod_{i \in \text{Forbid}(\mathcal{P})} (1 - p_i)$.

Given a set of paths $\mathcal{P}$, 1-survivability can be computed by using the inclusion-exclusion principle as follows:

$$\text{Survive}_1(\mathcal{P}) = \sum_{S \subseteq \mathcal{P}, S \neq \emptyset} (-1)^{|S|+1} \Pr(S)$$

The computation of $k$-
survivability is similar but more time-
consuming. For a set of paths $\mathcal{P}$ and a positive integer $k$,

we define $\text{comb}(\mathcal{P}, k) = \{ S \mid S \subseteq \mathcal{P}, |S| = k \}$; that is,$\text{comb}(\mathcal{P}, k)$ contains all $k$-subsets of $\mathcal{P}$. Given a set of paths
$\mathcal{P}$, $k$-
survivability can be computed by using the inclusion-
exclusion principle:

$$\text{Survive}_k(\mathcal{P}) = \sum_{\mathcal{R} \subseteq \text{comb}(\mathcal{P}, k), \mathcal{R} \neq \emptyset} (-1)^{|\mathcal{R}|+1} \Pr(\cup_{S \in \mathcal{R}} S)$$

B. Computing $k$-
survivability under the variable trap model

Let $M = \{ (A_i, p_i) \}_{i=1}^{M}$ be a variable trap model. Each variable trap $A_i$ can be represented as a collection of fixed traps
and their probabilities: $A_i = \{ (F_{i,j}, p_{i,j}) \}_{j=1}^{a_i}$. Given a path $P$ on a graph $G$, the forbidden index set of $P$ with respect to the $i$-th variable trap is $\text{Forbid}(P, i) = \{ j \mid P \cap F_{i,j} \neq \emptyset \}$. The probability that $P$ is not blocked equals $\Pr(P) = \prod_{i \in \text{Forbid}(P, i)} (1 - \sum_{j \in \text{Forbid}(P, i)} p_{i,j})$. The calculation of $k$-
survivability can be derived in the same manner as under the fixed trap model.

IV. THEORETICAL RESULTS

In this section, we show that $k$SP is NP-hard. We also show that maximizing $k$-
survivability leads to diversity in the sense that at most $k$ robots will follow the same path, if other paths are available.

A. NP-hardness of $k$SP

We show that $k$SP under the uniform fixed trap model
is NP-hard with $n = k = 1$. Since the fixed trap model
is a special case of the variable trap model, $k$SP under the
variable trap model is also NP-hard.

Our proof, similar to the NP-completeness proof of the
minimum color path problem [28], is a reduction from the
minimum set cover problem (MSCP), defined below, to $k$SP.

Input = $(S, C)$, where

1) $S = \{1, \ldots, a\}$ is a set of positive integers from 1 to $a$.
2) $C = \{C_1, \ldots, C_b\}$ is a collection of subsets of $S$.

Output = $C' \subseteq C$ a minimum cardinality collection
of subsets whose union is $S$.

Theorem 1. $k$SP under the uniform fixed trap model with $n = k = 1$ is NP-hard.

Proof. Let $(S, C)$ be an instance of MSCP. We construct an
instance $(G, M, \{(s, g)\})$ of $k$SP in polynomial time such
that an optimal solution in $k$SP can be transformed into a
minimum set cover of $(S, C)$ and vice versa.

We construct $G = (V, E)$, which is a subgraph of a grid
graph, as follows. First, for each element $i \in S$, we create
three vertices $v_{i}^{0,0} = (4i, 0)$, $v_{i}^{1,0} = (4i + 1, 0)$, and
$v_{i}^{3,0} = (4i + 3, 0)$. We create edges $(v_{i}^{0,0}, v_{i}^{1,0})$ for all $1 \leq i \leq a$ and
edges $(v_{i}^{1,0}, v_{i+1}^{0,0})$ for all $1 \leq i < a$. Our idea is to design a
gadget for each element $i \in S$ such that $i$ can be covered by
a set in $C$ if and only if a solution of $k$SP, which is a path,
passes through $v_{i}^{0,0}$ and $v_{i}^{3,0}$.

Second, for each $j \in \{1, \ldots, b\}$, we create vertices $v_{i}^{1,j} = (4i + 1, j)$ and $v_{i}^{3,j} = (4i + 3, j)$ for all $1 \leq i \leq a$. Then,
we create edges \((v_i^{1,j}, v_i^{1,j+1})\) and \((v_i^{3,j}, v_i^{3,j+1})\) for all \(0 \leq j < b\). Intuitively, the \(j\)-th row represents the \(j\)-th set in \(C\).

Finally, we create vertices \(v_i^{2,j} = (4i + 2, j)\), edges \((v_i^{1,j}, v_i^{2,j})\), and edges \((v_i^{2,j}, v_i^{3,j})\) for each \(i \in C_j\). We use these vertices to model the constraint that every element in \(S\) is covered by a set in \(C\). See Figure 4.

The uniform fixed trap model is \(M = \{(F_j, p)\}_{j=1}^b\) for an arbitrary choice \(p \in (0, 1)\), where \(F_j = \{v_i^{2,j} \mid i \in C_j\}\). The start vertex is \(v_1^{0,0}\) and the goal vertex is \(v_3^{0,0}\).

Let \(P\) be an optimal solution of the instance \((G, M, \{(v_1^{0,0}, v_3^{0,0})\})\) of kSP. By the construction of the graph, \(P\) passes every vertex \(v_i^{3,0}\) for all \(1 \leq i \leq a\). Moreover, for each \(1 \leq i \leq a\), the only way to get \(v_i^{3,0}\) is to pass through a vertex \(v_i^{2,j}\) for some \(1 \leq j \leq b\).

Since \(v_i^{2,j}\) exists if and only if \(i \in C_j\), the set \(C' = \{C_j \mid \exists 1 \leq i \leq a, P \text{ passes } v_i^{2,j}\}\) is a set cover of \(S\).

Moreover, since all traps have the same probability, maximizing 1-survivability is the same as minimizing \(\{j \mid \exists 1 \leq i \leq a, P \text{ passes } v_i^{2,j}\}\). Hence, \(C'\) is also an optimal solution of MSCP.

Transforming an optimal solution of MSCP to an optimal solution of kSP can be done similarly.

Note that this reduction relies on the fact that individual fixed traps might be formed from disconnected sets of vertices. However, even if we restrict individual fixed traps to be contiguous, the problem still appears to be hard, since kSP under the uniform fixed trap model with \(n = k = 1\) can be used to solve the barrier resilience problem [29]. The complexity of the barrier resilience problem is still open and currently no polynomial time algorithm exists.

Even approximating an optimal solution is hard:

**Theorem 2.** No polynomial time algorithm with constant approximation ratio for kSP with \(n = k = 1\) under the uniform fixed trap model exists unless \(P = NP\).

**Proof.** We show that if a \(r\)-approximation polynomial time algorithm for kSP exists for some constant \(r\), then we can solve MSCP in polynomial time.

Let \((S, C)\) be an instance of MSCP and \(c\) be the size of a minimum set cover. By using the same reduction as in the proof of Theorem 1, we obtain an instance 

\((G, M, \{(s, g)\})\) of kSP. Since \(c\) is the size of a minimum set cover, the optimal solution of \((G, M, \{(s, g)\})\) has value \((1 - p)^c\).

Suppose that a \(r\)-approximation algorithm for kSP exists, \(0 < r < 1\), and this approximation algorithm is guaranteed to find a solution with \(1\)-survivability at least \(r(1 - p)^c\). Since the choice of probability \(p\) in the reduction is arbitrary, we set \(p\) to be a value satisfying \(r > (1 - p)^c\). Because \(r(1 - p)^c > (1 - p)^{c+1}\) and a path can only pass through an integral number of fixed traps, the approximation algorithm must return a solution with value \((1 - p)^c\), which is an optimal solution of \((G, M, \{(s, g)\})\) and can be transformed into an optimal solution of \((S, C)\) in polynomial time.

**B. Properties of kSP**

We now show that \(k\)-survivability leads to diverse paths.

**Observation 3.** For kSP under the fixed trap model with \(k = 1\), if \(n\) paths with different forbidden index sets exist, then any optimal solution does not have duplicate paths.

**Proof.** Since paths with the same forbidden index sets are either all-safe or all-blocked, using paths with different forbidden index sets improves 1-survivability.

**Observation 4.** For kSP under the fixed trap model, if at least \([n/k]\) paths with different forbidden index sets exist, then at most \(k\) robots follow the same path in an optimal solution.

**Proof.** If more than \(k\) robots follow the same path, moving one robot to another path always improves \(k\)-survivability.

Note that when \(k\) increases, the number of different paths in optimal solutions may decrease. See Figure 5, which shows some high-survivability paths for different values of \(k\).

![Figure 5: Example paths with high \(k\)-survivability for different values of \(k\) under the fixed 5-disc trap model.](image)
A. Complete algorithm

In this section, we design a complete state space search algorithm for $k$SP under the uniform fixed trap model with $n = 2$ and $k = 1$. Although it is easy to extend this algorithm for larger $k$ and $n$, solving even small problems becomes computationally infeasible with this approach.

We need several definitions. A path $P$ is an ordered list of vertices. A path $P'$ extends another path $P$, if $P$ is a prefix of $P'$. A path $P'$ is a feasible extension of $P$ if either $P'$ ends at $g$ and $P'' = P'P$, or $P$ does not end at $g$ and $P''$ extends $P$ by one vertex. Let $\text{Ext}(P)$ denote the set of all paths that end at $g$ and are extensions of path $P$.

The complete algorithm is a state space search algorithm. Each state $t$ consists of two simple paths $(P_1, P_2)$ starting from $s$. A state $(P'_1, P'_2)$ is a successor of a state $(P_1, P_2)$ if $P'_1$ and $P'_2$ are feasible extensions of $P_1$ and $P_2$ respectively.

The initial state is $((s), (s))$ and the goal states are all states $(P_1, P_2)$ that both $P_1$ and $P_2$ end at $g$. We will find one goal state with maximum 1-survivability.

Since the state space is a tree, we can use a brute-force approach to traverse the tree to find an optimal solution. In order to speed up the brute-force approach, we design a heuristic function $h$ of states, where $h(t)$ is an upper-bound of 1-survivability of all goal states that are reachable by state $t$. As long as $h(t)$ is optimistic, then the tree search will find an optimal solution. Using the heuristic function, we can prune unnecessary branches and stop search when the algorithm reaches one of the goal states for the first time.

We construct a heuristic function $h$ as follows. Remember that when $n = 2$, 1-survivability of two paths $P_1$ and $P_2$ is $\text{Pr}((P_1) + \text{Pr}((P_2) - \text{Pr}(\{P_1, P_2\})$. Suppose that there is a function $\hat{h}$ for paths that $\hat{h}(P)$ is an upper-bound of $\text{Pr}(\{P'\})$ for all $P' \in \text{Ext}(P)$. Then, we obtain a heuristic function $h(P_1, P_2) = h(P_1) + h(P_2) - \text{Pr}(\{P_1, P_2\})$.

Now, we show how to construct a function $\hat{h}$. Let $M$ be the uniform fixed trap model. For any path $P$, 1-survivability of $P$ is $(1 - p)^{|\text{Forbid}(P)|}$, which only depends on the size of $\text{Forbid}(P)$. Let $\text{LB}(P)$ be the minimum number of additional fixed traps that any extension of path $P$ must pass through to reach the goal. Formally,

$$\text{LB}(P) = \min_{P' \in \text{Ext}(P)} |\text{Forbid}(P') \setminus \text{Forbid}(P)|.$$

Then, $(1 - p)^{|\text{Forbid}(P) + \text{LB}(P)|}$ is the least upper bound of $\text{Pr}(\{P'\})$ for all $P' \in \text{Ext}(P)$.

Note that computing $\text{LB}(P)$ exactly is the same as solving $k$SP under the uniform fixed trap model with $n = k = 1$, which is a NP-hard problem by Theorem 1. In order to get an upper bound of $\text{Pr}(\{P'\})$, where $P'$ is in $\text{Ext}(P)$, it suffices to obtain a lower bound of $\text{LB}(P)$.

Our idea of obtaining a lower bound of $\text{LB}(P)$ is as follows. Let $F_i$ be a fixed trap that $i$ does not belong to $\text{Forbid}(P)$. If an extension $P'$ of $P$ passes through one vertex of $F_i$, then charge $P'$ by $1/|F_i \cap V|$. Thus, if an extension $P'$ of $P$ passes through one $v \in V$, then we charge $P'$ by $\sum_{i \notin \text{Forbid}(P), v \in F_i} 1/|F_i \cap V|$. The minimum charge of any extension of $P$ that reaches the goal, $\text{LB}'(P)$, can be computed efficiently by using a shortest path algorithm.

It is easy to see that $\text{LB}'(P)$ is a lower bound of $\text{LB}(P)$ and we know

$$\text{LB}'(P) \leq \text{LB}(P) \leq \max_i |F_i| \cdot \text{LB}'(P).$$

Thus, we can use $\hat{h}(P) = (1 - p)^{|\text{Forbid}(P) + \text{LB}'(P)|}$ to obtain a heuristic function $h$.

B. Heuristic algorithm

The previous algorithm uses a heuristic function for pruning, but is guaranteed to find optimal solutions. The heuristic algorithm described in this section does not provide this guarantee. There are three phases: path generation, path selection, and path improvement. Due to the high-dimensional search space of $k$SP, we first generate a set of candidate paths with size $w \gg n$ to reduce the search space to these $w$ paths. Then, we heuristically find $n$ paths among the set of candidate paths as an initial solution. Finally, we use local search to improve the solution until the process is stabilized. Algorithm 1 outlines the approach.

Since computation of $k$-survivability is potentially expensive, we only use the computation of $k$-survivability in the last phase. Moreover, this heuristic algorithm only needs a black box to compute $k$-survivability, and the same algorithm can be used for both fixed trap and variable trap models.

1) Path generation: The purpose of this phase is to generate a set $\mathcal{R}$ of $w \gg n$ paths. We design two methods: random generation, and an iterative penalty approach.

   a) Random generation method: To generate one random path, we generate a random spanning tree first and then pick the unique path between $s$ and $g$ on the tree. We repeat this process until $w$ paths are generated.

   b) Iterative penalty method: Another way to generate $w$ paths is repeatedly apply a shortest path algorithm. After a shortest path $P$ is found, we increase the edge weights of all edges in $P$ and repeat. Akgün et al. discuss several variants of iterative penalty methods that have different ways to penalize the path [27].

Algorithm 1: Heuristic algorithm for $k$SP

\begin{verbatim}
input : $(G, M, \{(s_i, g_i)\}_{i=1}^n, k, w, T)$, where $(G, M, \{(s_i, g_i)\}_{i=1}^n)$ is an instance of $k$SP, $w$ is a parameter of the path generation, and $T$ is a parameter of the path improvement.
output: $n$ paths connecting $(s_i, g_i)$ respectively.
\end{verbatim}

$\mathcal{R} = \text{path_generation}(w)$
$S = \text{path_selection}(\mathcal{R})$
$S = \text{path_replacement}(S, \mathcal{R})$
$Q = \emptyset$

while $|Q| < T$

\begin{verbatim}
S = \text{path_shortening}(S)
Q = Q \cup \{S\}
S = \text{escape}(S)
\end{verbatim}

return the best solution in Q.
2) Path selection: The purpose of this phase is to generate a set $n$ paths among $w$ candidate paths generated in the path generation phase. Although we can design an algorithm to find $n$ paths that maximize $k$-survivability, since the computation of $k$-survivability is exponential in $n$, this approach would be expensive. Thus, our strategy is to use different heuristics to obtain an initial solution without evaluating $k$-survivability. Then, improve the initial solution based on $k$-survivability in the next phase.

We find an initial solution by solving a different but related optimization problem.

a) Distance-based heuristic: We use $d_G(P, P')$ to denote the distance between two paths $P$ and $P'$ on a graph $G$. One candidate of the distance function is discrete Fréchet distance [7] and other candidates of distance function can be found in Knepper’s thesis [4]. Based on the distance function, we can set up several optimization problems.

1) remote-clique problem: find

$$S = \arg \max_{S \subseteq \mathcal{R}, |S|=n} \sum_{P, P' \in S} d_G(P, P').$$

2) remote-edge problem: find

$$S = \arg \max_{S \subseteq \mathcal{R}, |S|=n} \min_{P, P' \in S, P \neq P'} d_G(P, P').$$

3) remote-pseudoforest problem: find

$$S = \arg \max_{S \subseteq \mathcal{R}, |S|=n} \sum_{P, P' \in S, P \neq P'} d_G(P, P').$$

The remote-edge problem is sensitive to the closest-pair of paths, since two solutions with the same closest pair of paths will have the same minimum distance, even if one solution is much longer than the other [7]. Since all these maximum diversity problems are NP-hard, we use heuristic methods to find a good solution [30].

b) Survivability-based heuristic: We also can use Erickson’s and LaValle’s notion of survivability [14] in our heuristic. We heuristically find $n$ paths with high survivability and use this set as an initial solution.

3) Path improvement: The purpose of this phase is to improve $k$-survivability of an initial solution $S$ by using local operations: path replacement and path shortening. Path replacement iteratively replaces one path to improve $k$-survivability. Path shortening iteratively replaces a subpath of one path to improve $k$-survivability.

We first apply path replacement to improve $k$-survivability and then apply path shortening. Since path improvement is a local search method, the search process may be trapped in a local maximum. Thus, when the search reaches a local maximum, we use a randomized method to escape from the local maximum and then apply path shortening again.

a) Path replacement: Replace one path in the current solution by another path in $\mathcal{R}$ giving the maximum $k$-survivability for the set; repeat until no further improvement can be made.

b) Path shortening: Find the maximum improvement of $k$-survivability that can be made by replacing one subpath of a path in the current solution by a shortest path on $G$ connecting the endpoints of the subpath. Repeat shortening until no further improvement can be made.

Although path shortening is very effective under the fixed $r$-disc trap model, path shortening may not be useful in general models. Moreover, for $k$SP with $k > 1$, shortening just one path at a time may lead to getting trapped in local maxima easily. For example, Figure 5c shows such a case; all four overlapping paths would need to be shortened simultaneously and in the same way to allow the four robots to follow a better route.

VI. EXPERIMENTAL RESULTS

In this section, we describe several experiments (in simulation) on different heuristic methods, and compare them in terms of computation time and $k$-survivability. Remember that our heuristic method consists of three phases. We suggest two choices in the path generation phase: random generation (RG) and iterative penalty (IP) methods. We suggest four choices in the path selection phase: remote-clique (RC), remote-edge (RE), remote-pseudoforest (RF), and survivability (SU). Finally, we test two additional methods in the path selection phase:

1) random (R): pick $n$ paths in $\mathcal{R}$ uniformly at random.  
2) first $n$ paths (FN): if the paths are generated by the iterative penalty method, we pick the first $n$ generated paths.

A. Experiment setup

We used an environment containing 2500 vertices and 80 rectangular obstacles under the fixed $r$-disc trap model, where $r = 5$ and $p = 0.004$. The environment is shown in Figure 5. We used the heuristic algorithm to find $n = 5$ paths with high $k$-survivability, for $k = 1 \ldots 4$. We generated $w = 100$ paths in the path generation phase and found $T = 3$ local maxima in the path improvement phase.

The heuristic algorithm is implemented in Java and all tests were conducted on a laptop (2010 MacBook Pro) with an Intel Core i5 2.4 GHz CPU and 8GB RAM. We repeated the experiments ten times and took the average of the results.

B. Results

We first show $k$-survivability of each phase for each method in Figure 6. When $k$ is small, path shortening effectively improves the $k$-survivability and the iterative penalty method tends to perform better. However, when $k = 4$, path shortening is not effective, since our algorithm only tries to
shorten one path at a time but escaping from a local minimal may require shortening several paths at the same time.

We measured the running time for all methods maximizing $k$-survivability; the running times for each method for $k = 1$ are shown in Table I. The naïve algorithms IP + FN and RG + R are the most efficient methods. This may hint that although $k$SP is hard in general, $k$SP under the fixed $r$-disc trap model may be tractable. For $k > 1$, the naïve algorithm IP + FN is slightly faster than other methods, but we omit the results due to the page limitation.

VII. CONCLUSION & FUTURE WORK

This work is preliminary, and considers only simple $k$-survivability problems; however, we believe that $k$-survivability motivates a wealth of interesting practical and theoretical problems. For example, the problem of $k$-survivability might be reversed to plan defenses against infiltration or attack. Not all applications of $k$-survivability need be violent. For example, $k$-survivability can be considered in the context of visibility or stealth, as has turned out to be central in multi-robot pursuit-evasion games [31]–[33] for search-and-rescue operations. With a model of feedback or communication, we imagine that $k$-survivability might also provide some insights into collaboration and cooperation problems such as those that arise in sports [34] or control of large robot swarms [35].

Several future directions of theoretical research are possible. Continuous-space models might be approached using variational calculus or optimal control techniques [15], [16]. Obstacles such that the risk of a path depends on the distance between the robot and the obstacle, as for paths in mined water [18] are a potential future direction, as are time-dependent obstacles.

REFERENCES


<table>
<thead>
<tr>
<th>Path generation method</th>
<th>Path selection method</th>
<th>Path generation</th>
<th>Path selection</th>
<th>Path replacement</th>
<th>Path improvement</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td>37.3</td>
<td>0.0</td>
<td>2124.3</td>
<td>17360.9</td>
<td>19533.3</td>
</tr>
<tr>
<td>Remote-clique</td>
<td></td>
<td>31.0</td>
<td>2894.6</td>
<td>1577.1</td>
<td>17329.0</td>
<td>21841.3</td>
</tr>
<tr>
<td>Remote-edge</td>
<td></td>
<td>29.3</td>
<td>2505.0</td>
<td>1788.5</td>
<td>16791.9</td>
<td>2125.8</td>
</tr>
<tr>
<td>Remote-pseudoforest</td>
<td></td>
<td>28.6</td>
<td>2649.5</td>
<td>1854.2</td>
<td>16441.6</td>
<td>20986.7</td>
</tr>
<tr>
<td>Survivability</td>
<td></td>
<td>27.2</td>
<td>4868.69</td>
<td>1613.3</td>
<td>18391.3</td>
<td>68733.7</td>
</tr>
</tbody>
</table>

**TABLE I: Running times measured in millisecond for $k = 1$.**


<table>
<thead>
<tr>
<th>Path generation method</th>
<th>Path selection method</th>
<th>Path generation</th>
<th>Path selection</th>
<th>Path replacement</th>
<th>Path improvement</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td>430.0</td>
<td>0.0</td>
<td>2475.7</td>
<td>15752.9</td>
<td>18676.3</td>
</tr>
<tr>
<td>Remote-clique</td>
<td></td>
<td>381.9</td>
<td>4344.8</td>
<td>3300.1</td>
<td>18795.2</td>
<td>26841.3</td>
</tr>
<tr>
<td>Remote-edge</td>
<td></td>
<td>384.7</td>
<td>3550.3</td>
<td>3017.6</td>
<td>17998.5</td>
<td>24972.5</td>
</tr>
<tr>
<td>Remote-pseudoforest</td>
<td></td>
<td>379.9</td>
<td>4112.4</td>
<td>2673.5</td>
<td>18782.7</td>
<td>25970.8</td>
</tr>
<tr>
<td>Survivability</td>
<td></td>
<td>380.0</td>
<td>56769.0</td>
<td>1991.2</td>
<td>15978.2</td>
<td>75444.8</td>
</tr>
<tr>
<td>First-$t$</td>
<td></td>
<td>375.3</td>
<td>0.0</td>
<td>2968.9</td>
<td>18934.4</td>
<td>22305.3</td>
</tr>
</tbody>
</table>