Theoretical Foundations of High-Speed Robot Team Deployment

Stefano Carpin, Timothy H. Chung, Brian M. Sadler

***Abstract*— In this paper we study the multi-robot deployment problem under hard temporal constraints. After proposing a model for this task, we consider the simplest deployment algorithm and we analyze the relationship between three fundamental parameters, the temporal deadline, the probability of success, and the number of robots. Because an exact analysis of even the simplest algorithm is computationally intractable, we derive an approximate bound leading to performance curves useful to answer design questions (how many robots are needed to get a certain performance guarantee?) or analysis questions (what is the probability of success given a certain deadline and number of robots?) Simulations show that the bounds are sharp and provide a useful tool to predict team deployment performance and tradeoffs.**

I. Introduction

A primary tenet of autonomous single or multi-robot navigation has been safety, for example control architectures often seek strong collision avoidance guarantees between robots, and between robots and the environment. This introduces strict constraints whose satisfaction can dictate speed, and in some scenarios results in stopping and backtracking. However, in some cases it is desirable to balance risk and reward. For example, with a team of robots we might accept a probability of collision or failure for a subset of agents, when the overall goal(s) may be achieved in a much shorter time. This approach is more strongly relevant with smaller platforms whose individual value is low, and where the physical consequences of collisions may be relatively inconsequential. Motivating scenarios include rapid deployment in a building during an emergency, or distributed information gathering, under a hard temporal deadline.

When time is of the essence and agents move rapidly, then failure is more likely to occur. Robots may break down, get lost, have collisions, and generally fail to arrive at an intended location in a specified time. In this scenario the performance envelope is strongly influenced by the number of agents deployed, the complexity of the environment, the availability and quality of prior information, the temporal deadline or goal, and an associated target probability of success for the overall mission.

To analyze this problem, we focus on the following scenario. A given environment has a number of relevant locations, and we wish to deploy a team whose end goal is to put an agent into each of the relevant locations. Now, how many agents should be deployed to ensure that, with high probability, at least one agent reaches each relevant location within an assigned temporal deadline T?

In this paper we lay the foundation to analytically model this type of problem (Section II), and derive performance curves capturing tradeoffs in operational tempo, number of agents, and environmental complexity. We model the environment as a graph with associated functions S that describe the probability of navigating an edge as a function of time, and that are monotonically increasing with time towards probability one. We consider a simple deployment strategy (Section III, Section IV) that is scalable and robust (in a sense to be described later). With this, we can analyze the probabilistic tradeoffs. We find performance bounds, and show through simulation that the bounds are tight (Section V). The paper ends with a discussion of related work (Section VI) and conclusions (Section VII).

II. Problem formulation

The problem can be formalized as follows. K robots are deployed in an environment with N *interesting* locations. The goal for the robot team is to spread in the environment as fast as possible so that eventually, with high probability, at least one robot reaches each of the N locations. Evidently, $K \geq N$. Numerous details need to be finalized to formally study this problem. The following list introduces some of them, together with some notation.

Traversability. Robots cannot arbitrarily move between any two locations. Connectivity between locations is modeled with a graph $G = (V, E)$ with N vertices representing the relevant locations. Let M be the number of edges, and let the graph be undirected. An edge between v_i and v_j means that it is possible to directly move between these two vertices. Because of this association, in the following the terms *vertex* and *location* will be used interchangeably. Note that starting from an occupancy grid map representing a given environment, we can automatically extract the

Stefano Carpin is with the School of Engineering, University of California, Merced, CA, USA. E-mail: scarpin@ucmerced.edu.

Timothy H. Chung is with the Naval Postgraduate School, Monterey, CA, USA. E-mail: thchung@nps.edu.

Brian M. Sadler is with the Army Research Laboratory, Adelphi, MD, USA. E-mail: brian.m.sadler6.civ@mail.mil.

corresponding graph, as described in our former work [4].

Velocity/Safety tradeoff. When moving from vertex v_i to vertex v_j, a robot should decide its own velocity. The slower it moves, the more likely it is to reach its target vertex v_j. On the contrary, when the robot moves very fast, it is more likely to incur some event (e.g., bumping into a wall) preventing it from successfully reaching its destination. This relationship is modeled by a function $S : \mathbb{R}^+ \to [0, 1]$. S gives the probability of successfully reaching a target vertex as a function of the time spent to navigate there. We assume $S(0) = 0$ and that its first derivative is nonnegative. Moreover $\lim_{t\to+\infty} S(t) = 1$. In general one should assume that different edges in the graph are characterized by different S functions. For example, two vertices representing adjacent rooms connected by a narrow door should be treated differently from two rooms connected by a wide corridor. When this more accurate representation is used, we will use S_i $(1 \leq i \leq M)$ to indicate the function associated with a given edge $e_i \in V$. Specific examples for S will be given later.

Homogeneity/Heterogeneity. Robots deployed in the environment may be different or identical. *Identical* here means not only that they have the same abilities (say they all share the same platform), but also that they all run the same controller. Alternatively, they may share the same hardware, but run different control algorithms, or run on different hardware and run different controllers.

Communication. Robots may or may not exchange information before the mission commences or while it unfolds.

Knowledge. Robots may or may not know the graph G. When full awareness is not assumed, a wide spectrum of partial knowledge can be used. For example: it may be known there are N vertices but information about edges is not given. Other models can be defined as well.

Localization. Robots may or may not know their location in the environment.

Deployment. Initially, all robots may be deployed in a single vertex or they may start at different locations.

The following two parameters define mission constraints:

- T_{max}: temporal deadline to complete the deployment task. This can be a *hard* deadline, i.e., the team fails if it does not complete its task before T_{max}, or a *soft* deadline, i.e., a penalty is payed for the time spent to complete the deployment after T_{max}.
- P_t: desired (target) probability of success. Recall that *success* is defined by the condition that one or more robots have reached each end goal location in the environment. Given that the probability of successfully moving between two vertices is 1 only in the limit, then because of the given temporal deadline the probability of successfully completing the task will be lower than 1. P_t may therefore specify a lower bound on this value.

In the following we study a specific version of the problem obtained by fixing the various parameters described above. For a given environment represented by a graph G, our goal is to explore the interplay between K, T_{max} and P_t. Relevant questions are the following. For a given deadline T_{max}, how many robots are needed to obtain a probability of success exceeding P_t? Alternatively, one may ask what is the probability of success given that K robots are deployed. The answer to these questions are of course dependent not only on the environment G, but also on the deployment algorithm. We explore these aspects in the following sections.

III. Basic case

We start by considering a situation that can be considered the simplest. The study of this simple problem instance paves the way to more complex variants. Let S be the same for all edges and let the team consist of fully homogeneous robots all executing the same control algorithm. Robots do not communicate with each other, they have full knowledge of the environment, and know their position on the graph. Robots are initially all deployed in the same vertex $d \in V$. Let the temporal deadline T_{max} be a hard deadline, i.e., the deployment effort fails if it is not completed within time T_{max}. For every vertex $v \in V$, let $h(v)$ be the distance from d measured in hops (i.e., number of edges). The assumption that all edges share the same function S is very strong and will be relaxed in the next section. However, this basic case is useful to gain meaningful insights. Moreover, if one models the environment with a very dense graph, i.e., a graph where the distance between two adjacent nodes is roughly constant, and the ambient environment is uniform in the sense that the difficulty in moving between two vertices is more or less constant, then this model can be applicable.

Based on the above assumptions, the problem can be solved assuming each robot executes the simple algorithm sketched in Algorithm 1.

```
1 choose random vertex v ∈ V (uniform selection);
2 compute shortest path s_{d⇝v} from d to v;
3 for each edge along s_{d⇝v} choose travel time t;
4 travel to v according to the computed velocities;
```

Algorithm 1: The simplest deployment algorithm

Regarding the shortest path (line 2), *shortest* means *least number of hops*, so this can be computed with Dijkstra's algorithm. Because we assumed S is the same for all edges, this will eventually minimize the total travel cost. Before carrying out a performance analysis for the whole team, let us first see how the third step can be solved.

A. Picking a velocity profile

Assume $s_{d \rightsquigarrow v}$ has w edges. How should the robot choose a velocity for each edge so that the probability of success is maximized while satisfying the given deadline? Let $t_1, \ldots, t_w$ be the time spent on each of the w edges. Our goal is to solve the following optimization problem:

$$\begin{aligned} \max_{t_1,\ldots t_w} \quad & S(t_1)S(t_2)\ldots S(t_w) \\ \text{s.t.} \quad & t_1 + t_2 + \cdots + t_w \leq T_{max} \\ & t_i \geq 0 \quad 1 \leq i \leq w \end{aligned}$$

Note that the objective function is the probability of successfully navigating through all edges and is the product of the individual probabilities (assuming independence between edges). Under the assumption that S is a non-decreasing function of t, it is straightforward to show that the optimal solution is obtained with $t_1 = t_2 = \cdots = t_w = T_{max}/w$ and the value of the optimal solution (probability of success) is $S(T_{max}/w)^w$. The problem is more complicated if one assumes different edges are characterized by different functions S_i, and we address this later.

B. Team performance

We now explore the space describing the relationships between K, T_{max} and P_t. The deployment task will not be successfully completed if at least one vertex v is not selected by any robot. Therefore the first issue to consider is how large K needs to be guarantee that, with sufficiently high probability, every vertex is selected by at least one robot. This is a classic occupancy problem, see e.g., [6] or [2].

Given K robots and N locations, let X be the random variable describing the number of locations not chosen by any robot. Assuming the selection process is uniform (each vertex is chosen with probability $\frac{1}{N}$), then it is known that

$$\Pr[X = 0] = \sum_{i=0}^{N} (-1)^i \binom{N}{i} \left(1 - \frac{i}{N}\right)^K.$$

This formula helps to establish a lower bound on the failure probability because if $X > 0$ (one or more vertices were not selected by any robot) the task can not be correctly completed. Evidently, $\Pr[X > 0] = 1 - \Pr[X = 0]$. So $\Pr[X > 0]$ gives a lower bound on the probability failure.

For example, for a given P_t^* and N, it is possible to determine which values of K will not guarantee a success probability of at least P_t^* for the mere problem of spreading robots out efficiently. Figure 1 shows the lower bound error trend for $N = 10$ as K varies from 10 to 50.

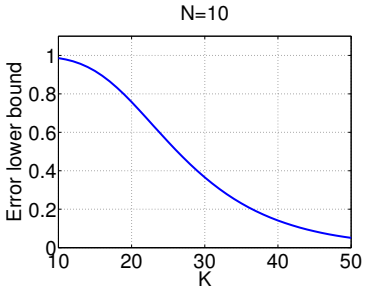


Fig. 1. $\Pr[X > 0]$ as a function of K for $N = 10$. The error is a lower bound inasmuch as it models failure due to robots not randomly spreading out to all vertices. However, the overall mission can also fail when robots do not manage to reach their selected target.

However, a sharper analysis is possible and needed because deployment may also fail even when a robot is assigned to every vertex. Define the following events:

- $\mathcal{A}$: "every vertex is selected by at least one robot";
- $\mathcal{B}$: "for every vertex $v \in V$, at least one of the robots that decided to go to v successfully manages to navigate to v".

The success of the mission is then the conjunction of events $\mathcal{A}$ and $\mathcal{B}$. The probability that both events occur can be written as

$$\Pr[\mathcal{A}, \mathcal{B}] = \Pr[\mathcal{B}|\mathcal{A}] \Pr[\mathcal{A}]. \tag{1}$$

We have already computed $\Pr[\mathcal{A}]$, because $\Pr[\mathcal{A}] = \Pr[X = 0]$. Therefore we need to find $\Pr[\mathcal{B}|\mathcal{A}]$. An exact computation of this quantity could be, in principle, addressed in the following way. For given values of K and N, let $\mathcal{S}_{K,N}$ be the set of all assignments of K robots to N rooms constrained to have at least one robot per room (i.e., we are conditioning on $\mathcal{A}$). Then we can write

$$\Pr[\mathcal{B}|\mathcal{A}] = \sum_{s \in \mathcal{S}_{K,N}} \Pr[\mathcal{B}|s, \mathcal{A}] \Pr[s|\mathcal{A}].$$

It immediately becomes evident this approach is not practical because $\mathcal{S}_{K,N}$ includes an exponential number of elements. Therefore, rather than trying to exactly compute this probability, we seek a bound. Let $\mathcal{B}_i$ ($1 \leq i \leq N$) be the event "at least one of the robots assigned to v_i successfully manages to reach it". Then $\mathcal{B} = \bigcap_{i=1}^{N} \mathcal{B}_i$. This quantity is in general difficult to compute because the various $\mathcal{B}_i$ are not independent and

so it does not nicely factorize. Instead consider $\bar{\mathcal{B}}$, the complement of $\mathcal{B}$. This is a *failure* probability and the following relationship therefore holds:

$$\Pr[\bar{\mathcal{B}}|\mathcal{A}] = 1 - \Pr[\mathcal{B}|\mathcal{A}] = \\ = \Pr\left[\bigcup_{i=1}^{N} \bar{\mathcal{B}}_i|\mathcal{A}\right] \leq \sum_{i=1}^{N} \Pr[\bar{\mathcal{B}}_i|\mathcal{A}]. \quad (2)$$

Hence we now need to determine $\Pr[\bar{\mathcal{B}}_i|\mathcal{A}]$. Recall that $h(v_i)$ is the number of hops from the deployment vertex d to vertex v_i. Then, based on the assumption that velocities are scheduled according to the method presented in the previous section, the probability that no robot successfully reaches vertex v_i can be computed as follows. Let n be the random variable indicating how many robots are trying to reach v_i. Then

$$\Pr[\bar{\mathcal{B}}_i|\mathcal{A}] = \\ = \sum_{k=1}^{K-(N-1)} \left(1 - S\left(\frac{T}{h(v_i)}\right)^{h(v_i)}\right)^k \Pr[n = k]. \quad (3)$$

This formula can be explained as follows. First, for a generic event A, assuming B_i is a partition of the total space, the total probability theorem states $\Pr[A] = \sum_i \Pr[A|B_i]\Pr[B_i]$. We partition the total space with respect to the number of robots n electing to reach v_i. Since we are conditioning on the event $\mathcal{A}$ (at least one robot per each vertex), n assumes values from $k = 1$ to $k = K - (N - 1)$ (note that n cannot be larger than $K - (N - 1)$ because at least $N - 1$ robots need to go to the remaining vertices.) For a given choice of $n = k$, the event $\bar{\mathcal{B}}_i$ occurs when all robots fail to reach their destination. Given that S provides the probability of success, the term

$$\left(1 - S\left(\frac{T}{h(v_i)}\right)^{h(v_i)}\right)^k$$

is the probability that all k robots assigned to v_i fail (assuming they act independently.) Now, $\Pr[n = k]$ can be computed considering that n follows a binomial distribution with parameters $K, 1/N$ conditioned on assuming values between 1 and $K-(N-1)$. Therefore, for $1 \leq k \leq K - (N - 1)$ its precise expression is

$$\Pr[n = k] = \eta \binom{K}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{K-k} \quad (4)$$

where η is a normalizing factor to ensure probabilities add up to 1. Finally, given that Eq. 2 provides a lower bound for $\Pr[\mathcal{B}|\mathcal{A}]$, combining Eq. 2 with Eq. 1 we get a lower bound for the success probability

How sharp is this bound? This question is in general hard to answer and depends on how *coupled* are the different events $\bar{\mathcal{B}}_i$. Stated differently, it depends on how loose is the bound $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$. So the bound will be informative when $\Pr[\bar{\mathcal{B}}_i \cap \bar{\mathcal{B}}_j \cap \dots]$ is small. In a later section we experimentally evaluate how informative is the bound given by Eq. 2.

IV. Dealing with different velocity profiles

One of the major assumptions we made in Section III is that all edges share the same function S expressing the tradeoff between velocity and probability of success. In this section we remove this assumption so that each edge is associated with a possibly different tradeoff function S_i. Under this more general model, given that a robot intends to move from d to v_i, it is not necessarily more convenient to start computing the shortest path from d to v_i using the number of hops (step 2 in Algorithm 1). In fact, one can easily build a simple graph instance where this idea does not work. Secondly, once a specific path $s_{d \rightsquigarrow v}$ has been determined, the approach outlined in Section III-A does not work anymore. Algorithm 2 is a revised version addressing these shortcomings, and we then discuss how its specific steps can be implemented.

```
1  choose random vertex v ∈ V (uniform selection);
2  P_min ← 0;
3  for each irreducible path s_{d⇝v} do
4      compute optimal velocity profile p for s_{d⇝v};
5      let P_succ be success probability of p;
6      if P_succ > P_min then
7          s_best ← s;
8          p_best ← p;
9          P_min ← P_succ;
10     end
11 end
12 travel to v along path s_best according to p_best;
```

Algorithm 2: Deployment algorithm with possibly different velocity/probability profiles for each edge.

The *for* loop starting at line 3 addresses the issue that the shortest path in terms of number of hops is not necessarily the best in terms of probability of success. Therefore an exhaustive search over the set of irreducible paths from d to v is performed. An *irreducible* path is here defined as a path without loops.[1] Irreducible paths can be computed in time $\mathcal{O}(N^2)$.

A. Picking a velocity profile with different S_i's

This subsection parallels Section III-A but considers the case where different S_i's are associated with the edges. This is the problem to be solved to implement

[1]Obviously, for the task we consider the presence of loops in a path is sub-optimal.

line 4 in Algorithm 2. Assuming $s_{d \rightsquigarrow v}$ has w edges, the optimization problem we need to solve is the following[2]

$$\begin{aligned} \max_{t_1,\dots t_w} \quad & S_1(t_1)S_2(t_2)\dots S_w(t_w) \\ \text{s.t.} \quad & t_1 + t_2 + \dots + t_w \leq T_{max} \\ & t_i \geq 0 \quad 1 \leq i \leq w. \end{aligned}$$

It is convenient to rewrite this problem using the following equivalent formulation:

$$\begin{aligned} \max_{t_1,\dots t_w} \quad & \log S_1(t_1) + \log S_2(t_2) + \dots + \log S_w(t_w) \\ \text{s.t.} \quad & t_1 + t_2 + \dots + t_w \leq T_{max} \\ & t_i \geq 0 \quad 1 \leq i \leq w. \end{aligned}$$

Problems like this can be solved in the continuum, but their solution is difficult. Alternatively, by introducing a discretization step it is possible to reduce it to a known combinatorial problem. Let Δt be a discretization step for the temporal dimension and let $L = \lfloor T_{max}/\Delta t \rfloor$ be the new temporal deadline in the discretized space. The problem is therefore to decide how to allocate the L time steps on the w edges. In other words, $t_i \in \mathbb{N}^+$ in the discretized space. The problem can be reduced to the multiple-choice knapsack problem [5] as follows. Let e_1 be the first edge along path $s_{d \rightsquigarrow v}$. During the optimization one can decide to allocate $t_1 \in \{1, 2, \dots, L\}$ time steps to e_1. For every possible choice of t_1, a corresponding contribution to the objective function is readily available, i.e., if $t_1 = k$ then its contribution to the objective function will be $\log S_1(k\Delta t)$. Therefore for edge e_1 one can build a set of couples $(k, \log S_1(k\Delta t))$ with $1 \leq k \leq L$, $k \in \mathbb{N}$. Similarly, for the next edge e_2 one can build a set of couples $(k, \log S_2(k\Delta t))$, and so on. In order to find the maximum for the objective function we need to pick one integer for every edge, subject to the constraint that the sum must be not larger than L. This is precisely an instance of the multiple-choice knapsack problem, where every edge defines a class. Using knapsack terminology, for every couple $(k, \log S_i(k\Delta t))$ k is the weight whereas $\log S_i(k\Delta t)$ is the value. The multiple choice knapsack problem is NP-Hard but can be solved in pseudo-polynomial time with a dynamic programming approach.

B. Team performance

In Section V it will be shown that the bound derived in Section III-B offers a good approximation of the actual deployment algorithm when all edges share the same S function. It would therefore be useful to derive a similar bound for the case where different edges are associated with different S_i's. The reasoning presented in Section III-B holds also for the case of different S_i's until Eq. 3, where the hypothesis that all edges have the same S function is used to compute $\Pr[\bar{\mathcal{B}}_i|\mathcal{A}]$. However, a bound can be inferred as follows. Our goal is to bound from above $\Pr[\bar{\mathcal{B}}_i|\mathcal{A}]$, so that we can further extend the right hand side of Eq. 2. In other words, if for every i we can find a value C_i such that $\Pr[\bar{\mathcal{B}}_i|\mathcal{A}] < C_i$, then we can replace Eq. 2 with

$$\begin{aligned} \Pr[\bar{\mathcal{B}}|\mathcal{A}] = 1 - \Pr[\mathcal{B}|\mathcal{A}] = \Pr\left[\bigcup_{i=1}^{N} \bar{\mathcal{B}}_i|\mathcal{A}\right] \leq \\ \leq \sum_{i=1}^{N} \Pr[\bar{\mathcal{B}}_i|\mathcal{A}] \leq \sum_{i=1}^{N} C_i. \end{aligned} \tag{5}$$

Consider a robot going to vertex v_i and let us build a worst case instance to bound the probability of success from below (and then bound the probability of failure from above to get C_i). In order to do so, we need to make a further hypothesis. With reference to Algorithm 2, let us put a bound on the maximum length of the irreducible paths we consider (lines 3 and 4). More precisely, indicating again with $h(v_i)$ the number of hops from the deployment vertex d to vertex v_i, we constrain the algorithm not to consider irreducible paths with more than $2h(v_i)$ edges[3]. Let us now assume all edges are characterized by the worst among all the S_i functions. Call this function S_w. Then the probability of success for a single robot is larger than $S_w(T/2h(v_i))^{h(v_i)}$. This is in fact the probability of the worst case instance, i.e., the case where the robot has to traverse the largest number of edges and each edge is characterized by the worst performance. Given a lower bound on the probability of success, we can get an upper bound on the probability of failure, i.e., we can rewrite Eq. 3 as follows:

$$\begin{aligned} \Pr[\bar{\mathcal{B}}_i|\mathcal{A}] \leq C_i = \\ \sum_{k=1}^{K-(N-1)} \left(1 - S_w\left(\frac{T}{2h(v_i)}\right)^{2h(v_i)}\right)^k \Pr[n = k]. \end{aligned} \tag{6}$$

Plugging Eq. 6 into Eq. 5 we therefore derive a new upper bound for $\Pr[\bar{\mathcal{B}}_i|\mathcal{A}]$ and then a lower bound for $\Pr[\mathcal{B}|\mathcal{A}]$. From there the same reasoning applied in Section III-B applies as is, and we conclude with a lower bound for $\Pr[\mathcal{A}, \mathcal{B}]$.

Two significant issues remain. The first one is the definition S_w, i.e., the worst among the S_i functions.

[2]To be precise, writing S_1 is not correct, as this is not the function associated with e_1 but rather the function associated with the first edge along the path $s_{d \rightsquigarrow v}$. The same is true for S_2, e_2, and so on. However, to avoid introducing further notation we accept this slight abuse in notation.

[3]While this constraint is introduced primarily to obtain an explicit bound, this also makes sense from a practical perspective. Taking detours may be advantageous, but it appears that taking very long detours (longer than $2h(v_i)$) is unlikely to help.

Given that in the worst case instance all edges share the same tradeoff between velocity and probability of success, we have reduced our problem to an instance of the formerly studied case where all S functions are the same. Then the optimal solution will indeed be the one where the robot spends $T_{max}/2h(v_i)$ time on each edge, as implied by Eq. 6. Therefore S_w is defined as the function with the lowest value for $T_{max}/2h(v_i)$. The second issue concerns the tightness of the bound we derived, given that we have introduced further approximations. We consider this through simulations in Section V.

C. Remarks

1) Characteristics of the simplest algorithm: the two algorithms we presented are most likely the simplest one can imagine. Still, as we have shown, characterizing their performance in analytic terms is not immediate and one may conjecture that it will be much harder for more complex deployment approaches. The algorithms, however, are also the most scalable and robust. In fact, they are fully distributed and assume no mutual knowledge or exchange of information, either before or during the mission. Assuming the availability of a large number of identical robots all controlled by the same algorithm, they can be added to the team being deployed without any reconfiguration effort. And, of course, the simplest algorithm can serve as a yardstick to compare against more sophisticated approaches relying on communication and explicit coordination.

2) Possible improvements: One could conjecture that the presented approach could be improved by sampling vertices with a non-uniform distribution. Indeed, one could think that more robots should be sent towards far vertices that are harder to reach, and less robots should be directed to nearby ones. The idea is in principle sound, but yields a true advantage only when the team is operating under a very demanding temporal deadline T_{max}. In that case, the term $\Pr[\mathcal{B}|\mathcal{A}]$ in Eq. 1 dominates and the idea pays off. However, any non-uniform distribution will decrease the term $\Pr[\mathcal{A}]$ in Eq. 1. Hence, when T_{max} is not too strict the approach may actually be counterproductive. It would be useful to find a criterion suggesting when one should switch to a uniform vs. non-uniform sampling based on T_{max} and G, but a closed form solution is not evident.

V. Simulations

In this section we present simulations for different deployment scenarios. Graphs are extracted from artificial maps or occupancy grid maps built with state-of-the art SLAM software [3] and publicly available data sets[4].

[4]Code and datasets producing the results presented in this section are available for download on `http://robotics.ucmerced.edu`.

A. Bound tightness when all edges share the same S function

The first experiment aims to experimentally evaluate the quality of the bound we derived in Section III-B. In this case we assume S is the same for all edges, and in Figure 2 we plot the specific function we use.

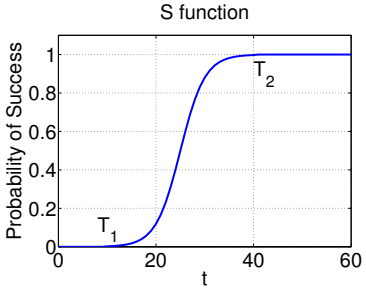


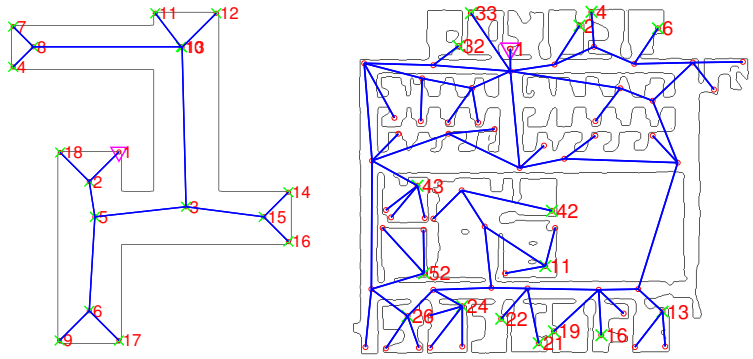
Fig. 2. S, the relationship between time spent and probability of success used in simulations where all edges share the same function.

Note that the probability of successfully completing a transition between two adjacent vertices is 0 for $t < T_1 = 10$ and is greater than 0.9975 for $t > T_2 = 40$.

Two different maps displayed in Figure 3 are used. The left one, sketched by hand, has 18 vertices, and robots have to be deployed in 17 vertices (all but the deployment site) within a temporal deadline of 220 time units. Robots are initially deployed in vertex 1. The second map consists of 70 vertices, and robots have to reach a subset of 17 vertices (those displayed with a number). The common deployment vertex is 1, and the temporal deadline is 3600 time units. The second map was built using the publicly available *sdr40* dataset [8]. These two maps are chosen as representative examples of simple and complex environments. Similar results were obtained for other datasets omitted from this paper for lack of space.

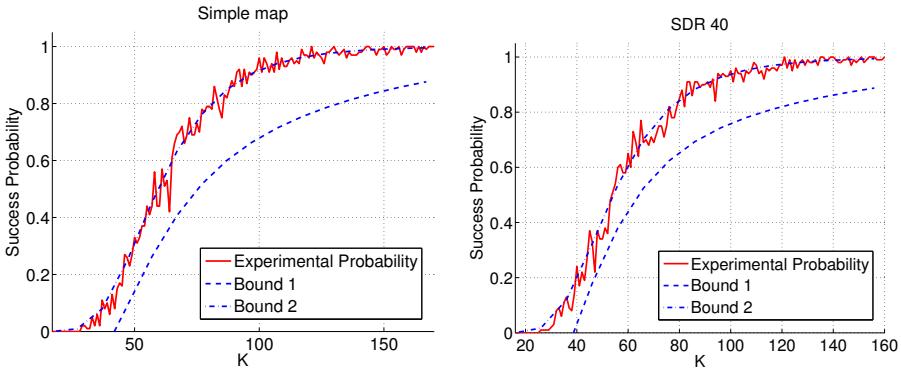
We compute the trend of the success rate lower bound for increasing values of K, starting with $K = N$. Then, we simulate deployments of robots for the same range of K values and we experimentally measure the average success rate. While performing this Monte-Carlo simulation we simulate 100 deployments for each value of K. Figure 4 contrasts the two trends we obtained for the two maps.

The figure shows that for the maps we consider the approximate bounds we derived accurately predict the actual performance. These curves can be interpreted as *performance curves*, because they capture the tradeoff between the the number of robots K, and the probability of success P_t. The curves are necessarily parametric in T_{max} and depend also on the underlying graph. They can be used to make decisions about the size of the robot team needed to perform a certain task, or to



(a) Simple map (b) Sdr40 map

Fig. 3. The two maps used to experimentally evaluate the analytic bounds. The left one is a simple environment drawn by hand, whereas the right one is built from a publicly available dataset. The deployment vertex is marked with a pink triangle, whereas goal vertices are indicated by green crosses. Edges between vertices only indicate that a path exists, but they do not imply it is a straight line.



(a) Simple map (b) Sdr40 map

Fig. 4. Comparison between approximate bounds for success probability (blue lines) and experimental success rate when all edges share the same S function. Because the computation of binomial coefficients for large values of K and k in Eq. 4 exceeds the computer's precision, two different approximations for $\binom{K}{k}$ are used and shown (see appendix). For every value of K *Bound 1* approximates the probability of success from below, whereas *Bound 2* approximates the probability of success from above. For large values of K the blue curves converge.

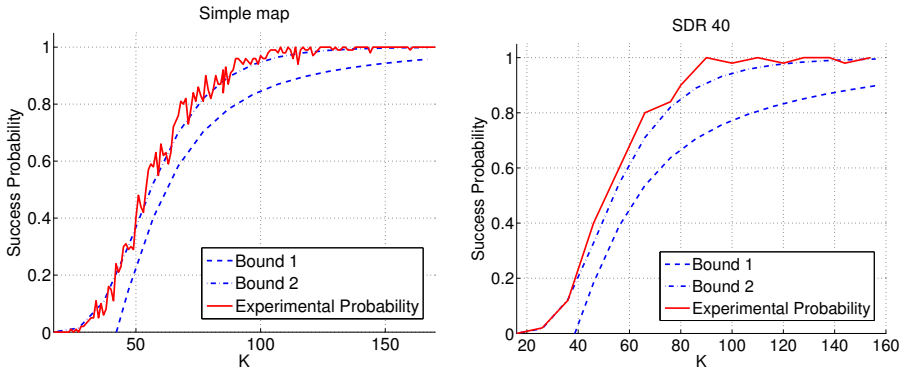
predict the probability of success for a given number of robots deployed in a specific map and subject to a given temporal deadline.

B. Bound tightness when edges have different S functions

The second experiment aims to evaluate whether the same predictions are possible when different S functions are associated with different edges. To this end, we use the same sigmoid function shown in Figure 2, but its parameters are tuned based on the distance between vertices. In particular, T_1 is proportional to the distance between the vertices. If the straight path between two vertices does not intersect any obstacle, then $T_2 = 2T_1$, otherwise $T_2 = 10T_1$. While this is clearly a crude approximation, it produces rich enough problem instances with highly diverse S functions associated with the various edges. This setup leads to instances of the multiple choice knapsack problem that can be exactly solved in reasonable time. For very large instances one could still apply the same method using an approximate algorithm to compute the solution to the knapsack problem. Figure 5 displays the same quantities shown in Figure 4. The charts confirm that the approximate bounds we derived are excellent predictors even when different S functions are associated with the various edges.

VI. Related Work

Numerous multirobot deployment problems have been studied in the past, but the problem of balancing risk and velocity has been rarely addressed. Due to space limitations, we here provide just a few pointers to former literature analyzing different aspects of deployment problems. Significant work has been devoted to the problem of deployment while maximizing objective functions like coverage or, more generally, sensor data quality. The seminal work by Bullo, et al. [1] and the



(a) Simple map (b) Sdr40 map

Fig. 5. Comparison between approximate bounds for success probability and experimental success rate when edges have different S functions.

numerous subsequent papers embracing the distributed Voronoi approach proposed therein fall under this category. When robots are deployed to install a communication network, emphasis has been devoted to maintaining connectivity or offering high communication bandwidth [7], [9]. This is usually the same standpoint assumed in the sensor network community. In urban search and rescue, multirobot deployment is sometimes aided by a human supervisor occasionally taking over when robots encounter difficulty [10].

To the best of our knowledge, the specific problem we have addressed in this manuscript is novel.

VII. Conclusions

In this paper we have introduced and analyzed the tradeoff between velocity and probability of success for the task of multi-robot deployment. This standpoint has been scarcely considered in the past, and we have identified various parameters that generate numerous different instances of this problem. For the simplest case (map known a priori, and robots able to self localize), we have analyzed a simple, robust, communication free, scalable algorithm. Because of the inherent complexity of an exact analysis, we derived approximate bounds leading to performance curves that can be used for analysis and design purposes. Our theoretical findings have been corroborated by simulations demonstrating that the approximate bounds are tight, showing the utility of the associated performance curves.

In the future we intend to extend this framework in various ways. For example, we will consider the case where the map is only partially known a-priori, or the situation where robots cannot reliably self-localize while moving towards their goal. In these cases we anticipate the use of communication between robots will be crucial. Moreover, we are investigating how to derive the S functions from robots moving in various reference test environments.

We emphasize that the analytical framework is general in the sense of modeling state transitions with probability of success vs. time curves. So, for example, the analysis could be applied to manipulation with risk of collision.

Appendix

For large values of K, the value of $\binom{K}{k}$ exceeds the computer's precision. Therefore in Fig. 4 and Fig. 5 we use the following inequalities

$$\left(\frac{K}{k}\right)^k \leq \binom{K}{k} \leq \frac{K^k}{k!}$$

to plot the two blue curves.

Acknoweldgments

The authors thank Dr. Nicola Basilico for the solution of the multiclass knapsack problem.

References

[1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing netorks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[2] W. Feller. *An Introduction to Probability Tehory and Its Applications*, volume 1. John Wiley, 1968.

[3] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping iwht Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):36–46, 2007.

[4] A. Kolling and S. Carpin. Extracting surveillance graphs from robot maps. In *Proceedings of IROS*, pages 2323–2328, 2008.

[5] S. Martello and P. Toth. *Knapsack problems*. John Wiley and Sons, 1990.

[6] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[7] Y. Pei and M.W. Mutka. Steiner traveler: relay deployment for remote sensing in heterogeneous multi-robot exploration. In *Proceedings of ICRA*, pages 1551–1556, 2012.

[8] Radish – the robotics data set repository. http://radish.sourceforge.net, 2009.

[9] M. Rooker and A. Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 2007.

[10] J. Wang, M. .Lewis, and P. Scerri. Cooperating robots for search and rescue. In *Proceeedings of Autonomous Agents and MultiAgent Systems*, 2004.