Unifying Geometric, Probabilistic, and Potential Field Approaches to Multi-Robot Deployment

Mac Schwager
GRASP Lab
University of Pennsylvania
3330 Walnut St, Philadelphia, PA 19106
schwager@seas.upenn.edu

Daniela Rus
Computer Science and Artificial
Intelligence Laboratory, MIT, 32 Vassar St
Cambridge, MA 02139, USA
rus@csail.mit.edu

Jean-Jacques Slotine
Nonlinear Systems Laboratory, MIT
77 Massachusetts Ave
Cambridge, MA 02139, USA
jjs@mit.edu

August 8, 2010

Abstract

This paper unifies and extends several different existing strategies for deploying groups of robots in an environment. A cost function is proposed that can be specialized to represent widely different multi-robot deployment tasks. It is shown that geometric and probabilistic deployment strategies that were previously seen as distinct are in fact related through this cost function, and differ only in the value of a single parameter. These strategies are also related to potential field based controllers through the same cost function, though the relationship is not as simple. Distributed controllers are then obtained from the gradient of the cost function and are proved to converge to a local minimum of the cost function. Three special cases are derived as examples: a Voronoi based coverage control task, a probabilistic minimum variance task, and a task using artificial potential fields. The performance of the three different controllers are compared in simulation. A result is also proved linking multi-robot deployment to nonconvex optimization problems, and multi-robot consensus (i.e. all robots moving to the same point) to convex optimization problems, which implies that multi-robot deployment is inherently more difficult than multi-robot consensus.

1 Introduction

One of the fundamental problems of multi-robot control is how to deploy a group of robots over an environment to carry out sensing, surveillance, data collection, or distributed servicing tasks. We use the term deployment to encompass any task in which the robots move out over an environment to reach some final fixed configuration, for example, coverage, herding, or formation control. A number of control strategies have been proposed to accomplish different multi-robot deployment tasks in a distributed and efficient way. In this paper we introduce a unifying principle that ties together many of these strategies. We show that many of the existing methods can be described as

special instances of gradient descent on a cost function. We propose a cost function that specializes to give several common algorithms for multi-robot deployment, including Voronoi-based controllers for sensor coverage, as in [8], controllers based on probabilistic models, as in [19], and artificial potential field-based controllers for herding, flocking, and consensus,[1] as in [10, 14, 15, 27].

Controllers for multi-robot deployment are useful for many applications involving distributed sensing and distributed actuation. For example controllers can be used to deploy underwater robots evenly over a coral reef to monitor coral health, or to deploy wheeled robots with cameras to spread out over a room for surveillance. Groups of robots can also be deployed to carry out actuation tasks, for example oil clean-up robots can be deployed over an oil spill, or de-mining robots can be positioned to service a mine field. We describe a framework that is relevant to both sensing and actuation tasks. We argue that Voronoi methods are best suited to distributed *actuation* tasks, while a continuous approximation to the Voronoi decomposition is more appropriate for distributed *sensing* tasks. Furthermore, even in distributed actuation tasks, using a continuous approximation to the Voronoi cell improves the robustness of the controller to numerical integration errors.

The controllers we describe are provably convergent, robust to individual robot failures, and can adapt to environments that change slowly with respect to the speed of the robots. They require that robots know the geometry of the environment and they know their own position in it using, for example, GPS or an indoor localization system. We also discuss how to accommodate the constraints of a communication network topology, but do not analyze this aspect of the problem in detail. In our setting, all the robots have identical dynamics and capabilities.

Related Work Cortés et al. [8] introduced a controller for multi-robot sensor coverage that works by continually driving the robots toward the centroids of their Voronoi cells. This inherently geometric strategy has seen many recent extensions to robots with a limited sensing radius in [7], to heterogeneous groups of robots and nonconvex environments in [3, 21], and to incorporate learning of unknown environments in [24]. A recent text that presents much of this work in a cohesive fashion is [4] and an excellent overview is given in [20]. Coverage controllers also have been successfully implemented on robotic systems in [22, 23]. In this work we adopt notational conventions from the Voronoi based coverage control literature. Other common methods for multi-robot deployment take a probabilistic perspective. For example [19] proposes an algorithm for positioning robots to maximize the probability of detecting an event that occurs in the environment. Distributed dynamic vehicle routing scenarios are considered in [1, 26], in which events occur according to a random process and are serviced by the robot closest to them. Another common method is for robots to drive away from or towards one another by following the negative gradient of artificial potential fields. These have been used for sensor coverage in [14], flocking and herding in [10, 27], and consensus (or rendezvous) in [15]. Despite the rather different models and objectives in these works, there are two common points which motivate us to find a unifying principle: 1) they all rely upon an optimization, and 2) they all use controllers that solve this optimization through the evolution of a dynamical system.

Some existing approaches do not fit under the framework we propose in this paper. A significant body of work has looked at multi-agent deployment as a motion planning problem. A survey of this work can be found in [6], and some significant contributions can be found in, for example, [5, 18] and the citations therein. Other authors have proposed information theoretic algorithms for sensor

[1] We use the term consensus in this paper to mean that all robots drive to a common point in the environment. This is also commonly called rendezvous in the literature.

networks which consider placing static sensors sequentially rather than driving robots with sensors using a distributed controller. Works such as [12,16] position sensor nodes to maximize information for the sake of estimating a Gaussian random process in the environment.

Contributions In the present work we focus on multi-agent deployment as an optimization problem. This is advantageous because it is amenable to geometric, probabilistic, and potential field interpretations, all of which have been seen in a separate light in the past. Our optimization approach ties together many existing methods that were previously seen as unrelated. Specifically, our contributions are:

1. We propose a cost function, putting particular emphasis on the role of a *mixing function*, a previously unrecognized component that captures critical assumptions about the deployment task.

2. We introduce a family of mixing functions with a free parameter, α, and show that different values of the parameter correspond to different assumptions about the deployment task, specifically showing that a minimum variance solution (i.e. a probabilistic strategy) is obtained with a parameter value of $\alpha = -1$, and Voronoi coverage (a geometric strategy) is recovered in the limit $\alpha \to -\infty$. A broad family of potential field based herding and consensus controllers are recovered when $\alpha = 1$, and by specializing two other components of the cost function.

3. We prove a new result linking the convexity of a cost function to the multi-agent phenomenon of consensus. We show that deployment tasks are fundamentally different from consensus, and that they require the optimization of a nonconvex cost function. This suggests that gradient descent controller designs, which are pervasive in the literature, can only be proved to converge to a local minimum of the cost function.

The paper is organized as follows. In Section 2 we introduce the cost function, describing the purpose of each of its parts including the mixing function. We then produce a class of provably stable distributed coverage controllers by taking the gradient of the cost function. In Section 3 we derive three special cases of the controller; a Voronoi controller, a minimum variance controller, and a potential field controller. Section 4 presents our results on the relation between the convexity of a cost function, and multi-robot consensus. Simulation results are given in Section 5 and conclusions are in Section 6.

2 Generalized Deployment

In this section we introduce a general multi-robot cost function. We will use this cost function to define a new class of multi-robot controllers by introducing a *mixing function*, which describes how information from different robots should be combined. We use the cost function to derive a stable gradient descent controller.

2.1 Cost Function

Let there be n robots[2], and let robot i have a state $p_i \in \mathcal{P} \subset \mathbb{R}^{d_p}$, where $\mathcal{P}$ is the state space of a robot, and d_p is the dimension of the space. The vector of all robot states is denoted $P = [p_1^T, \ldots, p_n^T]^T \in \mathcal{P}^n$, and we will call P the *configuration* of the robots. We want our robots to deploy over a bounded region $Q \subset \mathbb{R}^{d_q}$, which may or may not be the same as the state space $\mathcal{P}$ of the robots. For example, the robots may be constrained to move in the space over which they are deployed, so that $\mathcal{P} = Q$ as in [8], or the robots may hover over a planar region that they cover with cameras, so $\mathcal{P} \subset \mathbb{R}^3$ and $Q \subset \mathbb{R}^2$, as in [22].

For each robot, a cost of sensing, or servicing, a point $q \in Q$ is given by a function $f(p_i, q)$. For simplicity of analysis we assume that $f(p_i, q)$ takes on only non-negative values, and that it is differentiable with respect to p_i.[3] The sensor measurements of the n robots are combined in a function $g(f(p_1, q), \ldots, f(p_n, q))$, which we will call the *mixing function*. The mixing function embodies assumptions about the task; that is, by changing the mixing function we can derive Voronoi based coverage control, probabilistic coverage control, and a variety of other kinds of distributed controllers.

Combining these elements, we propose to use a cost function of the form

$$\mathcal{H}(P) = \int_Q g(f(p_1, q), \ldots, f(p_n, q))\phi(q)\, dq. \tag{1}$$

where $\phi : \mathbb{R}^{d_q} \mapsto \mathbb{R}_{>0}$ is a weighting of importance over the region Q (we use the notation $\mathbb{R}_{>0}$ to mean the set of positive real numbers and $\mathbb{R}^d_{>0}$ the set of vectors whose components are all positive, and likewise for $\mathbb{R}_{\geq 0}$ and $\mathbb{R}^d_{\geq 0}$). Intuitively, the cost of the group of robots sensing at a single arbitrary point q is represented by the integrand $g(f(p_1, q), \ldots, f(p_n, q))$. Integrating over all points in Q, weighted by their importance $\phi(q)$ gives the total cost of a configuration of the robots. We want to find controllers that stabilize the robots around configurations P^* that minimize $\mathcal{H}$. We will see in Section 4 that for coverage, and many other multi-agent problems, $\mathcal{H}$ is necessarily nonconvex, therefore gradient based controllers will yield *locally* optimal robot configurations. The cost function (1) will be shown to subsume several different kinds of existing multi-robot deployment cost functions. Drawing out the relations between these different deployment algorithms will suggest new insights into when one algorithm should be preferred over another.

Although this cost function is general enough to encompass a broad range of deployment tasks, there are some constraints inherent in the way we have formulated $\mathcal{H}$. Firstly, we implicitly assume that the robots have identical capabilities, since $f(p_i, q)$ is the same function for all robots. Also, since $f(p_i, q)$ does not depend on the positions of the other robots, it cannot capture the effect of one robot obscuring or interfering with another.

2.2 Mixing Function

The mixing function $g_\alpha : \mathbb{R}^n_{\geq 0} \mapsto \mathbb{R}$ describes how information from different robots should be combined to give an aggregate cost of the robots sensing at a point q. This is shown graphically in

[2] We will use the term robot throughout, though the framework is suitable for general mobile sensing agents, including biological ones.

[3] This requirement can be generalized considerably as in [7] to the case where $f(p_i, q)$ is piece-wise continuous with a finite number of jump discontinuities. A finite sensor footprint can be modeled with a single jump discontinuity.

Fig. 1 where the overlap of the two sensors is shown for illustrative purposes as the intersection of two circles. We propose a mixing function of the form

$$g_\alpha(f_1, \ldots, f_n) = \big(\sum_{i=1}^{n} f_i^\alpha\big)^{\frac{1}{\alpha}}, \tag{2}$$

with a free parameter $\alpha \in \mathbb{R}$. The arguments $f_i \geq 0$ are real valued, and in our context they are given by evaluating the sensor function $f(p_i, q)$, hence the notation f_i. To be precise, when $\alpha < 0$ the expression in (2) is undefined if $f_j = 0$ for some j, therefore in this case we define g_α by its limit, $g_\alpha(f_1, \ldots, 0, \ldots, f_n) = \lim_{f_j \to 0} \big(\sum_{i=1}^{n} f_i^\alpha\big)^{\frac{1}{\alpha}} = 0$.

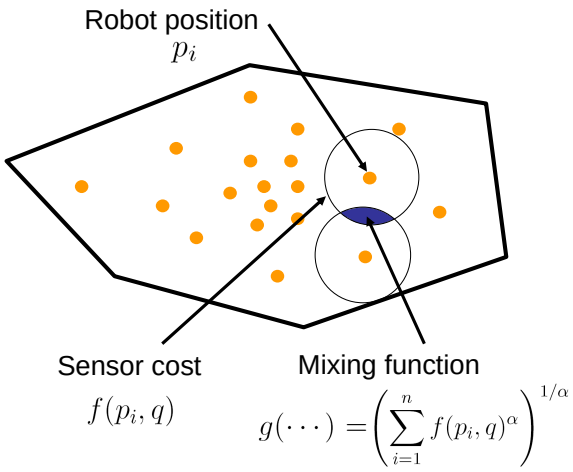
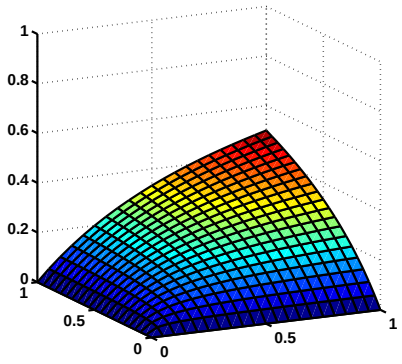


Figure 1: The mixing function is illustrated in this figure. The mixing function determines how information from the sensors of multiple robots is to be combined, shown graphically as the intersection of the two circles in the figure.

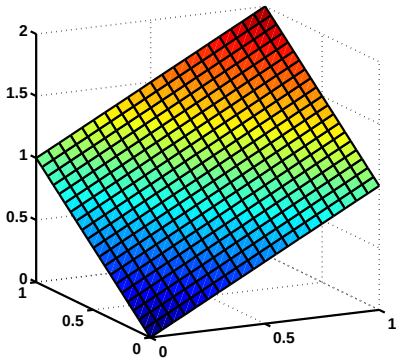
This mixing function has several important properties. Firstly, notice that for $\alpha \geq 1$ it is the p-norm of the vector $[f_1 \cdots f_n]^T$. Specifically, it is convex for $\alpha \geq 1$ and as $\alpha \to \infty$, $g_\alpha(\cdot) \to \max_i(\cdot)$, which is the ℓ^∞ norm. However, we are interested in the regime where $\alpha < 1$. In this case $g_\alpha(\cdot)$ is not a norm because it violates the triangle inequality. In this regime it is also nonconvex, leading to a nonconvex cost function, which is a necessary attribute of deployment problems, as we will prove in Section 4. One can readily verify[4] that as $\alpha \to -\infty$, $g(\cdot) \to \min_i(\cdot)$. From an intuitive point of view, with $\alpha < 1$, $g_\alpha(\cdot)$ is smaller than any of its arguments alone. That is, the cost of sensing at a point q with robots at p_i and p_j is smaller than the cost of sensing with either one of the robots individually. Furthermore, the decrease in g_α from the addition of a second robot is greater than that from the addition of a third robot, and so on. There is a successively smaller benefit to adding more robots. This property is often called supermodularity, and has been exploited in a rather different way in [16]. Surface plots of $g_\alpha(f_1, f_2)$ for $\alpha = -1$, 1, and 2 are shown in Figures 2(a), 2(b), and 2(c), respectively, and the decrease in $g_\alpha(\cdot)$ as the number of arguments grows is shown in

[4]We know $\lim_{\beta\to\infty}[\sum_i h_i^\beta]^{1/\beta} = \max_i h_i$. Write $\lim_{\alpha\to-\infty}[\sum_i f_i^\alpha]^{1/\alpha}$ as $\lim_{\beta\to\infty}[[\sum_i h_i^\beta]^{1/\beta}]^{-1}$ with $h_i = 1/f_i$ and $\beta = -\alpha$. We have $\lim_{\beta\to\infty}[[\sum_i h_i^\beta]^{1/\beta}]^{-1} = [\max_i h_i]^{-1} = [\frac{1}{\min_i f_i}]^{-1} = \min_i f_i$.

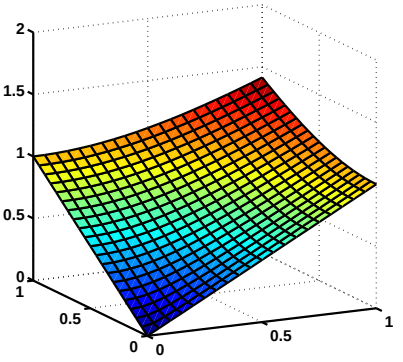
Figure 2(d). In this work we consider the number of robots to be fixed, but it is useful to illustrate the supermodularity property of the mixing function by considering the successive addition of new robots.



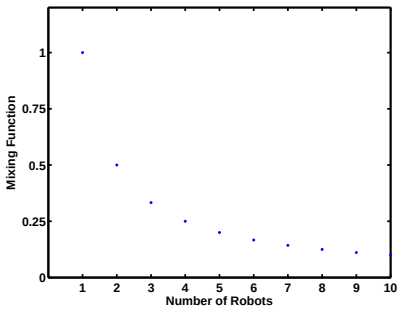
(a) Mixing Function ($\alpha = -1$)



(b) Mixing Function ($\alpha = 1$)



(c) Mixing Function ($\alpha = 2$)



(d) Supermodularity ($\alpha = -1$)

Figure 2: The proposed mixing function with $\alpha = -1$, 1, and 2 is shown in 2(a), 2(b), and 2(c), respectively. The function is convex for $\alpha \geq 1$ and nonconvex otherwise. The nonlinear decrease in the function as more sensors are added, a property known as supermodularity, is shown in Figure 2(d).

Including this mixing function in the cost function from (1) gives

$$\mathcal{H}_\alpha = \int_Q \Big(\sum_{i=1}^{n} f(p_i, q)^\alpha \Big)^{\frac{1}{\alpha}} \phi(q)\, dq. \tag{3}$$

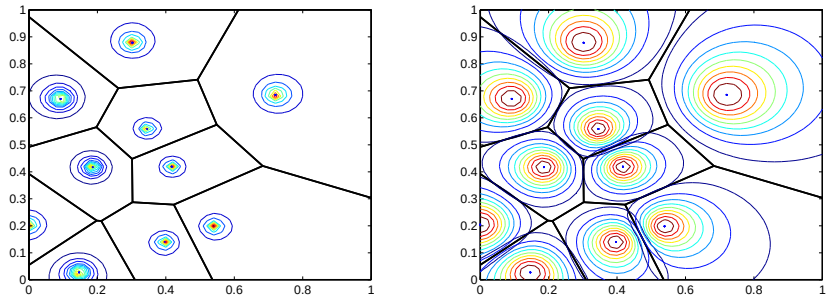
To model scenarios with a finite sensor footprint, we can also let $f(p_i, q)$ be infinite in some areas, in which case to keep the cost function bounded and differentiable it becomes necessary to include a prior w in the mixing function, yielding the variation $g_\alpha(f_1, \ldots, f_n) = \big(\sum_{i=1}^{n} f_i^\alpha + w^\alpha \big)^{1/\alpha}$. An application of this case was explored in [22] to design a controller for positioning multiple flying robots with downward facing cameras.

2.3 Gradient Control

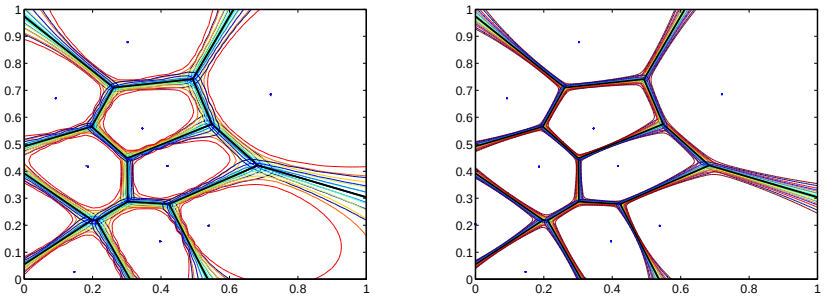
In order to derive a gradient descent controller, we take the derivative of the cost function $\mathcal{H}_\alpha$ with respect to the state of robot i to get

$$\frac{\partial \mathcal{H}_\alpha}{\partial p_i} = \int_Q \left(\frac{f(p_i, q)}{g_\alpha} \right)^{\alpha - 1} \frac{\partial f(p_i, q)}{\partial p_i} \phi(q)\, dq.$$

To provide some intuition about the meaning of this function, notice that in the case that $f(p_i, q)$ is strictly increasing, the function inside the integral $(f(p_i, q)/g_\alpha)^{\alpha-1}$ gives an approximation to the indicator function[5] of the Voronoi cell of agent i, the approximation improving as $\alpha \to -\infty$. This is shown graphically in Fig. 3. It can be readily verified that this function is continuous, and that at $f(p_i, q) = 0$ it takes the value 1, and at $f(p_j, q) = 0$ and $j \neq i$ it takes the value 0. For



(a) $\alpha = -.5$ (b) $\alpha = -1$



(c) $\alpha = -5$ (d) $\alpha = -10$

Figure 3: Contour plots of $(f(p_i, q)/g_\alpha)^{\alpha-1}$ are shown for a configuration of ten agent positions. The Voronoi tessellation is shown as well for comparison. As the parameter α approaches $-\infty$, $(f(p_i, q)/g_\alpha)^{\alpha-1}$ becomes closer to the indicator function of the Voronoi cell V_i.

simplicity, we choose the function $f(p_i, q)$ to be

$$f(p_i, q) = \frac{1}{2}\|q - p_i\|^2, \quad \text{so that} \quad \frac{\partial f(p_i, q)}{\partial p_i} = -(q - p_i).$$

Other choices of $f(p_i, q)$ were investigated in [7] and could be used here as well, including functions with discrete jumps that model a finite senor footprint. This function represents the cost of a

[5]The indicator function for a set $S \subset Q$ returns 1 for $q \in S$, and 0 otherwise.

single robot i sensing at the position q. Therefore the quadratic form is appropriate for light based sensors, such as cameras or laser scanners. Light intensity drops off as the inverse square of the distance from the source, so it is reasonable for the cost to be proportional to the square of the distance. For tasks in which robots have to drive to a point q for servicing, and we want the cost to be proportional to the distance traveled, it would be more appropriate to use $f(p_i, q) = \|q - p_i\|$, for example.

We propose to use a gradient-based controller

$$\dot{p}_i = -k\frac{\partial \mathcal{H}_\alpha}{\partial p_i} = k\int_Q \left(\frac{f(p_i,q)}{g_\alpha}\right)^{\alpha-1}(q-p_i)\phi(q)\,dq, \tag{4}$$

where $k > 0$ is a positive control gain. We assume that the robots have integrator dynamics, $\dot{p}_i = u_i$, so we can control their velocity directly. We have found experimentally, in [23] for ground vehicles and [22] for quadrotor air vehicles, that this is a fair assumption as long as a fast inner control loop is in place to track the desired $\dot{p}_i$.

We can equivalently express the n coupled equations in (4) as a single equation using the configuration vector P as

$$\dot{P} = -k\frac{d\mathcal{H}_\alpha}{dP}.$$

Our multi-robot system is therefore a gradient system, meaning the right hand side of the governing differential equation is proportional to the negative gradient of the scalar valued cost function $\mathcal{H}_\alpha$. Gradient systems have particularly simple and powerful convergence and stability properties, the most important of which will be given here.

Theorem 1 (Global Convergence) *Let $\Omega = \{P^* \mid d\mathcal{H}_\alpha/dP \mid_{P^*} = 0\}$ be the set of all critical points of $\mathcal{H}_\alpha$. All trajectories of the system $\dot{P} = -kd\mathcal{H}_\alpha/dP$ converge asymptotically to Ω.*

Proof: The theorem follows as a corollary to LaSalle's Invariance Principle [17, 25]. Let $\mathcal{H}_\alpha$ be the Lyapunov function candidate. Then $\dot{\mathcal{H}}_\alpha = -k\|d\mathcal{H}_\alpha/dP\|^2 \leq 0$, and since $\mathcal{H}_\alpha$ is radially unbounded, the trajectories of the system are bounded, therefore by LaSalle's Invariance Principle all trajectories converge to the largest invariant set contained in Ω. By the definition of the dynamics, Ω itself is an invariant set, therefore all trajectories converge to Ω. □

Remark 1 *This result does not necessarily imply that the trajectories converge to a single point in Ω. However, this is true if Ω is a set of isolated points. Furthermore, if the system ever reaches a point $P^* \in \Omega$, it will stay at that point for all time, whether or not it is an isolated critical point, since $\dot{P} = 0\ \forall t \geq 0$ at such a point.*

The following useful result pertains to the local stability of critical points of $\mathcal{H}_\alpha$.

Theorem 2 (Isolated Minima are Locally Stable) *Let P^* be a critical point of $\mathcal{H}_\alpha$. Then P^* is a locally asymptotically stable equilibrium of the gradient system $\dot{P} = -kd\mathcal{H}_\alpha/dP$ if and only if P^* is an isolated minimum of $\mathcal{H}_\alpha$.*

Proof: Please see [13] Chapter 9, Section 4, corollary to Theorem 1. □

Remark 2 *Theorem 1 is concerned with all critical points of $\mathcal{H}_\alpha$—maxima, minima, and saddle points. However, it is intuitively clear that the system ought to prefer minima. This intuition is made precise in Theorem 2. There are initial conditions for which the system will converge to a saddle point or a maximum, but these critical points are not locally stable. That is, a perturbation will cause the system to leave the critical point. Minima, on the other hand, are locally stable. They are robust to perturbations.*

Remark 3 (Network Requirements) *The computation of the controller requires that robot i knows the states of all the robots in the network. For this to be feasible there must either be a global supervisor or a fully connected network communication topology. It would be more useful if the controller depended only upon the states of robots with which it communicates. We suggest two methods to accomplish this, but we do not analyze them in detail in this paper. First, robot i can approximate its control law simply by computing (4) using only the states of the robots with which it is in communication. We expect this to give a good approximation because the function $(f(p_j, q)/g_\alpha)^{\alpha-1}$ depends weakly upon the states of agents that are not Voronoi neighbors, especially for small values of α, as evident from Fig. 3 . A rigorous stability analysis of this approximation scheme is difficult, however. A second option is for a robot i to use an estimated configuration vector, $\hat{P}$, in its calculation of the control law. The estimated configuration can be updated online using a standard distributed consensus algorithm (a so called "consensus estimator"). We expect that such a scheme may be amenable to a rigorous stability proof as its architecture is similar to adaptive control architectures. The investigation of these matters is left for future work.*

Remark 4 (Unknown Environments) *The controller requires prior information about the environment such as the importance weighting $\phi(q)$, the sensor function $f(p_i, q)$, and the geometry of the environment Q. One may naturally wonder what can be done if any of this information is lacking. In the case of an unknown weighting $\phi(q)$, a stable controller can be formulated to approximate $\phi(q)$ on-line while carrying out the deployment task, as described in [24]. We are currently looking to extend this method to unknown sensor functions $f(p_i, q)$ and to unknown environment geometries Q. It would be particularly useful to extend to the case of environments with unknown obstacles.*

3 Deriving Special Cases

In this section we show how the cost function (1) can be specialized to give three common kinds of deployment controllers, a Voronoi controller, which is geometric in nature, a minimum variance controller, which has a probabilistic interpretation, and a potential field controller. We conjecture that other deployment objectives beyond these three can be achieved with different choices of the mixing function parameter α.

3.1 Voronoi Coverage, $\alpha \to -\infty$

The Voronoi-based coverage controller described in [8] is based on a gradient descent of the cost function

$$\mathcal{H}_\mathcal{V} = \sum_{i=1}^{n} \int_{V_i} \frac{1}{2} \|q - p_i\|^2 \phi(q)\, dq,$$

where $V_i = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\}$ is the Voronoi cell of robot i and the use of the subscript $\mathcal{V}$ is to distinguish it from $\mathcal{H}$ and $\mathcal{H}_\alpha$. The Voronoi partition can equivalently be written using the min function as

$$\mathcal{H}_\mathcal{V} = \int_Q \min_i (\frac{1}{2}\|q - p_i\|^2)\phi(q)\, dq,$$

because a point q is in the Voronoi cell V_i if and only if $\|q - p_j\|$ is minimized for $j = i$. As noted in Section 2.2, $\lim_{\alpha \to -\infty} g_\alpha(f_1, \ldots, f_n) = \min_i f_i$. Therefore $\mathcal{H}_\mathcal{V}$ is a special instance of (3) with the mixing function $g_{-\infty} = lim_{\alpha \to -\infty} g_\alpha$ and $f(p_i, q) = 1/2\|q - p_i\|^2$.

The choice of the min function for a mixing function now warrants some reflection. Consider a distributed actuation scenario in which we want to position robots so as to service an event that occurs randomly at some point in the environment q. Suppose any robot is equally capable of rendering the service, robots have to physically travel to the event to render the service, and our objective is to service an event as quickly as possible. Naturally, an event should be serviced by the robot that is closest to it, as it will reach the event the most quickly. In this case, the min function is the appropriate choice for a mixing function. By using the min function we are saying that the cost incurred by all the robots due to the event at q is the same as that incurred by the robot that is closest to q.

On the other hand, consider a sensing task in which an event of interest occurs randomly at a point q and is sensed at a distance by sensors located on the robots. In this case the use of the min function is more difficult to justify. Using the min function in this instance would imply that even though both p_i and p_j have some sensory information about the event, the cost function only counts the information from the one that is closest to q. This seems to be a poor choice of cost function for sensing, since in such cases we would want to capture the intuition that two sensors are better than one. The mixing function (2) captures this intuition. Furthermore, even in distributed actuation tasks, using a continuous approximation to the Voronoi cell improves the robustness of the controller. The discrete, geometric nature of the Voronoi computation combined with the continuous controller can lead to chattering, and small sensing errors can result in large changes in the control input. Fortunately, the Voronoi tessellation can be approximated arbitrarily well by choosing a small value of α, thereby preserving the Voronoi controller behavior while improving robustness.

3.2 Minimum Variance Deployment, $\alpha = -1$

We show in this section that setting the mixing function parameter to $\alpha = -1$ causes the robots to minimize the expected variance of their measurement of the location of a target of interest. As a side effect, we will formulate an optimal Bayesian estimator for the location of the target given the measurements of the agents.

Suppose our agents are equipped with sensors that give a noisy measurement of the position of a target in the environment. Let the target position be given by a random variable q that takes on values in Q, and agent i gives a measurement $y_i = q + w$, where $w \sim N(0, I_2\sqrt{f(p_i, q)})$ is a bi-variate normally distributed random variable, and where I_2 is the 2×2 identity matrix. The variance of the measurement, $f(p_i, q)$, is a function of the position of the sensor and the target. Intuitively one would expect a sensor to localize a target with more precision the closer the target is to the sensor. Then the measurement likelihood of agent i is $\mathbb{P}(y_i \mid q : p_i) = 1/(2\pi f(p_i, q)) \exp\{-\|y_i - q\|^2/(2f(p_i, q))\}$, and the notation $\mathbb{P}(\cdot : p_i)$ is to emphasize that the distribution is a function of the

agent position. Assume the measurements of different agents conditioned on the target position are independent. Also, let $\phi(q)$ be the prior distribution of the target's position. Then Bayes rule gives the posterior distribution,

$$\mathbb{P}(q \mid y_1, \ldots, y_n) = \frac{\prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q)}{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q)\, dq}. \tag{5}$$

One can use the posterior to obtain a Bayesian estimate of the position of the event q given the measurements. For example, one may choose to estimate q using the mean, the median, or the maximum of the posterior in (5).

Our interest here, however, is not in estimating q. Instead we are interested in positioning the robots so that whatever estimate of q is obtained is the best possible one. To this end, we seek to position the robots to minimize the variance of their combined sensor measurements. The product of measurement likelihoods in the numerator of (5) can be simplified to a single likelihood function, which takes the form of an un-normalized Gaussian

$$\prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i) = A \exp\left\{-\frac{\|\bar{y} - q\|^2}{2g_{-1}(\cdot)}\right\},$$

whose variance is equivalent to our mixing function $g_{-1}(\cdot) = \left(\sum_{i=1}^n f(p_i, q)^{-1}\right)^{-1}$. The values of A and $\bar{y}$ are not important in this context, though we state them for completeness:

$$\bar{y} = g_{-1}(\cdot) \sum_{i=1}^n f(p_i, q)^{-1} y_i, \quad \text{and}$$

$$A = \frac{1}{(2\pi)^n \prod_{i=1}^n f(p_i, q)} \exp\left\{\frac{1}{2}\|\bar{y}\|^2 g_{-1}(\cdot) - \frac{1}{2}\sum_{i=1}^n \|y_i\|^2 f(p_i, q)\right\}.$$

If we want to position the robots so as to obtain the most decisive information from their sensors, we should move them to minimize this variance. Notice, however, that $g_{-1}(f(p_1, q), \ldots, f(p_n, q))$ is a random variable since it is a function of q. Taking the expectation over q of the likelihood variance gives our original cost function,

$$\mathcal{H}_{-1} = \mathbb{E}_q[g_{-1}(f(p_1, q), \ldots, f(p_n, q))] = \int_Q g_{-1}(f(p_1, q), \ldots, f(p_n, q))\phi(q)\, dq. \tag{6}$$

Thus we can interpret the coverage control optimization as finding the agent positions that minimize the expected variance of the likelihood function for an optimal Bayes estimator of the position of the target.

A more theoretically appealing criterion would be to position the agents to minimize the variance of the *posterior* distribution in (5). This gives the considerably more complicated cost function

$$\mathrm{Var}[q \mid y_1, \ldots, y_n] = \frac{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q) q q^T\, dq}{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q)\, dq} - \bar{q}\bar{q}^T, \tag{7}$$

where

$$\bar{q} = \mathbb{E}[q \mid y_1, \ldots, y_n] = \frac{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q) q\, dq}{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q)\, dq}.$$

The complication of this cost function and the fact that gradients can not be easily computed makes it a less practical option.

3.3 Potential Field Herding, $\phi(q) = \sum_{i=1}^{n} \delta(\|q - p_i\|)$

The third type of deployment controller we consider is significantly different from the previous two in that it does not involve an integral over the environment. Instead it relies on the idea that robots should push away from one another to spread out over an environment, but should not move too far from one another or else they will become disconnected. Surprisingly, however, we will show that this rather different deployment philosophy can be reconciled with our generalized cost function $\mathcal{H}$ in (1).

Let the importance function, $\phi(q)$, be given as a sum of delta-Dirac functions centered at each of the robot positions

$$\phi(q) = \sum_{i=1}^{n} \delta(\|q - p_i\|).$$

Substituting this for $\phi(q)$ in (1), the integral in $\mathcal{H}$ can then be evaluated analytically to give

$$\mathcal{H}_{\text{pot}} = \sum_{i=1}^{n} g(f(p_1, p_i), \ldots, f(p_n, p_i)),$$

and setting $g(f(p_1, p_i), \ldots, f(p_n, p_i)) = \sum_{j=1, j\neq i}^{n} f(p_j, p_i)$ gives a cost function for potential field based herding.

$$\mathcal{H}_{\text{pot}} = \sum_{i=1}^{n} \sum_{j=1, j\neq 1}^{n} f(p_j, p_i), \tag{8}$$

where $f(p_j, p_i)$ can be interpreted as an inter-agent potential function. One choice for $f(p_j, p_i)$ is

$$f(p_j, p_i) = \frac{1}{6}\|p_j - p_i\|^{-2} - \|p_j - p_i\|^{-1} \tag{9}$$

which, taking the gradient of (8), yields the controller

$$\dot{p}_i = k \sum_{j=1, j\neq i}^{n} \left(\|p_j - p_i\|^{-2} - \frac{1}{3}\|p_j - p_i\|^{-3} \right) \frac{p_j - p_i}{\|p_j - p_i\|}. \tag{10}$$

Controllers similar to this one have been studied in a number of works, for example [9,10,14,15,27]. There are numerous variations on this simple theme in the literature.

3.4 Computational Complexity

The gradient controllers described in this work must inevitably be discretized and implemented in a discrete time control loop. We show here that the computational complexity of one loop of the controller (when computed in a distributed fashion over the robot network) for all values of α is $O(d_q nm)$, where n is the number of robots, m is the number of grid squares in the integral computation, and d_q is the dimension of the space. For the herding case, this becomes $O(d_q n^2)$ since the integral simplifies to a sum over the robots.

For the Voronoi controller, we reason as follows. The typical decentralized algorithm for a single robot to compute its Voronoi cell (from [8]) runs in $O(d_q n)$ time. The time complexity for computing a discretized integral is linear in the number of grid squares, and at each grid square

requires a check if the center point is in the Voronoi cell, which is an $O(d_q n)$ operation. Therefore the time complexity of the integral is in $O(d_q nm)$. The Voronoi cell must be computed first, but the discretized integral dominates giving an overall time complexity of $O(d_q nm)$ at each step of the control loop.

For other values of α in (4) (including $\alpha = -1$, which gives the minimum variance controller described above) the controller does not require the computation of a Voronoi cell, but it does require the discretized spatial integral over the environment. We do not have to check if a point is in a polygon, but the integrand we evaluate, namely g_α is linear in n. Therefore the integral computation still has time complexity $O(d_q nm)$, which is the time complexity of the controller at each step of the control loop. The controller we propose in this paper is therefore significantly simpler in implementation (since it does not require the Voronoi computation), though it has the same computational complexity. It will be shown in Section 5 that the use of the continuous function rather than the Voronoi cell provides greater robustness to errors introduced by discretized integration. Finally, for the herding case, the integral is simply a sum over the robots, therefore, $m = n$ and we obtain an order$O(d_q n^2)$ computation at each times step.

4 Convexity and Consensus

Since we treat the multi-agent coverage problem as an optimization, it is natural to ask what sort of optimization we are dealing with, and what optimization tools can be brought to bear to solve it. We show in this section that the cost function in (3) is nonconvex, and that nonconvexity is a required feature of a large class of multi-agent problems, however undesirable this may be from an optimization perspective. Specifically, we demonstrate a link between the *convexity* of a cost function and the multi-agent phenomena known as *consensus*. For our purposes, consensus describes a multi-agent configuration in which all agents take on the same state, $p_1 = p_2 = \ldots = p_n$. Consensus is geometrically represented in the state space $\mathcal{P}^n$ as a d_p-dimensional hyperplane that passes through the origin (from the $d_p(n-1)$ independent equality constraints). This is illustrated by the diagonal line in Fig. 4 in a simplified 2D setting. We will prove, with some technical assumptions, that a multi-agent problem with a *convex* cost function admits at least one globally optimal consensus solution.

We begin with some basic definitions and facts from convex optimization which can be found in any standard text on the topic, for example [2]. A set $\Omega \subset \mathbb{R}^n$ is called convex if, for any two points in Ω, all points along the line segment joining them are also in Ω. Formally,

$$\alpha x + (1-\alpha)y \in \Omega \quad \forall x, y \in \Omega \quad \text{and} \quad \forall \alpha \in [0,1].$$

An important consequence of the convexity of Ω is that any convex combination of points in Ω is also in Ω. A convex combination of m points $x_i \in \Omega$ is one of the form

$$x = \sum_{i=1}^{m} \alpha_i x_i \quad \text{where} \quad \sum_{i=1}^{m} \alpha_i = 1 \quad \text{and} \quad \alpha_i \geq 0 \quad \forall i.$$

A function $f : \Omega \mapsto \mathbb{R}$ is called convex if

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \quad \forall x, y \in \Omega \quad \text{and} \quad \forall \alpha \in [0,1].$$

This is equivalent to saying that the set of all points lying on or above the function $f(x)$ is a convex set (this set is known as the epigraph of $f(x)$). A function is called strictly convex if the '$\leq$' can be

replaced with a '<' in the above relation. Also, we will use the word minimum to mean minimum or infimum if no minimum exists. We now state a theorem that follows from Weierstrass' Theorem and some well-known properties of convex functions.

Theorem 3 (Minima of Convex Functions) *For a convex function $f : \Omega \mapsto \mathbb{R}$, where the domain $\Omega \subset \mathbb{R}^n$ is convex, if any of the following are true:*

1. *Ω is bounded*
2. *There exists a scalar γ such that the level set $\{x \in \Omega \mid f(x) \leq \gamma\}$ is nonempty and bounded*
3. *f is such that $lim_{\|x\| \to \infty} f(x) = \infty$*

then the set of global minima of f is non-empty and convex.

We will apply this result to our multi-agent scenario. Consider a continuous multi-agent cost function $\mathcal{H} : \mathcal{P}^n \mapsto \mathbb{R}$. As before, an agent i has a state $p_i \in \mathcal{P} \subset \mathbb{R}_p^d$. It will be more convenient in this section to refer to a configuration of agents as a tuple $(p_1, \ldots, p_n) \in \mathcal{P}^n$, rather than the column vector notation used previously. Let us assume that agents are anonymous with respect to the cost function, by which we mean that the positions of any two agents can be interchanged without affecting the value of the cost function. This is formalized by the following assumption.

Assumption 1 (Anonymity of Agents) *The cost function $\mathcal{H}$ is such that*

$$\mathcal{H}(\ldots, p_i, \ldots, p_j, \ldots) = \mathcal{H}(\ldots, p_j, \ldots, p_i, \ldots) \quad \forall i, j \in \{1, \ldots, n\}.$$

Assumption 1 is in keeping with the ethos of complex, multi-agent systems, where the emphasis is on the global patterns that result from the interactions of many identical agents. Furthermore, let us assume that $\mathcal{H}$ and $\mathcal{P}^n$ satisfy at least one of the three properties in Theorem 3. Now we give the main result of this section.

Theorem 4 (Convexity and Consensus) *Under Assumption 1, if the cost function $\mathcal{H}(p_1, \ldots, p_n)$ is convex, $\mathcal{P}^n$ is convex, and one of the conditions in Theorem 3 is satisfied, then $\mathcal{H}(p_1, \ldots, p_n)$ has a global minimum such that $p_i = p_j \ \forall i, j \in \{1, \ldots, n\}$.*

Proof: Our argument rests upon Assumption 1 and the fact from Theorem 3 that the set of minima of a convex function $\mathcal{H}$ is a convex set. Let h^* be the set of minima, and let $(\ldots, p_i^*, \ldots, p_j^*, \ldots)$ be an optimal solution in that set. By Assumption 1, $(\ldots, p_j^*, \ldots, p_i^*, \ldots)$ is also an optimal solution for any i and j. Therefore all permutations of components in $(p_1^*, \ldots, p_n^*)$ are optima. Then by convexity of h^*, all convex combinations of points in h^* are in h^*. In particular, the point $(\bar{p}, \ldots, \bar{p})$, where $\bar{p} = 1/n \sum_{i=1}^n p_i$ is an optimal solution (since it is a convex combination of permutations of $(p_1, \ldots, p_n)$). $\square$

We show a geometric schematic of the proof argument in Fig. 4. The proof uses the fact that the convex set of minima must intersect the consensus hyperplane (the hyperplane where $p_i = p_j$ $\forall i, j$) at at least one point. A simple corollary follows.

Corollary 1 (Strict Convexity) *If the conditions of Theorem 4 are met and the cost function $\mathcal{H}(p_1, \ldots, p_n)$ is strictly convex, then the minimum is unique and is such that $p_i = p_j \ \forall i, j \in \{1, \ldots, n\}$.*

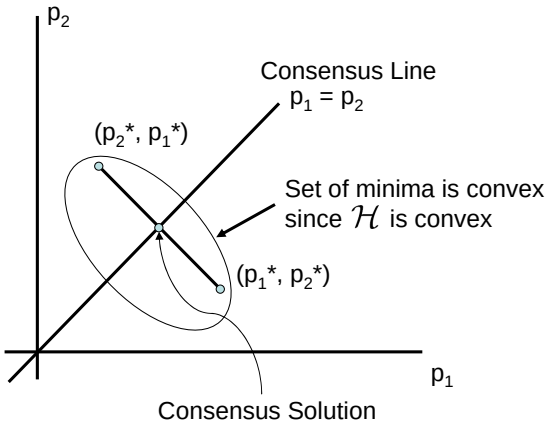


Figure 4: This schematic shows the geometrical intuition behind the proof of Theorem 4 in a simplified 2D setting. Corollary 1 is proved by noticing that the set of minima is a single point (the consensus solution) if $\mathcal{H}$ is strictly convex.

Proof: A strictly convex function has at most one minimum over a convex domain. □

Remark 5 (Consensus vs. Non-consensus) *Theorem 4 suggests that it is futile to search for convex cost functions for multi-robot deployment problems other than consensus. It delineates two classes of multi-agent behaviors reminiscent of complexity classes in the theory of computation. One class, which we will call consensus behaviors, can be described as optimizing a convex cost function. The other class, which we will call non-consensus behaviors, is fundamentally different in that it can only be described with nonconvex cost functions. This is important because if we wish to design an optimization to solve a multi-agent problem, and we know that the problem cannot be solved satisfactorily by all the agents taking the same state, then we must use a nonconvex cost function. Likewise if we observe a multi-agent behavior in nature which cannot be described by all agents reaching the same state (the construction of a termite nest, for example), then an optimization-based explanation of this behavior must be nonconvex.*

Remark 6 (Coverage is Nonconvex) *This is directly applicable to coverage problems. Indeed, coverage cannot be achieved with all agents moving to the same place, therefore coverage problems must involve the optimization of a nonconvex cost function. Our parameterized cost function $\mathcal{H}_\alpha$ from (3) is nonconvex for $\alpha < 1$, in which regime it corresponds to a coverage task (e.g. $\alpha \to -\infty$ for Voronoi and $\alpha = -1$ for minimum variance). It becomes convex (assuming f is convex) for $\alpha > 1$ in which regime it results in consensus. Theorem 4 explains why this is the case.*

4.1 Implications and Future Directions

Theorem 3 may seem disheartening from an algorithmic point of view. Convex optimization has a powerful and well characterized tool set guaranteed to reach global minima, but nonconvex optimization requires searching out special cases and special cost function properties. Often one must be satisfied with local minima. Distributed coverage controllers that use gradient methods

(such as those in this paper) guarantee convergence to local minima, which is all one can expect in a general nonconvex setting.

One may wonder what can be done to overcome the inherently local convergence properties of gradient controllers. Indeed, the picture is not as bleak as it may seem. It may be that the cost function is nonconvex and has multiple local minima which all have an equally low cost (or nearly so). It would seem that the cost function we discuss in this paper (1) is of this form. There are several different locally optimal configurations for the robots, but all are nearly as good as the global optimum. Putting analytical bounds on the difference between the global minimum and any local minimum has proved difficult, but is an area of ongoing research.

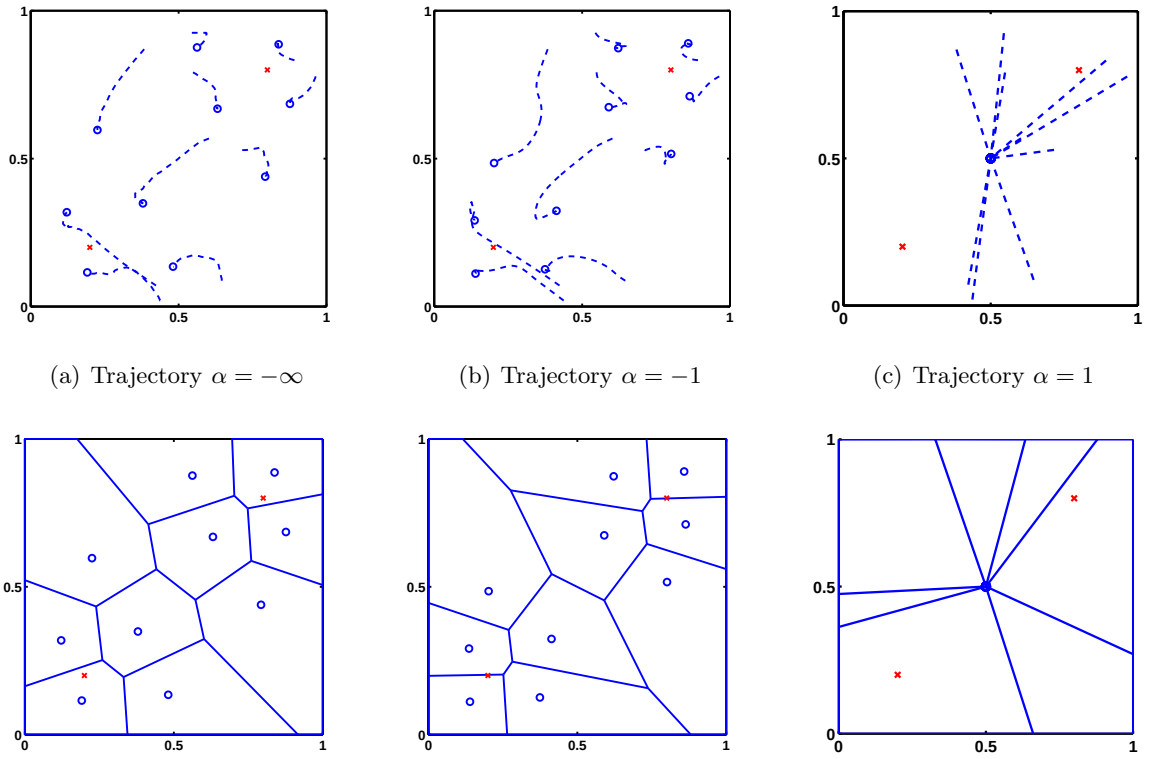
Alternately, one may wonder if any nonconvex optimization techniques can be implemented on a distributed network of robots. There are a umber of such techniques, however the ones that most readily suggest a distributed implementation are those based on gradient methods. For example, simulated annealing and deterministic annealing are optimization techniques in which the state follows the negative gradient of the cost function plus a perturbation term. The perturbation term is meant to "bump" the state out of local minima to explore new areas of the state space. The size of the perturbation decreases with each iteration so that eventually the optimization terminates. This technique can be readily applied to our multi-agent setting. There are some analytical guarantees of convergence to a global minimum, however they generally require strict conditions on the cost function [11]. Another possibility is to use branch and bound techniques, in which a lower and upper bound on the minimum are used to rule out sections of the state space. That is, if the lower bound on the minimum over one section of the state space is higher than the upper bound on the minimum for another, there is no need to search in that section. Once a portion of the state space has been identified, this method is amenable to the multi-agent setting, since the agents can simply drive themselves into a configuration that is in the relevant part of the state space and begin a gradient descent. The difficulty lies in computing which sections of the state space are relevant in a distributed way. It would seem that this would require a global supervisor to assign different parts of the state space to different agents, and then to compile the results of the upper and lower bounds for the different regions to decide in which regions to search. These are promising areas for future research.

5 Simulation Results

The controller for several different values of α and $\phi(q)$ were simulated in a Matlab environment. The environment Q was taken to be a unit square.

For the first set of simulations, the function $\phi(q)$ was set to be the sum of two Gaussian functions, one centered at $(.2, .2)$ and the other at $(.8, .8)$, both with variance .2. We expect to see a higher density of robots around areas of large $\phi(q)$. In our case, the robots group around the Gaussian centers. The results of a simulation with ten robots using the Voronoi based controller, which corresponds to $\alpha \to -\infty$, is shown in Figs. 5(a) and 5(d). Similar plots are shown for the minimum variance controller, with $\alpha = -1$, in Figs. 5(b) and 5(e), and the controller with $\alpha = 1$ in Figs. 5(c) and 5(f). Comparison of the controllers shows that the Voronoi based controller causes the robots to spread out more, while as α increases, the robots group more closely together. When $\alpha \geq 1$, the cost function becomes convex, and the robots all move to the same position as seen in Fig. 5(f), which corroborates our results relating convexity to consensus.

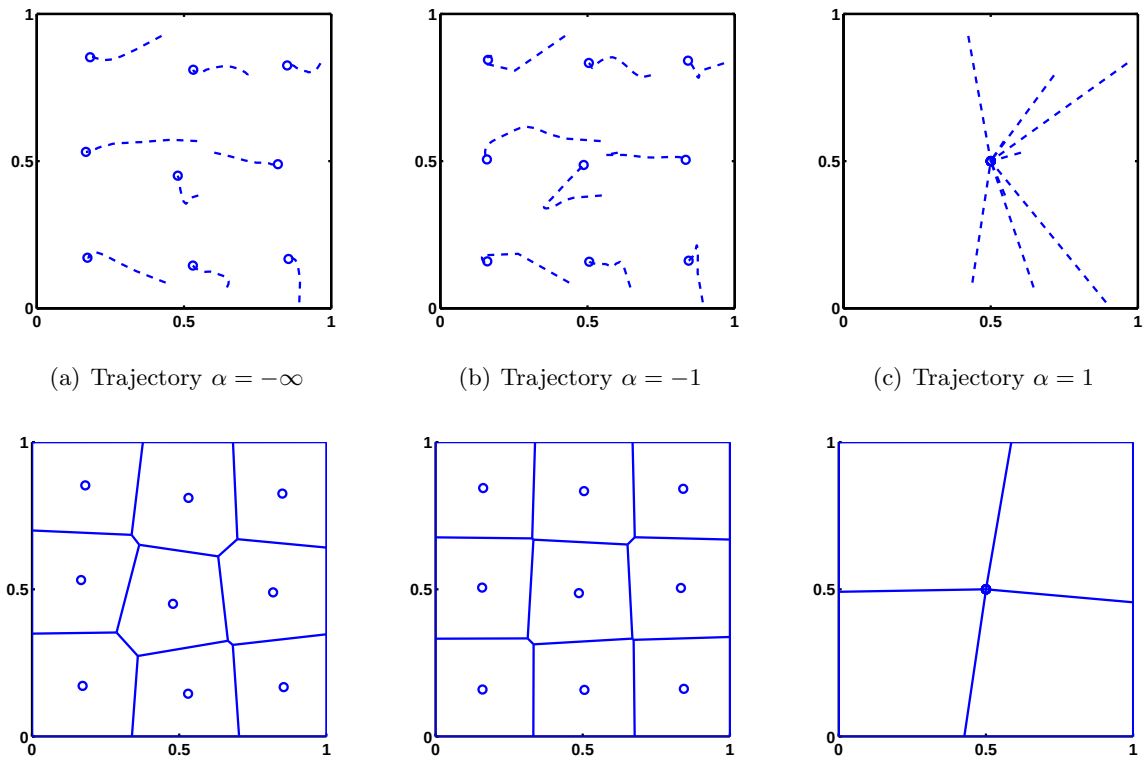
The second simulation scenario is similar to the first, but the weighting function, $\phi(q)$, is set



(a) Trajectory $\alpha = -\infty$ (b) Trajectory $\alpha = -1$ (c) Trajectory $\alpha = 1$

(d) Final Config. $\alpha = -\infty$ (e) Final Config. $\alpha = -1$ (f) Final Config. $\alpha = 1$

Figure 5: Trajectories and final configurations are shown for ten robots using the gradient control law with three different parameter values: $\alpha = -\infty$ for the Vornoi controller (5(a), 5(d)), $\alpha = -1$ for the minimum variance controller (5(b), 5(e)), and $\alpha = 1$, which leads to consensus (5(c), 5(f)). The weighting function was a sum of two Gaussians, whose centers are marked with red $\times$s. The Voronoi tessellation is shown for all scenarios for comparison, even though the right two controllers do not use the Voronoi cells for control.



(a) Trajectory $\alpha = -\infty$ (b) Trajectory $\alpha = -1$ (c) Trajectory $\alpha = 1$

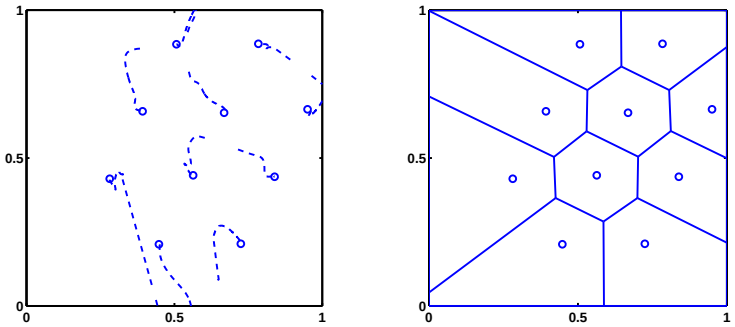
(d) Final Config. $\alpha = -\infty$ (e) Final Config. $\alpha = -1$ (f) Final Config. $\alpha = 1$

Figure 6: Trajectories and final configurations are shown for nine robots using the gradient control law with three different parameter values: $\alpha = -\infty$ for the Vornoi controller (6(a), 6(d)), $\alpha = -1$ for the minimum variance controller (6(b), 6(e)), and $\alpha = 1$, which leads to consensus (5(c), 6(f)). The weighting function $\phi(q)$ in this case was uniform, so the robots move to an even 3×3 grid for all $\alpha < 1$. The Voronoi tessellation is shown for all scenarios for comparison, even though the right two controllers do not use the Voronoi cells for control.

to be uniform, and we simulate a group of 9 robots. In this case, no area in the environment is preferred over another, so the robots move to an even 3×3 grid for all values of $\alpha < 1$. This is illustrated in Figs. 6(a) and 6(d) for $\alpha = -\infty$ and Figs. 6(b), and 6(e) for $\alpha = -1$.

Notice that even though both controllers converge to nearly a 3×3 grid, the one with $\alpha = -1$ more closely approaches the exact grid. The deviations from the exact 3×3 grid are caused by errors from the numerical integration required by the controller. Comparing these two simulations highlights one of the advantages of using a finite value of α, namely the controller is computed with a continuous function rather that a geometric Voronoi cell, and is therefore more robust to inaccuracies caused by numerical integration. As before, when $\alpha \geq 1$ the cost function becomes convex, and the robots all move to the same position, as shown in Figs. 6(c) and 6(f).

The third scenario shown in Figs. 7(a) and 7(b) uses the potential field controller from (10). This controller uses a sum of delta-Dirac functions for $\phi(q)$, which causes the robots to arrange themselves in the close-packed lattice pattern. Even though $\alpha = 1$ in this scenario, the cost function is nonconvex because the inter-agent potential, $f(p_j, p_i)$, is nonconvex as given by (9).



(a) Trajectory Pot. Field (b) Final Config. Pot. Field

Figure 7: Trajectories and final configurations (7(a), 7(b)) are shown for ten robots using the potential field controller. The Voronoi tessellation is shown for comparison, even though the controller does not use the Voronoi cells for control.

6 Conclusion

In this paper we introduce a unifying optimization framework for multi-robot deployment that brings together several different existing deployment algorithms. We point out that important properties of the underlying objective are embodied in the way sensor information or actuator capabilities are combined from different robots. We propose a parameterized function to accomplish this combination, where different parameter values are shown to lead to different kinds coverage algorithms. Finally, we prove that for deployment problems other than consensus, the underlying optimization is necessarily nonconvex, making global optimization an unrealistic objective, especially for gradient descent controllers.

Our work invites an immediate extension, which is how to approximate the gradient controller over a communication graph. We outlined two methods for doing this. In the future the stability and robustness properties of these methods should be characterized and other methods should be investigated as well. Also our recognition that deployment problems stem from nonconvex optimizations suggests some new research directions. Gradient descent controllers, which are the most common type in the multi-robot deployment literature, can only be expected to find local minima in general. Therefore it is worthwhile to look for cost functions with special properties that allow for global optimization despite being nonconvex. Also it would be interesting to investigate other nonconvex optimization methods that can be implemented in a multi-agent setting. We expect that these open questions will point the way toward new results in multi-robot control.

Acknowledgements

This work was done in the Distributed Robotics Laboratory at MIT. This work was supported in part by the MURI SMARTS project grant number N00014-09-1-1051, the MURI SWARMS project grant number W911NF-05-1-0219, NSF grant numbers IIS-0513755, IIS-0426838, CNS-0520305, CNS-0707601, EFRI-0735953, the MAST project, and The Boeing Company.

References

[1] A. Arsie and E. Frazzoli. Efficient routing of multiple vehicles with no explicit communications. *International Journal of Robust and Nonlinear Control*, 18(2):154–164, January 2007.

[2] D. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Nashua, NH, 2003.

[3] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart, and D. Rus. Voronoi coverage of non-convex environments with a group of networked robots. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, May 2010.

[4] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. June 2008. Manuscript preprint. Electronically available at http://coordinationbook.info.

[5] Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Complete distributed coverage of rectilinear environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, March 2000.

[6] H. Choset. Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[7] J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESIAM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.

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

[9] D. V. Dimarogonas and K. J. Kyriakopoulos. Connectedness preserving distributed swarm aggregation for multiple kinematic robots. *IEEE Transactions on Robotics*, 24(5):1213–1223, October 2008.

[10] V. Gazi and K. M. Passino. A class of repulsion/attraction forces for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[11] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652656, June 1994.

[12] M. L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom. Multisensor resource deployment using posterior Cramér-Rao bounds. *IEEE Transactions on Aerospace and Electronic Systems*, 40(2):399–416, 2004.

[13] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, Inc., Orlando, FL, 1974.

[14] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS02)*, Fukuoka, Japan, June 2002.

[15] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[16] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of 22nd Conference on Artificial Intelligence (AAAI)*, Vancouver, Canada, July 2007.

[17] J. LaSalle. Some extensions of liapunov's second method. *IRE Transactions on Circuit Theory*, 7(4):520–527, 1960.

[18] D. T. Latimer IV, S. Srinivasa, V.L. Shue, S. Sonne adnd H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 961–967, May 2002.

[19] W. Li and C. G. Cassandras. Distributed cooperative coverage control of sensor networks. In *Proceedings of the IEEE Conference on Decision ans Control, and the European Control Conference*, Seville, Spain, December 2005.

[20] S. Martínez, J. Cortés, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007.

[21] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira. Sensing and coverage for a network of heterogeneous robots. In *Proceedings of the IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[22] M. Schwager, B. Julian, and D. Rus. Optimal coverage for multiple hovering robots with downward-facing cameras. In *Proceedings of the International Conference on Robotics and Automation (ICRA 09)*, pages 3515–3522, Kobe, Japan, May 12–17 2009.

[23] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus. From theory to practice: Distributed coverage control experiments with groups of robots. In O. Khatib, V. Kumar, and G. Pappas, editors, *Experimental Robotics: The Eleventh International Symposium*, volume 54 of *Springer Tracts in Advanced Robotics (STAR)*, pages 127–136, Berlin, 2008. Springer-Verlag.

[24] M. Schwager, D. Rus, and J. J. Slotine. Decentralized, adaptive coverage control for networked robots. *International Journal of Robotics Research*, 28(3):357–375, March 2009.

[25] J. J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Upper Saddle River, NJ, 1991.

[26] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal of Control and Optimization*, 48(5):3224–3245, 2010.

[27] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, May 2007.