Distributed Gradient Optimization with Embodied Approximation

Yaroslav Litus, Richard T. Vaughan

Autonomy Lab, Simon Fraser University, Canada
ylitus@cs.sfu.ca

Abstract

We present an informal description of a general approach for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and/or time, into some optimal and initially unknown configuration. Our approach relies on using embodiment and spatial embeddedness as a surrogate for computational resources, permitting the reduction or elimination of communication or shared memory for conventional parallel computation. Intermediate stages of the gradient descent process are manifested by the locations of the robots, instead of being represented symbolically. Each point in the space-time evolution of the system can be considered an approximation of the solution, which is refined by the agents' motion in response to sensor measurements. For each agent, motion is approximately in the direction of the local antigradient of the global cost function. We illustrate this approach by giving solutions to two non-trivial realistic optimization tasks from the robotics domain. We suggest that embodied approximations can be used by living distributed systems to find affordable solutions to the optimization tasks they face.

Introduction

We know by now that The World Is Its Own Best Model (Brooks (1990)). Brooks' recommendation to avoid internalizing the world - a costly and error-prone process - but rather to sense and react to it directly whenever possible, has become part of the robotics canon. This paper makes explicit an inversion of this approach, in which where the world is used to externalize a computational process.

In particular, we present an informal description of a general approach of developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and time into some optimal and initially unknown configuration. This class of problems includes many classical mobile agent problems such as formation control, facility location, rendezvous, navigation and interference reduction. Given the intractability of finding optimal solutions to these large, high-dimensional joint state-space planning problems, gradient descent methods are commonly used to find approximate solutions.

The proposed approach is to use embodiment and spatial embeddedness as a surrogate for computational resources, permitting the reduction or elimination of communication or shared memory for conventional parallel computation. Intermediate stages of the gradient descent process are manifested by the locations of the robots, instead of being represented symbolically. Each point in the space-time evolution of the system can be considered an approximation of the solution, which is refined by the agents' motion in response to sensor measurements. For each agent, motion is approximately in the direction of the local antigradient of the global cost function.

Gradient-based formation control and navigation have been widely used in robotics (Zelek, 1999; Tanner and Kumar, 2005). The idea of embodied computation was recently presented in Hamann and Wörn (2007), however contrary to the authors' claim we believe that some globally defined algorithms can be implemented by embodied computation. Finally, Loizou and Kumar (2007) have shown biologically plausible navigation and tracking algorithms which involve using other agents locations to calculate a local gradient. To our knowledge, this paper is the first to explicitly present embodied approximation as a method to implement parallel gradient optimization algorithms.

We illustrate this approach by giving solutions to two non-trivial realistic optimization tasks from the robotics domain. This work is in the context of our interest in large-scale distributed systems such as animal colonies and multi-robot systems, which work together to solve complex tasks. We are interested in identifying mechanisms that exploit the characteristics of the embodied multi-agent domain to solve complex computational problems. We are particularly interested in solving practical resource allocation problems in multi-robot systems, with *energy autonomy* as the key motivating problem. This is the motivation for our choice of example problems: two different versions of energy-efficient robot-robot rendezvous, useful for for recharging or refueling, or as a component of various other tasks. While looking for ways to find meeting places which minimize the traveling costs for groups of robots, we developed fully decentralized heuristic methods which require no range information to converge to a good approximation of the optimal solution. These heuristics afford a very simple implementation.

The key insight that underlies our approach is that the spatial configurations of the agents themselves could be considered as an approximate solution to the entire problem. An individual agent can move itself, thus refining its local component of the current solution approximation. No representation of the problem, or the current solution, needs to be held by any robot: they manifest the solution by their physical configuration. This is an example of what Payton has called "world-embedded computation" (Payton et al., 2001) and exploits the property of "strong embodiment" identified by Brooks (1990), extended to a multi-agent system.

Distributed gradient optimization with embodied approximation

Assume a system of n spatially embedded agents is described by a spatial configuration $s \in H$ where $H = H_1 \times H_2 \times \ldots \times H_n$ and H_i is a vector space of possible configurations of agent i. We will denote the i-th component of s as s_i. Every agent has control over the first-order dynamics of its own configuration which allows it to continuously change its configuration possibly with some restrictions on the speed of this change. We also assume that every agent i can sense the spatial configuration of a set of other agents ψ_i. This set can change as the system evolves. Given some cost function $J : H \rightarrow R$ we want to rearrange the system into an optimal configuration $s^* = \arg\min_s J(s)$. Note that the cost function may be parametrized by initial agent configuration s_0.

If the function J has certain mathematical properties (e.g. is continuously differentiable) then a good approximation to an optimal configuration can be found in a centralized way by using one of many gradient optimization algorithms. It will start with some initial approximation and iteratively change it in constant or decreasing steps in the direction of (approximated) antigradient. Details of these particular algorithms are irrelevant for present discussion. This centralized solution will need a central processor with amount of memory of the order of the size of the state vector s as well as means to communicate s^* to all agents once an acceptable approximation is found. Once all agents know their component of s^* they can proceed directly towards it. If J is parametrized by initial configuration of agents S_0 then the means of communicating or sensing this configuration by the central processor are required.

Since the system of n agents in question has n processors that can work in parallel it is natural to try to use this resource to get a parallel solution. Distributed implementation of iterative algorithms are well-studied (Baudet, 1978; Bertsekas, 1982; Kung, 1976). It is known that if processors synchronously compute and communicate their partial results to other processors then the resulting distributed procedure is equivalent to a single-processor implementation thus preserving its convergence properties. Further, under certain realistic conditions (like small iteration steps and bounded communication delays) the synchrony assumption can be relaxed and processors can be allowed to compute at different speeds and communicate sporadically while still giving the same convergence properties as original serial algorithm (Tsitsiklis et al., 1986). Therefore, we can assign every agent i the task of iterative optimization of its own component s_i of the global state vector s by changing s_i in the direction of the corresponding component of the antigradient vector ∇J_i and periodically communicate the current value of s_i to other agents as well as receive updates of the current approximations of $s_j, j \neq i$ from other agents. Note, that the amount of information from other processors needed to compute ∇J_i depends on the particular problem. If ∇J_i depends only on s_i then no additional information is necessary and s_i can be computed independently of other processors. If ∇J_i depends on some other components of the global state s then the current values of these components should be received from the processors which compute them. We denote the set of agents that compute components necessary to calculate ∇J_i by ξ_i. If the assumptions of Tsitsiklis et al. (1986) are met, then eventually every agent will know the approximation s_i^* and will be able to proceed towards it.

However, a physical multi-agent system is much more than simply n parallel processors. Physical embodiment implies spatial embeddedness, and for some problems these remarkable dual properties allow us to drastically reduce or totally eliminate the need to communicate intermediate results of calculations, or indeed any *description* of the problems or solutions, substituting instead direct physical observations. In addition, instead of waiting for an iterated algorithm to converge agents can perform reconfiguration *during* the optimization thus giving the resulting algorithm an attractive anytime property. The high-level description of the approach is given in Algorithm 1.

Algorithm 1 Gradient optimization with embodied approximation

```
1:  for all agents $i$ do
2:      sense current configurations $s_j^{\psi}$ for $j \in \psi_i \cap \xi_i$
3:      update using communication current configurations
        $s_j^{\psi}$ for $j \in (\{1, 2, \ldots, i-1, i+1, \ldots, n\} \backslash \psi_i) \cap \xi_i$
4:      $D_i \leftarrow \nabla J_i(s_i, s_{j|j \in \xi_i})$
5:      if $\nabla J_i$ exists and component can be improved then
6:          move in direction $-D_i$
7:      else
8:          stay still
9:      end if
10:     communicate new configuration to other agents
11: end for
```

The key idea is to use agent configurations directly as approximations to corresponding components of the global goal configuration. Thus all agents move in the (approximated) antigradient direction which is calculated at Step 4 using their current configuration as the current approximation to the solution. The communication at Step 3 is necessary only if not all of the information from the relevant agents belonging to set ξ_i can be acquired by direct sensing at Step 2. In the best case no communication is required as all necessary information is available via sensing, so $\psi_i \supseteq \xi_i$. If some components nevertheless have to be acquired by communication, this could be done in an asynchronous sporadic manner as was argued above.

This approach makes agents themselves serve as a shared "memory" for the parallel computation they perform where the information about current approximation is embodied in physical configuration of the agents. Once the values of necessary components are available, gradient direction ∇J_i or an approximation to it can be calculated and agent can move in antigradient direction thus improving approximation of component s_i and global state s. In a certain sense, agent relocation becomes a part of a computation process, creating a parallel computer comprised by agent's processors and physical bodies of agents. The algorithm continues until convergence is reached which is detected in a way specific for the particular problem and algorithm employed.

Applications

We illustrate the idea of optimization with embodied approximation by two problems of energy-efficient multi-robot coordination which we have studied in previous work. The first problem is to assemble a team of robots at an initially unknown point which minimizes the total energy spent on relocation (Zebrowski et al., 2007). The second, more difficult problem is an instance of the multi-facility location problem which asks to plan a route for a team of robots to rendezvous a single dedicated service robot, possibly each at a *different* point (Litus et al., 2008, 2007).

Energy-efficient single-point rendezvous

Assume n robots are located at positions r_i, $i = 1 \dots n$. When a robot moves, it expends energy proportional to the length of its trajectory. Robots have individual energy costs c_i per unit of traveled distance, thus if robot i moves from a to b, it spends $c_i||a - b||$ units of energy. Now the task is to find a point p^* which minimizes the total energy spent by all robots for meeting at that point.

Definition 1 (Energy-efficient rendezvous problem)**.** *Given robot locations $r_i \in R^d, i = 1, \dots, n$ find rendezvous point*

$$p^* = arg\min_p J(p) = \sum_{i=1}^{n} c_i||p - r_i|| \qquad (1)$$

Historically, this problem has a variety of names including the Fermat-Steiner problem, Weber problem, single facility location problem, and the generalized Fermat-Torricelli problem. Though it is not possible to find a closed-form solution for this problem, the properties of its solutions are well known (see Gueron and Tessler (2002) for the case $n = 3$ and Kupitz and Martini (1997) for the general case). Effective numerical algorithms exist (Rosen and Xue, 1991). Interestingly, it is also possible to describe the solution using a mechanical interpretation as a system of idealized strings, pulleys and weights (Polya, 1968).

The antigradient of the cost function is

$$-\nabla J(x) = \sum_{i=1}^{n} u(x, r_i), u(a, b) = \frac{b - a}{||a - b||} \qquad (2)$$

That is, the antigradient at some point is simply a sum of unit vectors pointing in the directions of original robot locations. Note that antigradient is not defined at the locations of the robots themselves while one of these locations can be a solution. If points r_i are not collinear (not lying upon a straight line), the goal function in (1) is strictly convex, which ensures the uniqueness of p^* if $n \geq 3$. In this case solution is characterized by the following theorem (Kupitz and Martini, 1997).

Theorem 1. *If r_i are not collinear and for each point r_i*

$$||\sum_{j=1}^{n} c_j \frac{r_j - r_i}{||r_j - r_i||}|| > r_i, i \neq j \qquad (3)$$

then $p^* \neq r_i$ for any i and $\nabla J(p^*) = 0$ (the floating case). If (3) does not hold for some r_i then $p^* = r_i$ (the absorbed case).

A simple embodied approximation method gives a useful solution to this problem. Let all robots compute the solution point approximation simultaneously by checking the absorbed case condition in Theorem 1 and moving along antigradient direction if the condition is not met. This produces the distributed Algorithm 2 which runs until robots converge to a single point. Note that since robots are embodied they can not occupy the same point, thus we consider that two robots met whenever the distance between them is closer than some meeting range ϵ.

In order to calculate the antigradient at its current location, each robot needs to know the direction towards the original robot locations r_i. This could be achieved either by means of global localization and memorizing r_i or by setting static beacons at points r_i (hence the name of the algorithm). Both localization and static beacons are expensive solutions, but embodied approximation can be used once again to get rid of the necessity to calculate directions to r_i.

Once some of the robots move from their original locations r_i a new instance of the rendezvous problem is created. In this new instance all robots are located at the new

Algorithm 2 Static algorithm for energy-efficient rendezvous

```
1: for all robots $i$ do
2:     Update current location of self, $x_i$
3:     $D_i \leftarrow \sum_{i|x \neq r_i} c_i u(x, r_i), u(a,b) = \frac{a-b}{||a-b||}$
4:     if $x = r_i$ for some $i$ then
5:         $c \leftarrow c_i$
6:     else
7:         $c \leftarrow 0$
8:     end if
9:     if $||D_i|| < c$ then
10:        stop
11:    else
12:        move in direction $D_i$
13:    end if
14: end for
```

"original" locations r_i thus sensing the instantaneous directions to the robots is enough to approximate the antigradient value. Hence memorizing r_i or using static beacons is replaced by using the robots as dynamic beacons resulting in the distributed Algorithm 3.

Algorithm 3 Dynamic algorithm for energy-efficient rendezvous

```
1: for all robots $j$ do
2:     $A_j \leftarrow \{i | ||r_i - r_j|| \leq \epsilon\}$, meaning the set of robots
       which are closer to $r_j$ than meeting threshold $\epsilon$. Note
       that $j \in A$, thus $A$ has at least one element.
3:     $D_j \leftarrow \sum_{i \notin A} c_i \vec{d}(r_j, r_i)$.
4:     $c \leftarrow \sum_{i \in A} c_i$
5:     if $||D_j|| < c$ then
6:         stop
7:     else
8:         move in direction $D_i$
9:     end if
10: end for
```

A set of experiments was conducted to compare the performance of these algorithms with the performance of a centralized optimization algorithm (see Zebrowski et al. (2007) for details). Some typical results are shown in Table 1. The centralized static algorithm calculates the optimal meeting point p^* exactly once and commands all robots to proceed directly to that point. The centralized dynamic algorithm periodically recomputes the meeting point incorporating new positions of the robots while they move towards their rendezvous. It could be seen that distributed methods achieve the quality of solution comparable to centralized methods. The difference is explained by the fact that the trajectory along the antigradient is not straight and thus longer then the direct path to the optimal meeting locations (see Fig. 1). However, in applications that require scalability distributed algorithms with low demands on sensing and communication will be preferable even if they produce slightly worse results than centralized solutions. As the number of robots grows so does the running time of a centralized algorithm. Often this growth is fast enough to render the centralized solution impractical for a fairly small number of robots. At the same time scalable distributed solutions work with any number of robots, sometimes at a price of giving worse results.

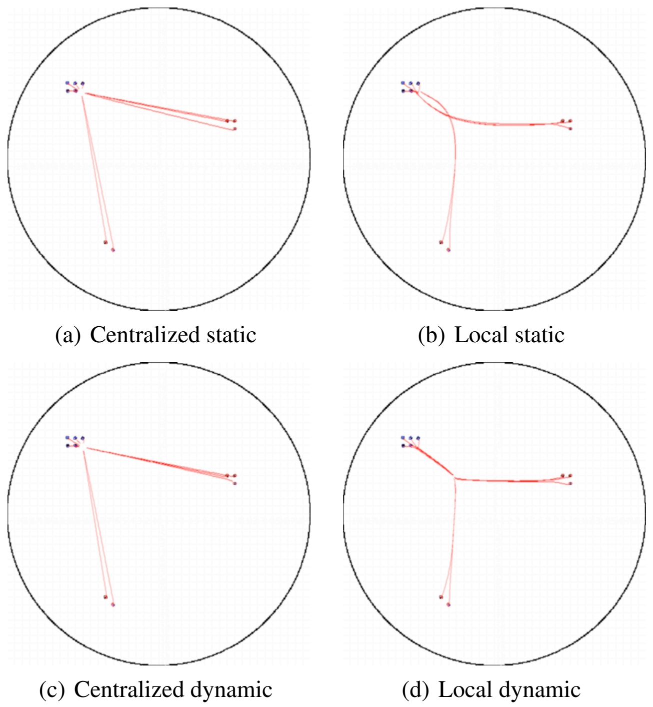

(a) Centralized static (b) Local static (c) Centralized dynamic (d) Local dynamic

Figure 1: Typical paths taken to single point rendezvous (Map 2)

Table 1: Mean Total Energy Used For Rendezvous. All Standard Deviations $< 5\%$

	Dynamic		Static	
Map	Centralized	Distributed	Centralized	Distributed
1	149.99	151.56	149.92	152.79
2	105.21	115.21	105.25	119.00
3	77.56	80.49	77.28	81.48
4	477.89	503.18	476.97	530.11

Ordered Frugal Feeding Problem

Consider a team of worker robots that can recharge by docking with a dedicated refueling (or equivalently, recharging) robot called a *tanker*, as described in (Zebrowski and Vaughan, 2005). The tanker robot could remain at a fixed location, acting as a conventional charging station, or it could move to rendezvous with worker robots. Simultaneously, worker robots can wait for the tanker to come to them, or they can move to meet the tanker.

By analogy to a mother animal attending her offspring we call this problem the "Frugal Feeding Problem". If we impose a total order in which worker robots must be met and charged, (perhaps based on urgency or some other priority scheme) we obtain the "Ordered Frugal Feeding Problem" considered here. As in a single point rendezvous problem we model locomotion costs as the weighted Euclidean distance between the origin and destination.

The Ordered Frugal Feeding Problem can be stated formally as follows:

Definition 2 (Ordered Frugal Feeding Problem)**.** *Given tanker location $p_0 \in \mathbb{R}^d$, worker locations $r_i \in \mathbb{R}^d, i = 1, \ldots, k$ find*

$$\min_{p_1,p_2,\ldots,p_k} J(p_1, p_2, \ldots, p_k) = \sum_{i=1}^{k} \big(w_0||p_i - p_{i-1}|| + w_i||r_i - p_i||\big); \quad (4)$$

Here $w_0||p_i - p_{i-1}||$ gives the cost of tanker relocation between points p_{i-1} and p_i, $w_i||r_i - p_i||$ gives the cost of moving worker i from its original position to meeting point p_i.

Definition (2) could be amended to require the tanker to return to its original location after attending all workers, perhaps to refuel itself, without affecting the presented results.

We denote solution points as p_i^*. The possibility of several robots being attended in one place is permitted, and captured by the possible coincidence of some meeting points. We define a *meeting* as the event when robots come within distance s of each other.

Unlike the single-point rendezvous case this problem involves several relocations of the system (tanker needs to go between all rendezvous points in turn). However, the embodied approximation approach is applicable and we have found a distributed algorithm which produces good approximations to the optimal solutions.

We start with considering the antigradient of the cost function J. Using Eq (2) we can express the i-th component of the antigradient as

$$-\nabla J_i(p_i) = w_0 u(p_i, p_{i-1}) + w_0 u(p_i, p_{i+1}) + w_i u(p_i, r_i), \quad (5)$$

for $i = 1..k-1$. The antigradient component for the last robot k includes one less term. Let the current position of every worker robot i represent its current approximation to the solution point p_i^*. Let the position of the tanker also represent its current approximation of the meeting point for the first robot to be charged. That is, point p_1 is simultaneously represented by tanker and worker 1. The system will perform parallel gradient descent with all robots moving along the corresponding approximation to the antigradient component given by Eq (5) calculated at their own position. In other words, robots 0 (the tanker)

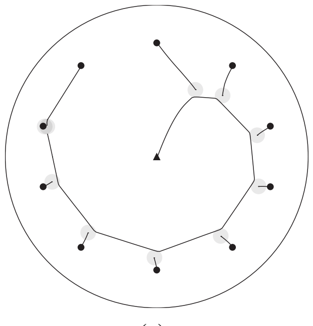
Algorithm 4 Distributed Algorithm for Ordered Frugal Feeding Problem

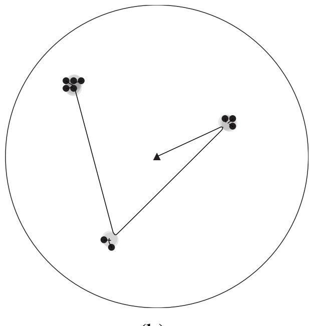
```
 1: $i \leftarrow 1$
 2: define $\vec{d}(x,y) = (y-x)/||x-y||$
 3: let $r_0$ be the current tanker position, $r_i$ be the position
    of worker $i$
 4: **while** $i \le n$ **do**
 5:   **if** $r_0$ is close to $r_i$ **then**
 6:     tanker charges worker $i$; $i \leftarrow i+1$
 7:   **else**
 8:     **if** $i = n$ (only one robot in queue) **then**
 9:       the lighter of tanker and worker goes towards the
          other
10:     **else**
11:       $r_{n+1} = r_n$
12:       $\vec{D}_0 \leftarrow w_1\vec{d}(r_0,r_1) + w_0\vec{d}(r_0,r_2)$.
13:       $\vec{D}_i \leftarrow w_0\vec{d}(r_i,r_0) + w_0\vec{d}(r_i,r_{i+1})$
14:       **for all** $i < j \le n$ **do**
15:         $\vec{D}_j \leftarrow w_0\vec{d}(r_j,r_{j-1}) + w_0\vec{d}(r_j,r_{j+1})$
16:       **end for**
17:       **for all** $j \in \{0, i, i+1, \ldots, n\}$ **do**
18:         **if** $||\vec{D}_j|| < w_j$ **then**
19:           robot $j$ stops
20:         **else**
21:           robot $j$ proceeds in the direction $\vec{D}_j$
22:         **end if**
23:       **end for**
24:     **end if**
25:   **end if**
26: **end while**
```

and 1 (the next robot to be charged) move along the approximated antigradient of the part of cost function $g(p_1) = w_0||p_0 - p_1|| + w_0||p_1 - p_2|| + w_1||p_1 - r_1||$ using the current position of robot 2 as an approximation to the unknown solution point p_2. The rest of the robots $j, j = 2, ..., k$ move along the approximated antigradient of the cost functions $f_j(p_j) = w_0||p_j - p_{j-1}|| + w_0||p_j - p_{j+1}|| + w_j||r_j - p_j||$ using r_{j-1}, r_{j+1} as approximations for the unknown solution points p_{j-1}, p_{j+1}. Algorithm 4 describes the procedure formally. Convergence of this algorithm is guaranteed, as analyzed in Litus et al. (2008).

Parallel computation of movement directions in steps 11-14 and simultaneous movement of all robots in steps 17-20 provide the scalability of this algorithm. Importantly, the per-robot, per-timestep cost is constant, so the method works for arbitrary population sizes. Unlike the single-point algorithms described in previous section, here every robot needs to know the direction to at most two other robots to calculate its movement direction.

More specifically, this algorithm requires every worker to know its own weight and the tanker weight, whether or not it is the head of the current charging queue, and the direc-


(a)


(b)

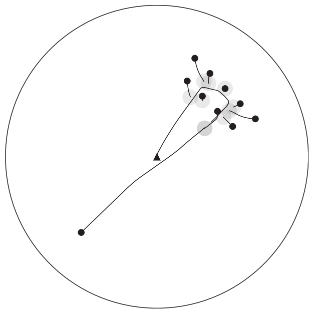

(c)

Figure 2: Some trajectories produced by distributed algorithm for Ordered Frugal Feeding Problem. Triangle represents tanker, small disks represent workers, large disks show the meeting ranges at the final point of each worker trajectory.

tion towards the next worker in the queue and the previous worker (or tanker if the robot is at the head of the queue). The tanker needs to know its own weight, the weights of the first two robots in the queue and the directions towards them. As a robot is met and charged, it is removed from the queue, and this update is broadcast to all robots.

As in the single point rendezvous case here instead of operating with the model of the world and searching for the complete solution, each robot uses the position of itself, its queue predecessor and successor as the current embodied approximation to the solution points. Every robot tries to improve the global solution quality by moving in the direction which decreases the part of the total cost function that concerns itself and its neighbors. If the robot finds itself located at the minimizing point for the current local configuration of robots, the robot stops. Fig 2 shows some trajectories produced by the algorithm.

To evaluate the performance of the algorithm we performed 3000 simulations running distributed algorithm for Ordered Frugal Feeding Problem on randomly generated instances of the problem, with initial robot locations drawn from the same uniform distribution, and 3 different uniform distributions of weights, with 1000 experiments for each weight distribution. In each experiment, 10 randomly weighted worker robots and a tanker were placed at random locations in an square arena with 20m sides. A meeting range of a 0.1m and a movement step length of 0.01m were set. The path traveled by each robot was recorded, and it's length was multiplied by the robot's weight, then summed for the population to give the total energy spent on performing the rendezvous. For comparison we computed optimal solutions of each instance using discrete search (Litus et al., 2007) on a regular grid with 40 ticks per dimension covering the embedding hypercube of original robot locations. The regularity of the grid allows us compute a lower bound of the optimal cost based on the result of discrete search by subtracting the maximum possible error due to discretization (no such quality bounds are readily available for numerical approximation methods to the continuous problem, and no closed form solution is known).

We calculated the upper bound of approximation factor for every problem instance by dividing distributed algorithm for Ordered Frugal Feeding Problem results by the lower bound of optimal cost. Table 2 reports the statistics of these approximation factor bounds for each distribution.

Table 2: Statistics of approximation factor bounds

	$w_i = 1$	$w_i \sim U[1,3)$	$w_i \sim U[1,100]$
Mean	1.19	1.22	1.31
Median	1.19	1.20	1.25
St. dev.	0.03	0.08	0.23
Skewness	0.67	1.18	4.27
Kurtosis	0.65	1.65	31.82

The results show increasing variation of the approximation factor bounds with increasing variation in robot weights. Indeed, this distributed algorithm for Ordered Frugal Feeding Problem can cause the tanker to move suboptimally in the direction of two light workers and then return to the third heavy worker. Hence, this distributed algorithm for Ordered Frugal Feeding Problem does not have a constant approximation factor. However, the average quality of solutions for uniformly distributed problem appears very good. Thus, the method could be used where the guaranteed quality of results obtained in non-scalable centralized manner should be sacrificed in favor of simple decentralized scalable solution with good average performance.

Conclusion

A general approach for development of decentralized distributed gradient descent optimization algorithms for teams of embodied agents is presented. This approach uses embodiment and spatial embedding as valuable computational resources which allow to reduce or eliminate communication or shared memory requirements for parallel computation. Spatial configuration of agents serves as an embodied representation of the current approximation to the global solution which is accessed by means of sensing and refined by agents moving along local antigradient directions. Two non-trivial optimization tasks of energy-efficient path planning for mutli-robot teams were used to illustrate the embodied approximation approach. The resulting distributed algorithms show good results in experimental evaluation. These algorithms use very simple mechanisms that result in energy efficient group behavior. In both cases, computationally complex optimization tasks are solved using biologically affordable machinery of bearing-only sensing. Thus, it seems promising to look for examples of optimization by means of embodied approximation in groups of animals. Such animal strategies could and should be used to increase the efficiency, and eventually autonomy, of robot teams.

Admittedly the extent to which embodied approximation can substitute communication or shared memory is limited by the properties of sensors and environment. Limited sensor range and occlusions will limit the number of state vector components that could be accessed by sensing. However, some optimization problems by their nature need only local and limited exchange of information. For other problems embodied approximation may still significantly reduce the need for inter-agent communication and should be considered as one of the available resources.

Future work includes application of embodied approximation approach to other optimization problems emerging in multi-agent teams. A search for biological examples of parallel gradient optimization with embodied approximation is another interesting direction. Finally, accurate mathematical models of spatially embedded multi-agent systems with certain sensing capabilities could be developed and the computation power of such systems can be theoretically studied taking into account embodied approximation as a computational resource.

References

Baudet, G. M. (1978). Asynchronous iterative methods for multiprocessors. *J. ACM*, 25(2):226–244.

Bertsekas, D. P. (1982). Distributed dynamic programming. *IEEE Trans. Automat. Contr.*, 27(3):610–616.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15.

Gueron, S. and Tessler, R. (2002). The fermat-steiner problem. *The American Mathematical Monthly*, 109(5):55–129.

Hamann, H. and Wörn, H. (2007). Embodied computation. *Parallel Processing Letters*, 17(3):287 – 298.

Kung, H. (1976). Synchronized and asynchronous parallel algorithms for multiprocessors. In *Algorithms and Complexity*, pages 153–200. New York:Academic.

Kupitz, Y. and Martini, H. (1997). Geometric aspects of the generalized Fermat-Torricelli problem. *Bolyai Society Mathematical Studies*, 6:55–129.

Litus, Y., Vaughan, R., and Zebrowski, P. (2008). A distributed heuristic for energy-efficient multi-robot multi-place rendezvous. Submitted for publication.

Litus, Y., Vaughan, R. T., and Zebrowski, P. (2007). The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Loizou, S. and Kumar, V. (2007). Biologically inspired bearing-only navigation and tracking. In *Proceedings of 46th IEEE Conference on Decision and Control*, New Orleans, USA.

Payton, D., Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone robotics. *Auton. Robots*, 11(3):319–324.

Polya, G. (1968). *Mathematics and plausible reasoning, 2 vols*. Princeton, 2 edition. vol 1. Induction and analogy in mathematics; vol 2. Patterns of plausible inference.

Rosen, J. and Xue, G.-L. (1991). Computational comparison of two algorithms for the euclidean single facility location problem. *ORSA Journal on Computing*, 3(3).

Tanner, H. G. and Kumar, A. (2005). Formation stabilization of multiple agents using decentralized navigation functions. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA.

Tsitsiklis, J. N., Bertsekas, D. P., and Athans, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. Automat. Contr.*, 31(9):803–812.

Zebrowski, P., Litus, Y., and Vaughan, R. T. (2007). Energy efficient robot rendezvous. In *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pages 139–148.

Zebrowski, P. and Vaughan, R. (2005). Recharging robot teams: A tanker approach. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 803–810, Seattle, Washington.

Zelek, J. (1999). Dynamic discovery and path planning for a mobile robot at a cocktail party. *Robot Motion and Control, 1999. RoMoCo '99. Proceedings of the First Workshop on*, pages 285–290.