In *Proc. of the 9. European Conf. on Artificial Life*
pp. 776-785, Lisbon, Portugal, Sept. 10-14, 2007

Near-Optimal Mobile Robot Recharging With The Rate-Maximizing Forager

Jens Wawerla and Richard T. Vaughan

Autonomy Lab, Simon Fraser University, Burnaby, British Columbia, Canada.
{jwawerla,vaughan}@sfu.ca

Abstract. We examine the practical problem of a mobile autonomous robot performing a long-duration survey task, during which it must recharge its batteries periodically. We present a scalable, online, heuristic method that allows the robot to recharge efficiently, thus maximizing its rate of work. The method is a direct application of the *rate-maximizing foraging* model, which seeks to explain the behaviour of animals solving related problems. Simulation results suggest that the method performs very well compared to optimal and naive heuristic approaches.

1 Introduction

We examine the problem of a mobile autonomous robot performing a long-duration survey task. The robot has limited battery capacity, so it must recharge from time to time. Time and energy spent in recharging are not available for doing work, so, to maximize the robot's work-rate, and thus its utility, we should recharge as efficiently as possible. Computing the perfectly optimal robot behaviour is computationally intractable or impossible in typical circumstances, so we seek useful heuristic methods.

This paper presents a version of this task that serves a real-world need, along with a practical solution: an heuristic method that is a direct implementation of a model of animal foraging behaviour, with some attractive features. We provide empirical simulation results comparing the forager's performance with the optimal behaviour and naive heuristics, suggesting that the new method performs well.

1.1 Problem domain and motivation

A single, self-sufficient and self-controlling mobile robot must visit a (possibly very long) series of waypoints. The number of waypoints, the energy cost of travelling between them and the robot's limited onboard energy storage capacity mean that the robot must recharge its onboard energy store (we assume a battery), possibly many times. The series of waypoints could be given *a priori*, or could be dynamically generated based on previous data. An examples of such a scenario is the Networked Aquatic Microbial System (NAMOS) robotic boat devised by researchers at USC and UCLA [9] that measures the spatio-temporal

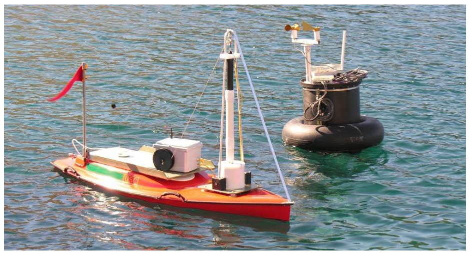


Fig. 1. Application: the USC/UCLA NAMOS robot boat: a mobile aquatic observatory with anchored buoy base-station. A docking/recharging system is under development.

distribution of plankton, and related phenomena, in bodies of water. In this application, a series of waypoints is generated by an adaptive sampling algorithm that attempts to maximize the value of the next sample [12].

We are collaborating with the NAMOS team to add energetic self-sufficiency to the NAMOS boat. Currently the duration and physical size of a water survey performed by the boat is limited by its battery capacity. Adding autonomous recharging will enable the boat to do multi-day or -week long missions without the expense of human intervention, and providing more, better, data to the team's biologists.

1.2 Related work

Research has been done on many aspects of self-sufficiency; ranging from docking mechanisms and control, action-selection, energy-efficient path-planing, to unconventional robot fuels [2]. The work most relevant here is that on energy-aware path-planning and when-to-recharge decision making.

Aside from the huge literature on general cost-minimizing optimization and path planning, we see some work dedicated to energy gathering and recharging. Wettergreen et al. [11] present a solar powered robot that plans motion synchronous to the sun in order to maximize energy gathered by solar cells. The problem of path-planning while minimizing energy expense is also discussed in [10]. Both these authors use conventional planning methods.

Litus et al. [3] describe heuristic methods to achieve energy-efficient rendezvous between robots, with suggested application to robot-to-robot recharging.

As for deciding when to recharge, the standard method is a fixed threshold policy, where the robot switches to recharging activity once its energy level drops below some predefined threshold. Jung's docking work employs this policy [6], for example. Austin et al. [1] have the robot recharge after a constant time has elapsed, approximating the energy threshold while being easier to measure. The threshold approach is attractively simple, but its efficiency depends critically on selecting a suitable threshold. In the experiments below, we use a fixed threshold strategy as a baseline performance target.

An interesting conceptual extension is to a multi-robot threshold. In a multi-robot scenario with one shared charging station, Michaud et al. [4] use a threshold on the 'remaining operational time' value. This value takes the energetic state of all robots into account and is calculated for each robot. This allows the robots to cooperate and share a charging station.

A more carefully motivated approach to decision making under self-sufficiency is discussed by Spier and McFarland [7]. Here a 'cue-deficit' model is used to determine the action to be taken. This work is closest to our work, since it investigates choosing between multiple resources. The major difference is that Spier considers multiple resources, for each of which the robot has a certain deficit, whereas we consider one resource: energy, which can be obtained from multiple sources.

In the remainder we state the problem formally, describe three different algorithms to solve it and compare their performance to the optimal solution.

2 Problem Statement

A mobile robot must visit an ordered series of n points $P = p_1, p_2, ...p_n$ in the shortest possible time. We are especially interested in very large series, so that brute-force planning is infeasible. For simplicity of presentation we assume all points are co-planar, but the solution methods do not require this.

In addition to the robot, the world contains one charging station at a known location c. The robot has two options for recharging: it can recharge from the charging station with a known charging current I_c or it can use on-board solar cells and recharge with a current of I_s. To use the charging station, the robot must be at c. The solar cells can be used at any location, but we assume that the robot cannot drive while recharging from the solar cells [1] but that solar charging has negligible other cost. We further assume the driving velocity v of the robot is constant and known, and so is the current required for driving I_d. The battery which powers the robot has a known capacity of B_{max} and an instantaneous charge B where $0 \leq B \leq B_{max}$. Below we neglect the units and assume that current is measured in amps (A), time in seconds (s), distance in meters (m), speed in ms^{-1}, and energy storage capacity in As. The robot begins with a full battery charge.

The robot's recharging model has this simple form:

1. When reaching a charging station, the robot recharges completely, i.e. until $B = B_{max}$.
2. If the robot runs out of energy, and is not at a charging station, it can always use its solar cells to recharge. It recharges only enough to reach the next goal point.

This is a sensible model because there is always a locomotion cost involved in going to the charging station. In order to maximize the return on this cost, the

[1] This models, for example, the robot needing to unfold the solar array before use.

robot should gain as much energy as possible from a charger visit. There is no locomotion cost for using solar cells, but the time spent recharging is lost for working: an opportunity cost. Since in most scenarios the solar current is smaller than the charger current, it makes sense for the robot to only solar charge as long as necessary to reach the next goal location. Thus, when solar charging, sample points and chargers are reached as early as possible and with an empty battery, which maximizes the benefit of a charging station.

3 Solutions

3.1 Brute-force Optimal

Optimal solutions can be calculated with the following simple brute force method. Build a binary tree with the first point p_1 as the root node. The first (left-hand) child node corresponds to point p_2. The cost of traversing from the root to this child is the time required to travel from p_1 directly to p_2 including any required solar cell recharging time (should the distance be greater than that allowed by a single battery charge). The second (right-hand) child node corresponds to the charger location C. The cost of traversing from the root to this child is the time required to travel from p_1 to the charger C then fully recharge the battery and afterwords travel to p_2, again including possible solar charging time. This step is repeated recursively for each child until the last way point is considered. To find the most efficient solution one finds the leaf node with the shortest travel time. Traversing back from this node gives the optimal work-and-charging strategy.

With complexity $O(2^n)$ this method is not practical for large series. It can not be applied to infinite series or dynamically-generated series, such as those created by an adaptive sampling strategy. It is included here only to help evaluate the heuristic methods.

3.2 Fixed Threshold

Here we use remaining battery charge B as our decision variable. For lack of any general principled approach, we use the following conservative strategy: find the largest distance between any waypoint and the charger $l = \max\{d(p_i, c) | 1 \leq i \leq n\}$ where $d(a, b)$ is the distance between points a and b. The charging threshold is set so that the robot has enough energy left to reach the charger even in the worst case, that is the waypoint that is furthest away from the charger. Therefore the threshold is $B_t = \frac{l}{v} I_d$. This means that the robot chooses to travel to the next waypoint via the charging station if $B < B_t$. Note, this does not guarantee that the robot will always be able to reach the charger, in cases where I_d is high or B_{max} is low, the threshold might actually be higher the B_{max}. In cases like this the robot can rely on power from the sun.

Without any empirical analysis we can see that the simple fixed threshold strategy will not perform very well, because it does not take the distance of the current waypoint to the charger into account. This can lead to situations where the robot is unnecessarily cautious and thus sacrifices performance. An adaptive threshold may overcome this limitation.

3.3 Adaptive Threshold

With the adaptive threshold, we set the threshold for each path segment by checking if the robot has enough energy to travel from the current waypoint along the current path segment and still make it from the next waypoint to the charger. First we calculate the travel distance $l = d(p_i, p_{i+1}) + d(p_{i+1}, c)$ where i denotes the waypoint the robot is currently at. Then the threshold is again $B_t = \frac{l}{v} I_d$. So the robot travels to the next waypoint via the charger if $B < B_t$ otherwise it takes the direct route. Besides taking locality into count, this method has the benefit that if only requires information about the next waypoint, so the path can be generated dynamically, as required by our NAMOS application.

Performance problems may still arise from the usage of a threshold. There are situations where it is beneficial to refuel even though the battery is not even close to empty, but since the energy source is close by the overhead is very small in doing so. The next method attempts to take this into account without increasing computational complexity.

3.4 Rate Maximization

This method is an application of foraging theory (see [8] for an excellent exposition). This body of theory attempts to explain and give a model for animal behaviour while foraging, that is, gathering food and other resources. The *rate-maximizing forager* model attempts to explain the choices animals make when presented with alternative food sources, and maps neatly onto our problem.

Specifically, the rate-maximizing forager model predicts that an animal faced with several exclusive food options will maximize the net energy rate

$$R = \frac{\sum_{j=1}^{m} p_j \lambda_j e_j}{1 + \sum_{j=1}^{m} p_j \lambda_j h_i}$$

where e_i is the net energy gain from food item i, h_i is the handling time, λ_i is the encounter rate and p_i is the attack probability for this item. Without reproducing the details of [8] a rate maximizing forager applies the zero-one rule and attacks always only prey that yield a higher net energy rate than the combined rates of all food items that considered individually have a lower rate. Note that this model describes a solitary forager: Seth [5] provides a probablistic model that may better describe the behaviour of multiple foragers in a competitive situation.

We turn our robot into a rate maximizer by the following approach. For a given path segment $S_i = \overline{p_i p_{i+1}}$, starting at p_i and ending at p_{i+1}, we have to consider two energy gain rates. In case the robot travels directly to the next waypoint, the rate equals the solar current, that is $R_{d_i} = I_s | \forall i < n$. In the other case where the robot goes to the charger first, fully recharges its battery and then travels to the next waypoint, the energy gain rate is

$$R_{c_i} = \frac{I_s T_{s_i} + I_c T_{c_i} - I_d T_{d_i}}{T_{s_i} + T_{c_i} + T_{d_i}} \tag{1}$$

Here T_{s_i} is the time spent charging from solar cells on segment i, which only occurs if the robot cannot make it to the charger or from the charger to the next waypoint p_{i+1} on the remaining battery capacity B_i. The time spent recharging from the charging station is

$$T_{c_i} = \frac{B_{max} - B_i - I_d(|\overline{p_i c}|)}{I_c}$$

In most case the robot has to take a detour to reach the charger, therefore we subtract the extra energy spend from the energy gained. The time required for the detour is

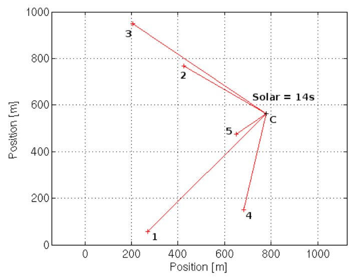
$$T_{d_i} = \frac{|\overline{p_i c}| + |\overline{c p_{i+1}}| - |\overline{p_i p_{i+1}}|}{v}$$

Now the robot can select the option which provides the highest energy gain rate, but it has more options to choose from if it's battery is full enough. What about the next path segment or the one after that? At any given waypoint, the robot only has to make a decision about going direct or via the charger to the next waypoint. So if the robot chooses to go via the charger we can stop our rate analysis for this segment, since future segments do not influence the robot's decision anymore. But if we do not recharge, it is worth evaluating the energy gain rate of the next segment. Again going from p_{i+1} directly to p_{i+2} is trivial $R_{d_{i+1}} = I_s$. In the other case where the robot travels via the charger to p_{i+2} we use eq. 1. But this time for the segment $S_{i+1} = \overline{p_{i+1} p_{i+2}}$ and we also have to calculate B_{i+1} by subtracting what we spend on the first segment so $B_{i+1} = B_i - \frac{|\overline{p_i p_{i+1}}|}{v} I_d$. This way we can iteratively calculate the energy rate for each segment. We can stop this iteration once the projected battery level B_k reaches zero. Because the robot has to recharge at this point, future segments do not influence the current decision.

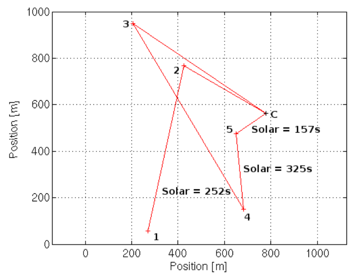
The rate-based decision is simply, if the energy gain rate from the charger option of the current segment $R_{c_i} = \max\{I_s, R_{c_i}, R_{c_{i+1}} ... R_{c_k}\}$ choose to recharge from the charger now, otherwise proceed directly to the next waypoint and re-iterate.

The complexity of this method is $O(kn)$ where is k is the number of waypoints the robot can maximally reach with one battery charge.

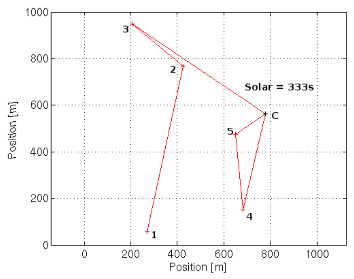
Rate maximizing foraging animals are clearly not limited to two energy resources, as Stephens and Krebs [8] show. We restricted ourselves because of scalability issues of the optimal method. But extending the rate maximizer to deal with m energy sources is straight forward. On each segment s_i we calculate the energy gain rate R_{j_i} for each energy source j. Combinatorial problems do not occur since we terminate the current analysis on each branch on which the robot recharges. As in the two resource case we interactively calculate rates till the battery is depleted. The charging option with the highest rate is the one the robot selects. In this case the complexity is $O(knm)$ with k being the number of waypoints the robot can visit with one battery charge and n is the total number of waypoints of the trajectory.



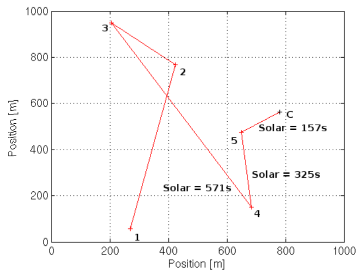
(a) Fixed threshold (t=6955s)



(b) Adaptive threshold (t=5742s)



(c) Rate Maximizer (t=4544s)



(d) Optimal (t=4446s)

Fig. 2. An example problem, where each of the four methods found a different solution.

3.5 Example solutions

Before discussing the statistical results, we give an example of the type of plans produced. The example problem shown in Fig. 2, five waypoints, one charging station, is interesting in that each of the four methods produced a different solution, and typical in their relative performance. The robot starts with a full battery at waypoint 1. To ensure all robots finish in the identical state (which masks artifacts caused by experiments with few waypoints), the charger C is the final waypoint and the trial ends when the robot is totally charged.

The fixed threshold method (Fig.2(a)) demonstrates the conservatism of its high threshold setting, with the robot visiting the charger C 5 times: once between every waypoint. Nevertheless, due to the remoteness of waypoint 3, the robot has to solar charge for 14 seconds on the return from 3 to C. The complete run takes 6955 seconds. The adaptive threshold method (Fig.2(b)) visits the charger between waypoints 2 and 3. It spends a long time solar charging towards the end of the run, which takes 5742 seconds to complete. The rate maximizer method (Fig.2(c)) takes only 79% of the adaptive threshold time. It recharges slightly later: in between waypoints 3 and 4, taking a detour that requires a solar charge, but having visited the charger empty, the robot completes the run without charging again. The optimal method (Fig.2(c)) is very slightly faster than the rate maximizer, taking 99% of the time. The brute-force search found a solution that recharges only at the final return to the charger, and uses the solar cells to arrive there empty.

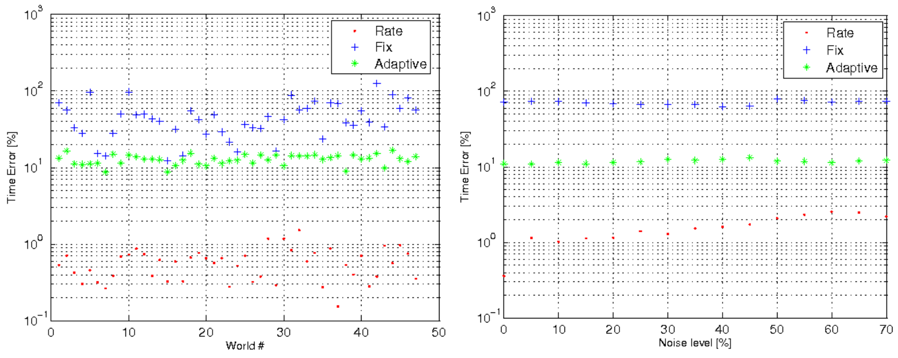


Fig. 3. (Left) Exp.1: The mean error, compared to optimal performance, of each of the three heuristic methods, over a range of parameter settings. Note the Y axis is a log scale. (Right) Exp.3: Performance of the Rate Maximizer under cost estimate error.

4 Experiments

4.1 Experiment 1: series of 20 randomly-placed points

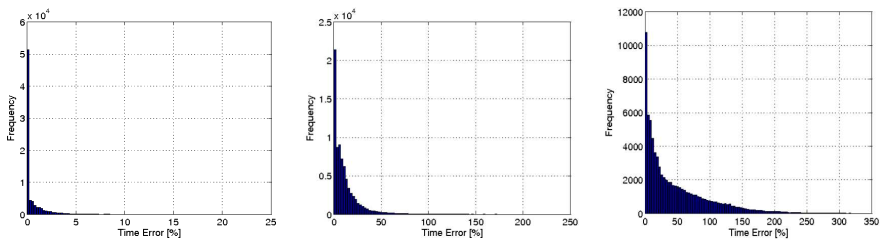
Though we are interested mainly in large-scale instances of the surveying-and-charging problem, we also wish to evaluate the performance of the heuristic methods compared to the optimal. To allow for reasonably fast computation of the optimal brute force method we are limited to 20 waypoints.

We explored the performance of all four methods under a wide variety of parameters. The number of waypoints and the charging station current were fixed. The rest of the setup was as follows: (i) generate a world with 20 waypoints and a single charging station placed at random; (ii) set values of B_{max}, I_s, I_d; (iii) for each of the 4 methods, measure the time it takes the robot to complete the course (again, ending up fully charged at C). Repeat this process for 50 different worlds, each with 34 different values of B_{max}, 7 of I_s and 7 of I_d, for a total of 1666 configurations per world, and 83,300 trials overall. Parameter ranges were chosen to explore the most significant part of the state space.

Presenting these results is challenging, since so many parameters are varied. The absolute values of performance are not very meaningful since it depends on the length of the trajectory which is randomly generated. Therefore for each trial we subtract the time taken by the optimal method, to give an error signal which we can use to asses the heuristic methods. Figure 3 (left) shows the mean error of each method compared to the optimal solution averaged over all parameter sets in each of the 50 worlds. It shows that the adaptive threshold performs better (has smaller error) then the fixed threshold method and that the rate maximizer performs better then the other two methods. Table 1 (Experiment 1) gives the mean and standard deviation of the time error for all experiments. The distribution of these errors is depicted in Fig. 4, which shows a histogram of the error. Note the axes are scaled differently on all the graphs.

Method	Experiment 1		Experiment 2		Experiment 3	
	μ	σ	μ	σ	μ	σ
Rate Maximizer	0.6	1.3	-	-	1.6	3.9
Fixed Threshold	47.5	51.1	55.8	52.2	70.6	68.9
Adaptive Threshold	12.8	18.7	14.8	28.2	11.8	16.7

Table 1. Summary of statistical results for all three experiments



(a) Rate Maximization (b) Adaptive Threshold (c) Fixed Threshold

Fig. 4. Histograms of the relative error. The differences are so large that the axes are scaled differently to preserve visible detail.

We repeated this experiment with various regular patterns of waypoints, producing very similar results. We omit details here for lack of space.

4.2 Experiment 2: series of 1000 randomly-placed points

To evaluate the methods' performance in long-duration tasks, we generated 100 trajectories with 1000 waypoints each and ran the three heuristic methods over all parameter combinations for each trajectory. Here it is impossible to compare the performance of each method to an optimal solution, since it could not be calculated in a reasonable amount of time for these large problems. Instead we compared the threshold methods against the rate method. Table 1 (Experiment 2) shows how the threshold methods performed compared to the rate maximization. On average the rate maximizer performs 14% better then the adaptive threshold and 55% better then the fixed threshold method.

4.3 Experiment 3: robustness to cost estimate error

We repeat experiment 2, this time randomly varying the actual cost of driving the robot between waypoints, to simulate unbiased error in the robot's *a priori* cost estimates. The error was varied from 0% to 70 % in 5% intervals. 20 simulations per noise level and parameter set were performed.

As Table 1 (Experiment 3) shows, the performance of the rate maximizer decreases with increased cost estimate error. However, the rate maximizer still performs significantly better than the other methods. And as Fig. 3 (right) shows, the error increases with increasing noise but remains relatively small, suggesting that the rate maximizer is fairly noise tolerant.

5 Conclusion

We have tackled the natural optimization problem of deciding where and when for a long-lived robot to recharge, given two alternative energy supplies with different dynamic properties. We gave a scalable, heuristic solution based on a model from the biological literature that explains the choices that animals *should* make when faced by equivalent problems.

The idea behind the solution is to maximize the local *rate* of energy intake. Given finite energy storage capacity this has the natural result of maximizing the rate at which energy can be spent on work. Thus by adopting this strategy we observe that the work-rate of the simulated systems we observed is very close to optimal. Yet the method is scalable, i.e. requires computation and memory independent of the length of the problem. It is simple to implement and appears to be reasonably resistant to error in cost estimates. The method dramatically outperforms naive threshold-based heuristics.

References

1. D. Austin, L. Fletcher, and A. Zelinsky. Mobile robotics in the long term - exploring the fourth dimension. In *Proc. Int. Conf. on Intelligent Robots and Systems*, pp. 613–618, 2001.
2. I. Ieropoulos, J. Greenman, and C. Melhuish. Imitating metabolism: Energy autonomy in biologically inspired robotics. In *Proc. of Symp. on Imitation in Animals and Artifacts*, pp. 191–194, 2003.
3. Y. Litus, R. T. Vaughan, and P. Zebrowski. The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 27–32, Apr. 2007.
4. F. Michaud and E. Robichaud. Sharing charging stations for long-term activity of autonomous robots. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 2746–2751, 2002.
5. A. K. Seth. The ecology of action selection: Insights from artificial life. *Phil. Trans. R. Soc. B*, In Press.
6. M. Silverman, D. M. Nies, B. Jung, and G. S. Sukhatme. Staying alive: A docking station for autonomous robot recharging. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1050–1055, May 2002.
7. E. Spier and D. McFarland. Possibly optimal decision-making under self-sufficiency and autonomy. *Journal of Theoretical Biology*, 189(3):317–331, Dec. 1997.
8. D. W. Stephens and J. R. Krebs. *Foraging Theory.* Princton Uni. Press, 1986.
9. G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron. The design and development of a wireless robotic networked aquatic microbial observing system. *Environmental Engineering Science*, 24(2) pp. 205–215, 2006.
10. Z. Sun and J. Reif. On energy-minimizing paths on terrains for a mobile robot. In *Proc. Int. Conf. on Robotics and Automation*, vol. 3, pp. 3782–3788, Sept. 2003.
11. D. Wettergreen, P. Tompkins, C. Urmson, M. Wagner, and W. Whittaker. Sun-synchronous robotic exploration: Technical description and field experimentation. *The Int. Journal of Robotics Research*, 24(1):3–30, Jan. 2005.
12. B. Zhang and G. S. Sukhatme. Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3673–3680, 2007.