TANKER BASED ROBOT RECHARGING

by

Pawel Zebrowski

BASc., Simon Fraser University, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2007

APPROVAL

Name:	Pawel Zebrowski
Degree:	Master of Science
Title of thesis:	Tanker Based Robot Recharging
Examining Committee:	Dr. Robert Hadley, Professor, Computing Science Simon Fraser University Chair
	Dr. Richard Vaughan, Assistant Professor, Computing Science Simon Fraser University Senior Supervisor
	Dr. Greg Mori, Assistant Professor, Computing Science Simon Fraser University Supervisor
	Dr. Ze-Nian Li, Professor, Computing Science Simon Fraser University SFU Examiner
Date Approved:	2 August 2007

Abstract

Teams of autonomous mobile robots wishing to sustain long term operation need a means of refueling. Traditionally, this has been accomplished by requiring that each robot return to a central charging station. Since energy is a scarce resource, this must be done in an energy efficient way. This thesis proposes an energy efficient tanker based approach for recharging robot teams. A service robot is employed to transport energy from a charging station to other robots. Energy efficient path planning for this robot is shown to be NP-hard. Several novel heuristic techniques for joint robot motion planning to achieve efficient tanker-robot meetings are presented and analyzed. A heuristic is developed which is perfectly scalable, and shown both analytically and experimentally to produce energy efficient robot paths.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robots and energy

Energy maintenance is a key requirement in creating long-lived autonomous robots. An autonomous robot, no matter how sophisticated its Artificial Intelligence, will have its life-span and work-load limited by the available energy. This problem is in common with all living things, and is so fundamental that we believe it may place interesting constraints on the design of intelligent autonomous systems.

Consider a team of robots moving throughout an environment in order to take temperature readings. Since each robot has a finite amount of energy, at some point it will need to be recharged, or cease functioning. The usual approach to solving this problem is to outfit all robots with means to recharge themselves, usually by visiting a charging device at some fixed location. This approach places certain demands on the sensing and computation of the robots, for example finding a fixed known location requires localization, which is certainly possible but can be very computationally expensive to achieve in indoor robots. Yet this technology is mature enough that at least one off-the-shelf recharging system[1] is available for research robots, as is a low-cost domestic floor-sweeping robot, the Roomba Discovery[2] with autonomous base-station recharging.

There is an alternative approach: a robot 'tanker' system in which a special-purpose robot collects energy from a source and distributes it to one or more worker robots. This may have two advantages compared to conventional autonomous recharging: first in terms of cost and complexity of worker robots, and second in overall system efficiency.

[1] http://www.activrobots.com/ACCESSORIES/DXDock.html

[2] http://www.irobot.com/consumer

1.2 The tanker approach

The tanker based approach is suggested as a useful alternative to individual recharging approaches for systems of autonomous robots. In the most simple form, an energy-transporting tanker coexists in an environment consisting of worker robots ('workers') designed to perform some task (for example: forage) and is required to find and recharge worker robots in need of energy, while maintaining its own energy level. Such a tanker must be able to locate and approach worker robots, as well as return to a charging location and recharge itself.

This approach to robot recharging has apparently not been proposed or studied before. The benefits of the tanker approach over a self-charging system can be seen in the division of labor. Worker robots are able to largely retain their task scope, with no need for extra sensing or algorithmic complexity. Tankers have only one task, to search for and recharge workers. Further, this division of labor leads to a division in complexity. For worker robots, the coupling of their intended tasks and the recharging task is not required. As an example, a worker can stop working and wait for a tanker, instead of concerning itself with the complexities of returning to a charging station. A certain level of modularity is also introduced, consistent with the idea of interchangeable parts. This is likely to lead to cost saving and ease of replacement in case of failure.

A further benefit is that of energy efficiency. A team of robots that need to independently travel to a central charging station may not be the most efficient use of energy. Under certain conditions, the introduction of a tanker robot can yield much more efficient energy usage, as will be shown later.

1.3 The need to meet

In order to achieve tanker based recharging, the system of tanker and worker robots needs to have a means of exchanging energy. This is likely to require a tanker and worker to physically interact (for example, connect to each other) in order to transfer energy. For such interaction to take place, a tanker and worker must meet. Further, since the overall system goal is sustaining function given a finite energy capacity, energy efficiency is a key goal in prescribing meeting mechanisms.

Such a system has numerous degrees of freedom which can make the problem difficult to solve. First is the combinatorial problem of deciding in what order the tanker should meet each worker. In all cases except a *rendezvous*, this is a non trivial problem. Following this is the continuous problem of deciding where each meeting should take place and which robots should attend. Next is the problem of scheduling meetings, prevalent when workers discharge at different rates and require

recharging at different times, or when the tanker does not have enough capacity to recharge all workers without recharging itself. Last is the problem involving obstacles, navigation, localization, and interference, all while maintaining energy efficiency. In this thesis we tackle the problem of deciding where to meet, and briefly examine the combinatorial problem as well as the problem of obstacles and interference.

Given the complexity of tanker based recharging, we chose to focus on finding methods for achieving energy efficient meetings. This is done in order to limit the problem scope to a manageable size. Despite showing only partial solutions to tanker based recharging, the results presented here are useful since an increase in efficiency can lead to an increase in system utility.

1.4 Related work

1.4.1 Robot docking and recharging

Autonomous robot recharging has historically been accomplished by having robots return to a central charging station at a fixed location. This technique goes back as far as 1953 when W. G. Walter demonstrated autonomous robot "tortoises" going back to a home base [40]. The tortoises used light following to identify and approach a base which, once entered, established physical contact and allowed for recharging. Since then, numerous designs based on this principle have been presented.

In 1999, Nourbakhsh et al. [26] developed "Sage," an autonomous robotic tour guide to be employed full time at the Carnegie Museum of Natural History. The goal of the project was to develop a highly reliable robotic tour guide capable of giving interactive tours to visitors with minimal service intervention. As part of this goal, Sage had to have energy autonomy, which was accomplished with the help of a charging station. Reliable docking was achieved using a CCD camera and a three dimensional landmark placed directly above the charging station. Sage used a custom built "plug" to interface with the charging station.

In 2000, Oh, Zelinksy, and Taylor [27] presented an approach based on aircraft landing. They used a Nomad XR4000 to divide the task of identifying and approaching the docking station into long range and short range operations. Infrared sensors were used for long range detection of the charging station and to bring the robot close enough for laser target tracking to take over. A physical connection was established with the help of a flexible power plug.

Several other authors present variations for docking and charging autonomous robots. Silverman et al. [35] show similar results using a Pioneer 2-DX robot. Kartoun et al. show a vision based recharging system designed for an ER-1 Evolution Robotics mobile robot in [15]. Emami et al. [12]

show a very tolerant physical design.

It should be noted that docking station based recharging has matured enough as a technology to become useful in commercial applications. iRobot's low-cost domestic cleaning robots make use of a "self-charging home base" in order to recharge.

1.4.2 Path planning

The problem of path planning for multiple robots to assemble in one location, the *rendezvous problem*, has received some attention. Most authors consider the setting where robots have incomplete information about locations of other robots which makes it difficult to coordinate and agree on a single meeting point.

Dudek and Roy [11] study the rendezvous problem for an unknown environment in the absence of communication. They propose several techniques whereby each robot evaluates its environment while exploring, and ranks environmental features to be classified as landmarks. Different methods are used in selecting rendezvous points from the list of suitable landmarks.

Schlude [34] explores the problem of achieving multi-robot rendezvous. He presents a method which results in robots moving towards a common rendezvous point with the help of "Contraction Functions." He identifies similarities of the resulting rendezvous point to the Weber point, prevalent in facility location problems.

Further examples of rendezvous under varying assumptions can be seen in [19] and [9]. Ando et al. [1] describe and prove a distributed rendezvous algorithm for robots with limited visibility. Other authors concentrate on selecting a meeting point to ensure that some specific properties hold, or to optimize the formation during the convergence. Smith et al. [36] describe a scheme which makes the convergence process more organized in a certain mathematical sense.

Lanthier et al. [17] present an algorithm for finding the meeting point which minimizes the maximum individual travel costs to a single meeting point on a weighted terrain. This is in contrast to our work, which considers total system energy usage, and is a problem that has not been addressed previously.

1.4.3 Energy efficiency

Energy efficiency in general motion planning is well studied. The Distributed Energy-efficient Autonomous Robots (DEAR) group of Purdue University has contributed a significant body of work to the study of energy efficiency in mobile robots. In [24], Mei et al. study a method for comparing

the energy efficiency of proposed motion plans. They devise a method which considers turns, accelerations, and velocities in order to calculate expected energy usage of a given path, and compare it to other candidate paths. In [23], they propose an energy efficient exploration algorithm which augments existing exploration techniques to reduce duplicate coverage and employ energy efficient motion planning. Yongguo Mei's PhD thesis [22] thoroughly explores several energy efficiency issues in multi-robot systems.

Wang et al. [41] study energy efficiency in mobile sensor networks. They consider all the actuation costs for a mobile sensor along with several road conditions, and propose an optimal or close to optimal velocity schedule for each. Sun and Reif [37] build on the work of Rowe [32,33] to present results for energy efficient motion planning on more challenging terrains, considering friction and gravity.

1.4.4 Facility location

Energy efficient path planning for one service robot to visit each worker robot, which we will later formally define and name, is novel. However, the analytical component of the problem has similarities to the weighted Fermat-Torricelli problem. Further, as mentioned earlier, Konrad Schlude [34] alludes to the similarities between finding a rendezvous location, and finding the Weber point given all robot locations. The facility location problem class is well studied and has a complex background. Drezner [10] provides a good survey of this family of problems.

1.5 Thesis outline

Chapter 2 The feasibility of tanker based recharging is examined. A tanker is placed in a simulated multi-room environment along with several worker robots and charging stations. A naïve approach is taken to all problems required in achieving tanker based recharging, and some simplifying assumptions are made. Several design techniques are tested and compared, but no optimality results are sought. A description is provided for hardware required in achieving tanker based recharging.

Chapter 3 The rendezvous problem is introduced, and techniques presented for achieving rendezvous in an energy efficient manner. Two methods are compared in different environments and energy consumption examined.

Chapter 4 The frugal feeding problem is introduced, and shown to be NP-hard. Special cases of the problem are identified and solutions suggested. Discrete and numerical techniques are described for solving the meeting location component of the frugal feeding problem.

Chapter 5 The frugal feeding heuristic is presented and analyzed. A variety of experiments are performed to demonstrate the utility of the heuristic as well as the Nelder-Mead numerical solution from the previous chapter. The effects of obstacles and interference are examined. Complete solutions to the frugal feeding problem are demonstrated by pairing both the frugal feeding heuristic and Nelder-Mead with ordering algorithms. Computation time requirements are discussed.

Chapter 6 Tanker based recharging results are discussed and compared to traditional docking station recharging. Future work is presented, along with an outline of unsolved problems remaining in tanker based recharging.

Chapter 2

Tanker Recharging

The work elaborated in this chapter is based on the work by Pawel Zebrowski and Richard Vaughan [45], presented at the International Conference on Advance Robotics (ICAR 2005), Seattle, Washington, July 18-20, 2005.

2.1 Feasibility

To begin, we set out to examine whether tanker based recharging is a feasible approach to recharging robot teams. We start by adopting a naïve approach to problems encountered during design. The goal is to demonstrate experimentally that under some set of conditions, a tanker is able to maintain acceptable levels of energy in itself and several worker robots. This is done by conducting experiments in a simulated real-world environment using a number of worker robots and several charging stations. Care is taken to achieve an accurate simulation, although several simplifications are assumed intentionally as part of the naïve approach. Several design techniques are demonstrated and evaluated.

2.2 Experiments

The experiments in this chapter demonstrate tanker based recharging in simulation using Player/Stage [13]. The simulated world is a two dimensional floorplan map of a hospital (a standard Stage environment) constructed from a blueprint of this hospital. Within the world is one tanker robot and a number of worker robots, each able to traverse the hospital rooms and corridors.

Each robot in this world is equipped with an energy device which allows for the simulation of energy depletion and acquisition. The energy device consumes energy at a rate proportional to the number of devices on a robot and the mass of the robot (as defined in the simulation). For example, a Pioneer 3-DX based robot consumes less energy per time unit than a Nomadic Technology Nomad 200 based robot equipped with a laser range finder. It should be noted that a motionless robot continues to consume energy at a slow rate to simulate energy consumed by circuitry.

The world contains within it 'energy squares' which are idealized charging stations. The energy squares have the ability to give unlimited amounts of energy. Unlike a robot, which can acquire, store, and use a finite amount of energy, an energy square models a wall outlet or other public utility connected energy source. Each robot is equipped with a 'nose' that is used to transfer energy between charging location and robot, or robot and robot. This is an abstraction from the physical mechanisms required to achieve such a system. See Section 2.5 for an example implementation.

2.2.1 Devices

The tanker is modeled after a Nomadic Technology 200. It is simulated by a sixteen-sided polygon with sixteen sonar range sensors. The tanker is also equipped with a laser range finder modeled after a SICK LMS laser range finder. These devices are used for tanker navigation throughout the hospital. The workers are modeled after a Pioneer 3-DX. They contain sonar range sensors, which are used for navigation.

Each robot is equipped with a dynamically changeable fiducial device, used primarily to distinguish a worker in need of energy from all other workers. Any worker robot with fiducial id $fid > 200$ is considered to be in need of energy. On real robots, this could be achieved using colour indicator lights and a blobfinder device. Fiducial id's are also used for statistical purposes, and to keep a list of energy awaiting worker robots when the tanker cannot immediately recharge them. This, however, is not crucial to the successful implementation of our experiments.

2.2.2 Controller

The tanker controller is implemented based on the Subsumption Architecture [7]. This approach is chosen as the robot needs to perform a series of simple tasks which have a static priority that is easily defined. The general groups of tasks are defined as follows, in order of priority:

1. maintain mobility by avoiding obstacles and cyclic motion,

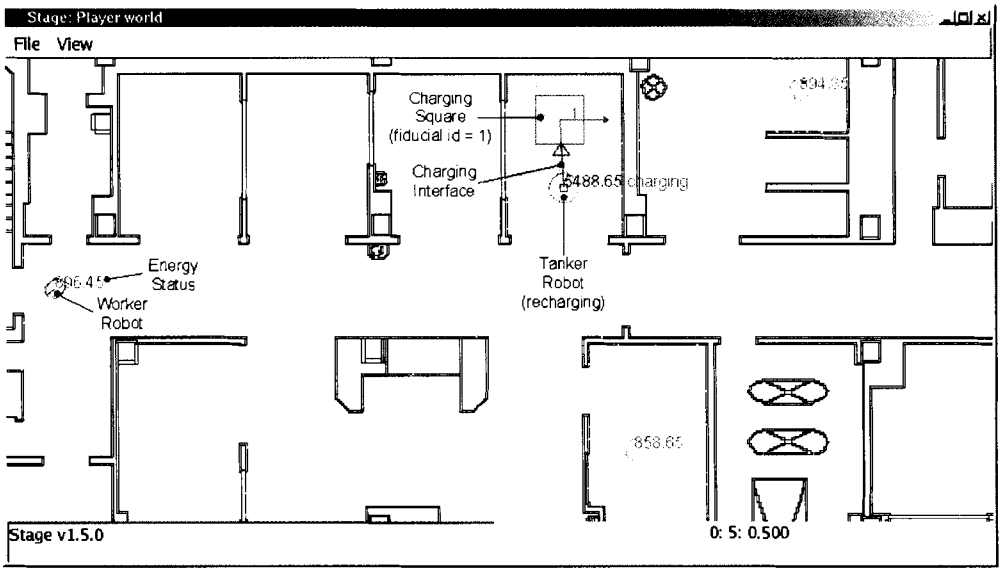


Figure 2.1: A portion of the simulated world.

2. find charging locations and return to a charging location if tanker energy is below threshold, and

3. find worker robots in need of energy and recharge them.

The state transition diagram in Figure 2.2 defines the states and transitions of a tanker. Notice that this state transition diagram is not dependent on any particular method, but is adaptable based on environment. For example, the "return to energy" state is implemented using a breadcrumb trail-following method that combines some features of planning and mapping [39] but can easily be replaced with a more complex localization scheme.

Execution begins in the energy search state, where the tanker's goal is to find a charging location. Finding a charging location is key to sustaining energy beyond the initial amount provided, since the tanker depends on a reliable source of energy. No assumption is made that the energy locations are known *a priori*. Once a charging location is found, the tanker begins a cycle of searching and charging worker robots, and returning to a charging location to recharge itself. This cycle continues indefinitely. The tanker always chooses to recharge itself before recharging workers, in order to sustain functionality.

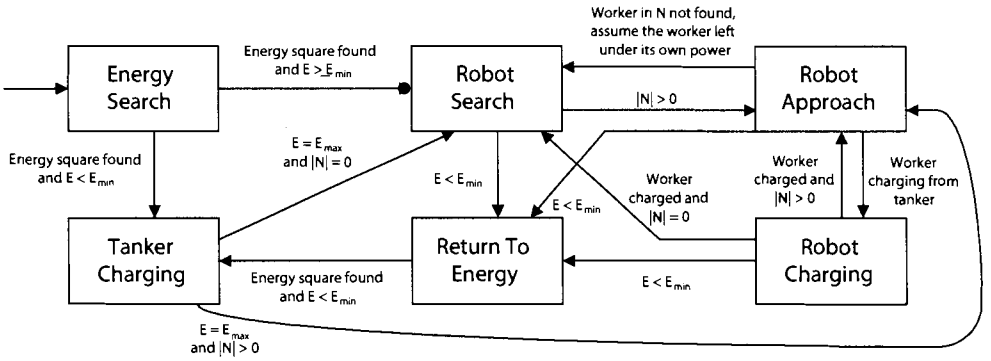


Figure 2.2: The tanker state transition diagram.

2.2.3 Implementation

Data structures

The tanker depends on several data structures that define the action of the robot. Let the 'worker vector' N be a list of tuples $< fid, x, y, \theta >$ (fiducial id, position, and orientation) which identifies the last seen location of all workers discovered as being in need of energy. Let the breadcrumb trail vector B be a list of tuples $< x, y, \theta >$ (position and orientation) which identifies a series of points along the path to the last seen charging square. Let the energy threshold E_{min} be the amount of energy below which the tanker begins seeking a charging location to recharge, and E_{max} be tanker's maximum attainable energy level.

Program flow

In the energy search state, the tanker moves around its environment in an attempt to locate a source of energy. An energy square is identifiable by a range of specific fiducial ids. While in this state, the tanker adds the location of any workers in need of energy to the worker vector N, but does not proceed to help them at this time. Upon finding a charging station, the tanker begins dropping breadcrumbs during all subsequent actions in order to reliably return to this location when needed. When a charging location is once again identified, a new breadcrumb trail is started.

In the robot search state, the tanker looks for worker robots in need of recharging. Upon locating one, the tanker adds the robot's location information to the worker vector N. When the worker vector N is non-empty, the tanker proceeds directly to the location of some robot in N, determined by the chosen charging order method. During this worker approach state, the tanker continues noting other

worker robots in need of energy. In the worker recharging state, the tanker transfers energy from itself to the worker robot.

When the tanker's energy level $E < E_{min}$, the tanker immediately begins following breadcrumbs in reverse order to get back to the last seen charging location. Once at the charging location, the tanker is able to recharge itself. When finished, it resumes looking for robots to add to the worker vector N, or approaching some robot in N.

Navigation

The tanker achieves obstacle avoidance with help from the Vector Field Histogram (VFH) method [5]. VFH attempts to guide the tanker in traversing its environment from its current pose p to some specified pose p' while avoiding static and dynamic obstacles. It does this by constructing a two-dimensional Cartesian histogram grid as a world model, then reducing this grid into a one-dimensional polar histogram centered around the robot. This polar histogram describes the obstacle density around the robot, and is used to select the most suitable direction of travel.

Since VFH is a local path planner, its use may result in cyclic motion while trying to negotiate local minima in the environment. To suppress this cyclic motion, a cyclic motion detection technique is used. This technique is achieved by placing a virtual 'tail' on the tanker consisting of n breadcrumbs. Let p_i be the position of the tanker at time i. Let T_i be the tail at time i. Let j and k be times. $T_j =< p_0, p_1, ..., p_j >$ for $j < n$ and $T_k =< p_{k-n}, ..., p_k >$ for $k \geq n$. The tail can then be thought of as a series of the last n breadcrumbs taken at some regular interval. The algorithm compares the current pose p_c to every location in the tail T_c. Confidence C that the tanker is in a cycle is then defined as the number of p_i that are near p_c. In this implementation, p_i is considered near p_c if it is within one meter from p_c.

Cyclic motion confidence C remains low as long as the tanker traverses an area not seen within the last n time units. However, if the tanker becomes trapped in a short cycle, the number of tail points p_i near p_c will increase. When confidence C reaches some threshold the tanker begins taking evasive action, in this case moving towards an open area until the confidence drops below the threshold. This occurs as soon as the tanker leaves the area in which the cycle occurred, thereby stopping the cyclic motion and returning control to normal navigation.

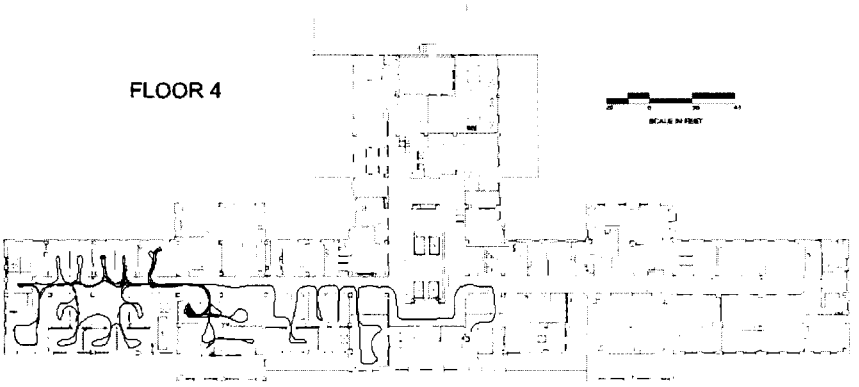


Figure 2.3: Typical path taken with right wall following.

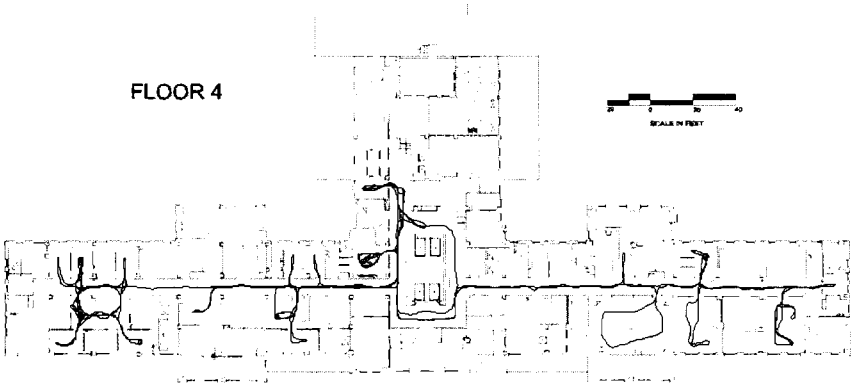


Figure 2.4: Typical path taken with random walking.

Searching

The tanker spends most of its time searching for worker robots and energy locations. Two searching techniques are explored: right wall following and random walking.

Right wall following is achieved with the help of VFH, by setting the destination in front and to the right of the tanker's current location. See Figure 2.3 for a typical path taken during one simulated hour.

Random walking is defined as follows. Every $t = 10$ seconds, a new destination d is chosen in an area $x = -10...10$ and $y = -10...10$ where the tanker is defined to be at $(0, 0)$. A bias towards destinations in front of the robot is added to promote non-cyclic movements. Should the tanker come within 1 meter of destination d before time t elapses, a new destination d' is chosen immediately. See Figure 2.4 for a typical path taken during one simulated hour.

Return to energy

Reliable return to a charging location is assured with the help of a breadcrumb trail algorithm. A breadcrumb $b_t =< x, y, \theta >$ is a coordinate at time t. The breadcrumb trail T_n after last seeing the charging station n seconds ago is defined as $T_n =< b_c, b_0, b_1, ..., b_n >$ where b_c is the location of the last seen charging station. When the tanker reaches an energy level $E < E_{min}$, it stops performing its current action and proceeds to the last dropped breadcrumb b_n. Upon coming within 1 meter of b_n, T is truncated after $b_n - 1$, and the tanker proceeds to $b_n - 1$. This continues until reaching b_c and therefore the charging location. An optimization is added such that should the tanker be within 1 meter of any other breadcrumb b_j where $j < n$ and $j \leq k$ for all b_k within 1 meter of the tanker, the list of breadcrumbs is truncated after $b_j - 1$. This results in the tanker not repeating any cyclic behavior it performed during searching.

Worker robot recharging

The tanker identifies that a worker is charging from it by monitoring energy usage per second (watts). When watts used increases above the nominal rate, the tanker concludes that some robot is charging. The tanker then immediately stops moving to facilitate the charging. While charging, at intervals of 30 seconds, the tanker initiates a 360-degree rotation during which all robots within 1 meter and facing the tanker are identified as charging. Their id's are noted and removed from the worker vector N. When the tanker's watts fall back to the nominal rate, the tanker resumes its previous task.

2.2.4 Procedure

Experiments are run in simulation using Player/Stage to demonstrate tanker based recharging. An experiment consists of 7 simulated one-hour trials. Each trial begins by placing the tanker and 9 worker robots in the simulated hospital world. A worker robot is to explore its environment, avoiding obstacles, until its energy level reaches some threshold, at which point it is to indicate the need for energy, stop moving, and wait for the tanker. Even while motionless, the worker continues to consume energy.

A record is kept of the tanker's state every simulated second. From this, the following data is used to gauge experiment performance:

1. number of worker robots in need of energy found,

2. number of worker robots recharged,

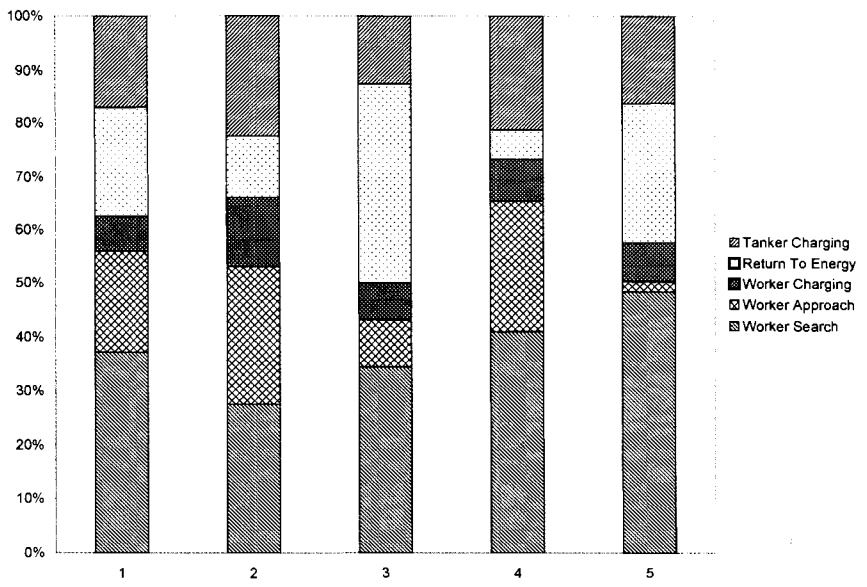


Figure 2.5: Percent of time spent in each state: 1. Random walking, breadcrumbs, worker vector, FIFO. 2. Right wall following, breadcrumbs, worker vector, FIFO. 3. Random walking, breadcrumbs disabled, worker vector, FIFO. 4. Random walking, breadcrumbs, worker vector disabled, FIFO. 5. Random walking, breadcrumbs, worker vector, LIFO.

3. percentage of time spent in each state, and

4. mean energy.

This data is used to gauge the effectiveness of several performance enhancements as well as certain design tradeoffs. Two performance enhancements tested are breadcrumb trails when seeking energy, and the worker vector. The design tradeoffs tested are two different search strategies, and two processing techniques for the worker vector.

2.3 Results

2.3.1 Searching

Two techniques are explored for searching: random walking and right wall following. Right wall following provides for a detailed search with a slow coverage of the entire environment, while the random algorithm covers the entire environment with little detailed search. As a result, robots further

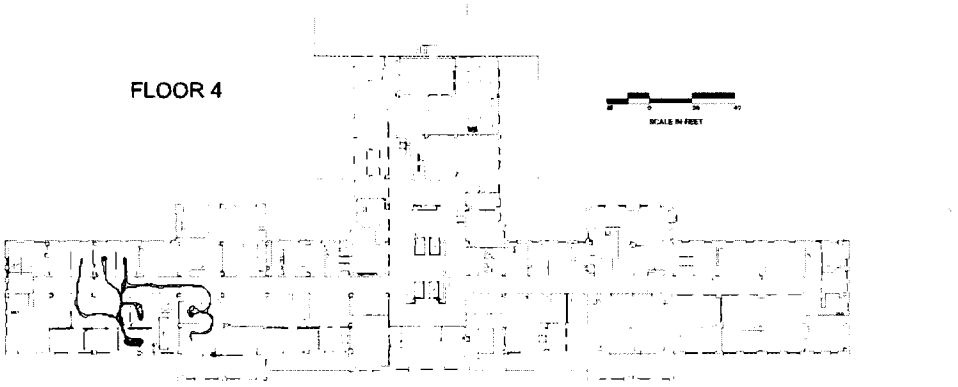


Figure 2.6: Typical path taken with worker vector disabled.

away are usually only discovered by the random algorithm, while robots hidden in rooms are only discovered by right wall following. Figure 2.4 and 2.3 demonstrate the coverage of each searching algorithm. Table 2.1 shows that right wall following performed significantly better than random walking, finding an average of 14 worker robots in need of charging, as compared to 4.5 using random walking. Further, robot searching only consumed 28% of the tanker's time while using right wall following, compared to 37% with random walking.

Table 2.1: Mean number of workers found varying search technique.

	Mean number of workers found
Random Walking	4.5
Right Wall Following	14

2.3.2 Worker vector

We found that the worker vector reduces the mean time spent searching for depleted workers by 4%. It is interesting to note that a significant reduction of time spent in the energy seeking state is also noticeable. With the worker vector enabled, 21% of time is spent returning to a charging location, as compared to 5% with the vector disabled. Examining a typical path taken with the list disabled (Figure 2.6) shows that this is a result of the tanker staying closer to the charging location, and often retracing its path several times after recharging.

Table 2.2: Mean number of workers recharged with and without the worker vector.

	Mean number of workers recharged
Enabled	3.25
Disabled	3.66

2.3.3 Breadcrumb trail

Experiments were performed with the breadcrumb trail enabled and disabled, storing only the pose of a charging location. Out of 7 trials with breadcrumb trail disabled, 2 terminated prematurely because the tanker could not return to a charging station before becoming depleted of energy. Further, the mean percent energy level in trials with breadcrumb trails enabled is 72% compared to 55% with breadcrumb trails disabled. The percent of time spent returning to a charging location with breadcrumbs enabled is 21%, compared to 38% disabled. This leads to a conclusion that keeping a breadcrumb trail reduces the amount of time spent returning to charging locations, which allows for more time to be spent searching for depleted workers.

Table 2.3: Mean percent energy level with and without the breadcrumb trail.

	Mean percent energy level
Enabled	72%
Disabled	55%

2.3.4 Charging order

Two methods of charging depleted worker robots were tested: first in, first charged (FIFO) and charge last seen robot first (LIFO). First in, first charged guarantees that each robot will eventually be recharged, while charge last seen robot first makes the optimization of attending to closest robots first, resulting in better performance, but potential starvation. On average, first in first charged resulted in 19% of time being spent seeking depleted workers, while charge last seen robot first spent only 2%. First in first charged resulted in 37% of time spent searching for depleted workers, as compared to charge last seen robot first, with 49%. Starvation did not occur in any of the trials.

2.4 Discussion

We have shown that the tanker method for recharging systems of autonomous mobile robots is a feasible solution to the recharging problem. Several techniques are explored and compared to yield

a functioning tanker robot in simulation. This section discusses the significance of results obtained as well as possible improvements.

2.4.1 Search techniques

The two search techniques tested produced varying results. Right wall following can be seen to charge significantly more workers than random walking, and it does so with less time spent searching. However, the advantage of right wall following over random walking does not indicate a superior search technique. Both methods perform nearly identical tasks: navigate the environment due to some algorithm without storing historical data, planning, or otherwise intelligent reasoning. The reason that right wall following proves superior in our experiments is because the majority of worker robots tend to remain in rooms, often far away from corridors. This, however, is not true in general. Consider a system of transport robots. It is foreseeable that such a system will favor spending time in corridors between rooms, making random walking a more effective search algorithm. Given this, a provably superior search algorithm may need to be based on a search strategy similar to frontier-based exploration [43] or some other more complex method.

2.4.2 Worker vector

Table 2.2 shows that the worker vector does not result in an increase in the number of worker robots recharged. It is however witnessed that a slight reduction (4%) in the percent of time spent searching for workers does occur. It is conjectured that the lack of a perceivable performance benefit results from the following phenomenon.

It is observed that without the worker vector, the tanker stays significantly closer to the last seen charging station than with the vector enabled. This is once again attributed to the simplicity of the search technique. After tanker recharging is complete, the tanker commences its search strategy. Having no worker vector to remind it where it left off, it begins just as before, retracing large portions of the same path already traversed after the last recharge. In this sense, the worker vector acts more like a bookmark to remind the tanker where it left off before returning to recharge. This results in a more complete coverage of the entire environment. A more sophisticated search algorithm would likely account for this.

By remaining close to the last seen charging station, the tanker is able to traverse an area populated with robots at more frequent intervals, thereby having more opportunity to recharge workers in this area. This results in workers near the charging location receiving immediate assistance. With

the worker vector enabled (and therefore broader coverage) a worker must wait longer before being recharged. It is therefore reasoned that the lack of perceivable performance benefit resulting from the worker vector is due to the tanker recharging the same group of robots repeatedly, thereby artificially increasing the charged worker count, instead of recharging all robots less frequently. This, of course, is not the desired result, as the goal is to sustain the energy level of the entire system by recharging all worker robots. Without the worker vector, a subset of workers is never discovered, and never recharged.

2.4.3 Breadcrumb trail

It can be seen from the experiments that a tanker is capable of reaching some distant target without the help of breadcrumb trails. This is witnessed when, with the worker vector enabled, the tanker is able to find the last worker seen, even if the location is a long distance away. The same cannot be expected in real world experiments with imperfect localization. The breadcrumb trail is used as a mechanism for reliably returning to an energy source, regardless of localization properties. It is shown in [39] that breadcrumb trails are reliable even with imperfect localization, which allows the tanker to make no assumptions about localization properties *a priori*. We conjecture that breadcrumbs would have further improved performance had they been employed in the worker vector as well.

2.4.4 Charging order

Charging order is seen to have a significant influence on the percentage of time spent seeking workers in the worker vector. The reason for this is clear: having traversed a path encountering n workers, LIFO allows for the robots to be charged in reverse order. However, with FIFO, the tanker needs to traverse the path again, in reverse, then begin charging robots in the original order added. This extra traversal consumes time that can be used for other tasks. Further, should the tanker encounter any new workers while traversing this path, it will have to perform yet another cycle.

It should be noted that LIFO is in theory susceptible to starvation. The tanker may never get the chance to approach the first worker seen if others keep being recharged. While not witnessed in experimentation, it is easy to set up a starvation scenario. For this reason, LIFO should not be considered a superior algorithm unless starvation is acceptable.

A more suitable algorithm would be one that allows for the path optimization of LIFO, but is not susceptible to starvation. For example, an algorithm that incorporates priorities increasing with time could be used.

2.5 Physical Interaction

In order to experiment with tanker based recharging in a real world setting, a means of exchanging energy between tanker and a charging station, as well as tanker and worker robots must be constructed. Below are the details of this work in progress. In particular, designs are outlined for the components involved: charging station, tanker, and worker robots.

The goal is to have a tanker robot that has a single physical interface for giving and receiving electrical energy. This way, the coupling of tanker and charging station is similar to the coupling of tanker and worker robot. Further, care is taken to make this coupling as easy as possible.

2.5.1 Charging station

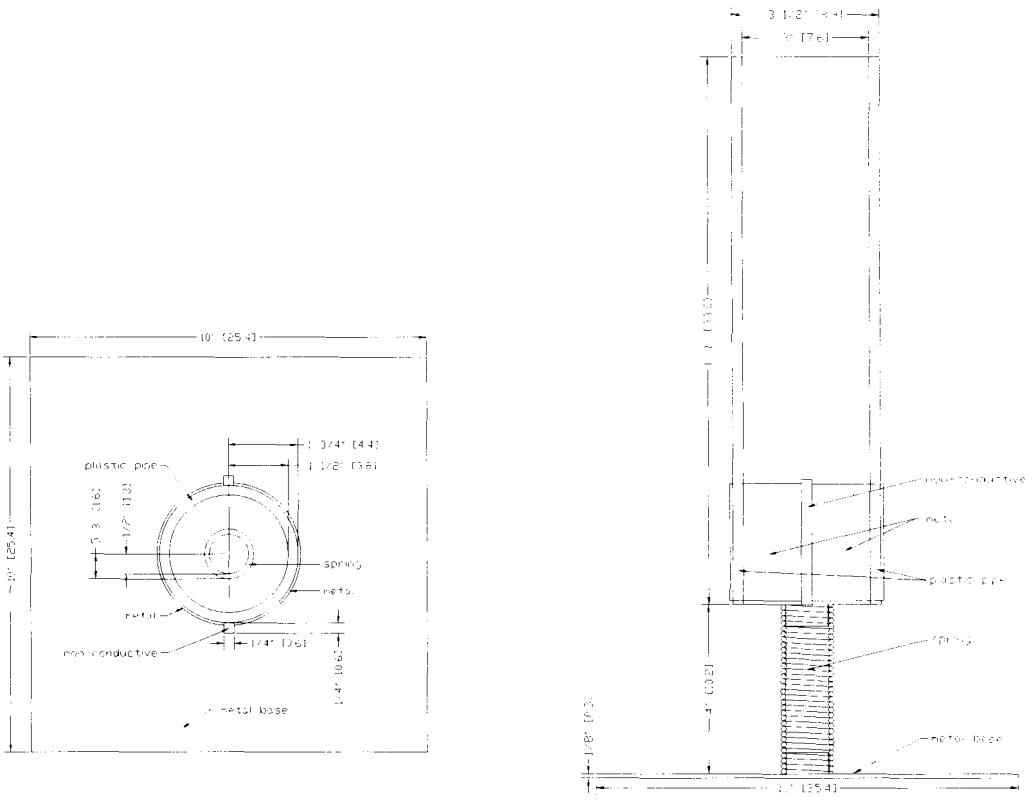
Our charging method uses a custom built charging station, and a Pioneer 3-DX outfitted with a SICK LMS laser range finder and ActivMedia 2 degree-of-freedom grippers. Figure 2.7 shows a schematic of the proposed charging station. The station consists of a cylinder mounted vertically with two metal contact plates mounted along the outside. These metal contacts are to be gripped by a tanker (much like a puck would) in order to complete the charging circuit. Two physical barriers mounted vertically prevent the tanker from short circuiting the charging station in the event that one gripper lands directly on both charging plates.

The cylinder is mounted on a spring to allow for tolerance when approached by a tanker. It is also constructed on a smooth base that allows the entire station to slide, giving additional tolerance. The cylinder is tall enough to allow for our tanker to use its laser range finder to identify the shape, and has a coloured 'hat' to aid in navigation.

2.5.2 Tanker

Figure 2.8 shows a Pioneer 3-DX about to connect to a charging station. The tanker is outfitted with 'mittens': metal contacts mounted to the grippers which physically touch the contacts on the charging station. With this approach, the act of interfacing with the charging station turns into a problem similar to locating and moving a puck. Wawerla et al. [42] describe an algorithm for achieving this.

Since our robots run on direct current, polarity needs to be observed. It is not possible to know the polarity before the robot connects, therefore circuitry is added to detect and rectify the polarity when a connection is established.



(a) Top (b) Side

Figure 2.7: Charging station schematic.

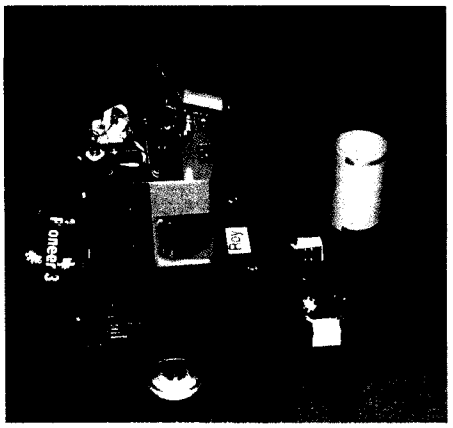


Figure 2.8: Tanker approaching a charging station. Printed by permission of Richard Vaughan.

2.5.3 Worker

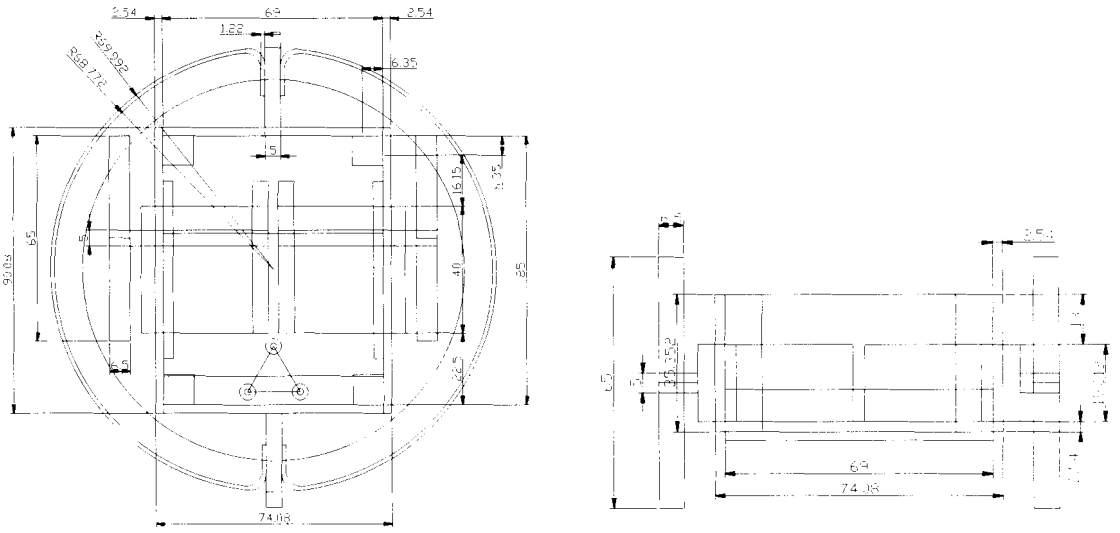
Work is in progress on construction of worker robots called 'chatterboxes.' Figure 2.9 shows the chatterbox design, which will be puck shaped to allow for interfacing with the tanker robot. The chatterboxes will have contacts along the outside of their perimeter to mimic the shape of a charging station. Once located and connected, the tanker will be able to give energy to a worker.

2.6 Conclusion

A novel means of recharging robot teams is presented. The use of an energy distributing tanker robot in a system of autonomous worker robots as a solution for achieving long term multi-robot systems is demonstrated. A tanker robot is constructed in simulation and demonstrated to be adequate at performing the prescribed task. We describe both a general tanker architecture suitable for adaptation given some target environment as well as design details used in achieving the tanker used in our experiments.

Several design techniques are implemented and evaluated for suitability and performance. Metrics are shown that demonstrate the utility of each technique, as well as a discussion about the intended and emergent behaviour.

A design is presented for implementing tanker based recharging on real robots. A tanker capable of using the same physical interface to recharge itself and give charge to a worker is shown. A charging station and worker robot are described.



(a) Top (b) Front

Figure 2.9: Chatterbox schematic.

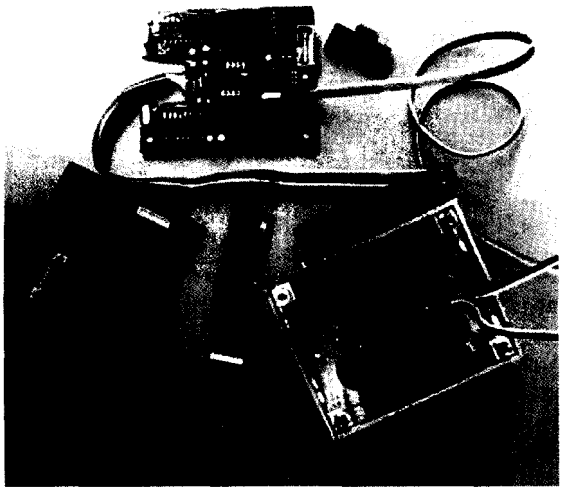


Figure 2.10: A disassembled chatterbox. Printed by permission of Barry Shell.

Chapter 3

Rendezvous

The work elaborated in this chapter is based on the work by Pawel Zebrowski, Yaroslav Litus, and Richard Vaughan [44], presented at the Fourth Canadian Conference on Computer and Robot Vision (CRV 2007), Montreal, QC., May 28-30, 2007.

3.1 Introduction

In order to achieve tanker based recharging, the tanker and worker robots need to meet. In this chapter we examine the problem of finding the most energy efficient meeting location given the locations and movement costs of all robots. Two methods, 'global' and 'local' are described and their properties compared. Both methods are tested in simulation. The experimental setup which is used henceforth is discussed.

3.2 Rendezvous as docking station recharging

The rendezvous problem can be thought to describe the best case energy usage estimate should all workers be required to dock at a single charging station. Consider n robots in some environment and some unique optimally energy efficient rendezvous location p^*. If a charging station is located at p^*, then the cost of a rendezvous achieved at this point will be the minimum total cost required if each robot was to return to the charging station and charge itself. This allows us to compare tanker based recharging results to traditional base station recharging energy usage. See Section 6.2 for a comparison of tanker based recharging versus docking station recharging.

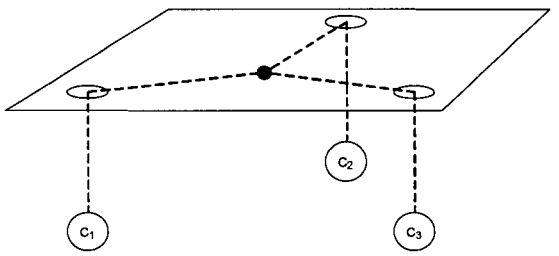


Figure 3.1: A mechanical interpretation of the rendezvous problem.

3.3 Problem characterization

Assume n robots are located at positions r_i, $i = 1 \ldots n$. 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:

$$p^* = \arg\min_p \sum_{i=1}^{n} c_i||p - r_i|| \tag{3.1}$$

Problem (3.1) is well known in optimization where it belongs to the family of *facility location* problems. Historically, it 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 [14] for the case $n = 3$ and [16] for the general case). Effective numerical algorithms exist [31]. Interestingly, it is also possible to describe the solution using a mechanical interpretation as a system of idealized strings, pulleys, and weights [29]. Figure 3.1 shows an example of such a system in which weight c_i represent the locomotion cost of robot R_i and the holes (pulleys) represent robot location. When this system is allowed to reach equilibrium, the knot will come to rest at the optimal rendezvous point p^*.

We will now briefly state the properties of the solution point. First, p^* cannot be located outside the convex hull formed by r_i. Second, if points r_i are not collinear (not lying upon a straight line), the goal function in (3.1) is strictly convex, which ensures the uniqueness of p^* if $n \geq 3$. Finally, it is possible to prove the following theorem [16]:

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

$$||\sum_{j=1}^{n} c_j \frac{r_j - r_i}{||r_j - r_i||}|| > ||r_i||, i \neq j \tag{3.2}$$

then $p^* \neq r_i$ for any i and $\sum_{i=1}^{n} c_i \frac{p^* - r_i}{||p^* - r_i||} = 0$ (the floating case). If (3.2) does not hold for some r_i then $p^* = r_i$ (the absorbed case).

Intuitively, in the floating case all robots meet at some point which is not the starting point of any robot: all robots move. In the absorbed case, one robot stays still and the others drive to it.

If r_i are collinear, then there may exist more than one point where minimum energy cost is incurred and this method may fail. This is not a significant practical problem for two reasons. First, in most real world situations, the robots are unlikely to be perfectly aligned (with the exception of 1-degree of freedom robots such as trains). Second, even in the collinear case the 'local dynamic' method presented below converges to a unique instance of the possible minima.

3.4 Solution

3.4.1 Global method

Since in our setting, each robot has complete information about other robot locations, the straightforward way to rendezvous is for each robot to find an approximation of p^* using a numerical method and then move towards it. Alternatively, if communication is possible, one robot can calculate $q^* \simeq p^*$ and broadcast q^* to the other robots. We will call this approach the *global* method because it computes a single point in global world coordinates common to all robots. The global method provides a complete solution to the problem provided the complete traversal costs are computable in advance, i.e. a complete map of obstacles is available in advance, and robot travel costs do not change. Any changes to travel costs that are detected during run-time will require a complete recalculation to ensure the best solution. In the experiments below, we will refer to the global method using only the static initial conditions as the *global static* method. If the global method is recalculated to take into account new information, we call this the *global dynamic* method.

In our experiments, we use the Nelder-Mead minimization method [25], which is a numerical method for minimizing an objective function in n-dimensional space. The method uses a simplex: a polytope of $n+1$ vertices, to continually improve its estimate of the minimum points of the objective function. This is done until the simplex is smaller than the desired approximation precision.

In the next section we propose a fast local heuristic method which allows robots to move towards p^* without ever calculating its location. An advantage of this method is that, in its dynamic form it naturally adapts to run-time changes in robot travel costs.

3.4.2 Local method

In this heuristic *local* method, each robot's physical trajectory approximates a gradient descent towards point p^* on the total movement cost function landscape. Since in the general case the gradient of the cost function is *not* pointing directly at p^*, the gradient must be reevaluated at suitable intervals. Each robot moves in the direction of the local gradient, which is periodically recalculated, until all robots arrive approximately at the rendezvous point. At no point is the complete landscape or a complete trajectory computed.

This method can be considered an example of the "information surfing" technique described by Bourgault et al. in [6]. Our contribution is the design of novel algorithms that approximate the minimal global energy cost objective function, and the empirical evaluation that suggests this is a practical solution to the rendezvous problem.

First, we introduce the function which returns a unit length vector in the direction from point a to point b:

$$\vec{d}(a,b) = \frac{b-a}{||b-a||} \tag{3.3}$$

We present two versions of this algorithm. Each robot runs the same algorithm asynchronously in parallel. The *local static* method uses only the initial robot locations and fixed movement costs. The *local dynamic* method assumes that robots periodically broadcast their current position estimate as they drive; the algorithm uses the latest position estimate for each robot. The two methods have different costs and benefits, which we describe below.

Algorithm 1: local static

1. Update current location of self, x.
2. Calculate $\vec{D} = \sum_{i|x \neq r_i} c_i \vec{d}(x, r_i)$.
3. If $x = r_i$ for some i, set $c = c_i$, otherwise set $c = 0$.
4. If $||\vec{D}|| < c$ then stop. Otherwise proceed in the direction $\vec{D}$.
5. Go to step 1.

Algorithm 2: local dynamic

Let self be the robot originally located at r_j

1. Update current location and movement cost of self, r_j, c_j. If information about other robots was received, update it as well.
2. Let $A = \{i, \text{where} ||r_i - r_j|| \leq \epsilon\}$, meaning the set of robots which are closer to r_j than some threshold ϵ. Note that $j \in A$, thus A has at least one element.
3. Calculate $\vec{D}_j = \sum_{i \notin A} c_i \vec{d}(x, r_i)$.
4. Set $c = \sum_{i \in A} c_i$.
5. If $||\vec{D}_j|| < c$ then stop. Otherwise proceed in the direction $\vec{D}_j$.
6. Broadcast own position and movement cost.
7. Go to step 1.

Both versions of the local method converge to the optimal meeting point (or any one of the optimal points in the collinear case).

The local approach has important benefits in comparison with the global methods. First, if the convergence trajectories performed using the local method are only slightly longer than the optimal straight paths to p^* (as shown empirically below), then the robots may meet a little sooner since they do not need to wait until the calculation of p^* is over to start moving. Second, if conditions change during the progress of convergence, the local method will incorporate changes instantly, adapting to the new information without the need for a computationally expensive recalculation of the meeting point. Conditions may change for several reasons, for example:

1. a robot may need to quit the rendezvous routine or a new robots may enter,
2. a robot may pick up additional load which will make it more costly to move (or the inverse), or
3. a robot may need to depart from its trajectory towards the meeting point because of environmental obstacles. In the extreme case the meeting point calculated using the original locations of the robots may end up *inside* a newly-discovered obstacle, and a replacement meeting place must be found.

Any of these events could cause the optimal meeting place to move significantly. This means that, for the global methods, a complete solution must be computed from scratch. In the last example, a robot may encounter new pieces of an obstacle it is navigating around, causing a stream of recalculations that may produce useless meeting points that turn out to be inside yet more obstacles. In contrast, the local methods quickly compute only the direction to move *right now*, that is towards the current best meeting place.

3.5 Experiments

A set of experiments is performed to demonstrate and compare the proposed global and local methods. For further comparison, a naïve method in which the robots meet at their center of mass is also tested. Each of these three methods is tested in its static and dynamic variants, for a total of six experiments.

3.5.1 World and simplifying assumptions

Experiments are performed in simulation using the Player/Stage robot control and simulation system [13]. The world is an empty circular arena 40 meters in diameter, containing ten mobile robots modeled after the ActivMedia Pioneer 3-DX with SICK LMS laser range finders. Each robot is assigned an individual weight c_i which describes its energy consumption per unit distance traveled. Total energy used as robot i moves from a to b is assumed to be $c_i||a - b||$.

During the experiment, robots are not visible to each other nor can they collide with each other. This unrealistic model is chosen in order to eliminate the effects of inter-robot spatial interference in this study. Spatial interference is a significant issue in multi-robot systems, particularly when the robots *must* operate in the same region, and rendezvous is the extreme case. Spatial interference is the critical limiting factor in how closely robots can approach the ideal rendezvous point, and will strongly determine the design of stopping conditions for any rendezvous algorithm. We recognize the importance of this topic [48], but believe it can usefully be ignored here to examine pure rendezvous performance, where our robots are treated as points that do not interfere with each other. Further, robots are assumed to have perfect localization, again to avoid influencing results based on the details of any one localization technique. The impact of localization error on the stability of our methods is an interesting area for future study.

3.5.2 Task

Given some initial arrangement of robots in the environment, the task is to meet at the unique location that minimizes the total system energy used on locomotion. Robots are deemed to have successfully met if for any two robots r_m and r_n the distance between them is less than some threshold s. In these experiments s is 1 meter.

The metric used to evaluate the performance of each method is the total system energy E used to achieve the rendezvous, where

$$E = \sum_{i=1}^{n} c_i d_i \tag{3.4}$$

and d_i is the length of the trajectory of robot i.

3.5.3 Robot controller

The experiment depends on two controllers: a robot controller and a processing controller. Communication between controllers is implemented with UDP datagrams. The robot controller is responsible for controlling robot movement within the environment. It reports the robot's current position to the processing controller, waits for the controller to prescribe the next move, then moves the robot to this location while avoiding obstacles. Obstacle avoidance uses the standard Player implementation of Borenstein's Vector Field Histogram method [5].

3.5.4 Processing controller

A single centralized processing controller is used to offload computation from the individual low-level robot controllers. It tracks each robot's position and computes each robot's next move using the chosen method. In static methods only the robot's original starting positions are considered. In dynamic methods only the most recently received robot positions are considered.

Global rendezvous method

This controller implements the algorithm described in 3.4.1. Given the location of each robot r_i it is possible to construct a cost function for meeting at any location p in the environment. It is then possible to find the minimum of this function, as described by Equation 3.1. Our method achieves this using the Nelder-Mead algorithm [25]. Nelder-Mead returns an approximation q^* to the minimum cost location p^*. The processing controller then prescribes this as the location each

robot should go to. In the dynamic variant q^* is recomputed periodically using the latest robot positions. In the static variant, q^* is recomputed using only the initial positions.

Local rendezvous method

The local method works by computing the conjugate gradient and prescribing the direction each robot should move in based on this gradient. In both static and dynamic variants, the local gradient at the robot's current position is recomputed periodically. The static variant implements Algorithm 1 above. The dynamic variant implements Algorithm 2 above.

Center-of-mass method

This method calculates the center of mass given the arrangements of robots. The center of mass is simply the weighted vector sum of the robot positions. This point is then prescribed as the meeting place for each robot to move to. In the static variant, the rendezvous point is calculated exactly once from the initial robot positions. In the dynamic variant, the meeting point is recalculated periodically using the latest robot positions.

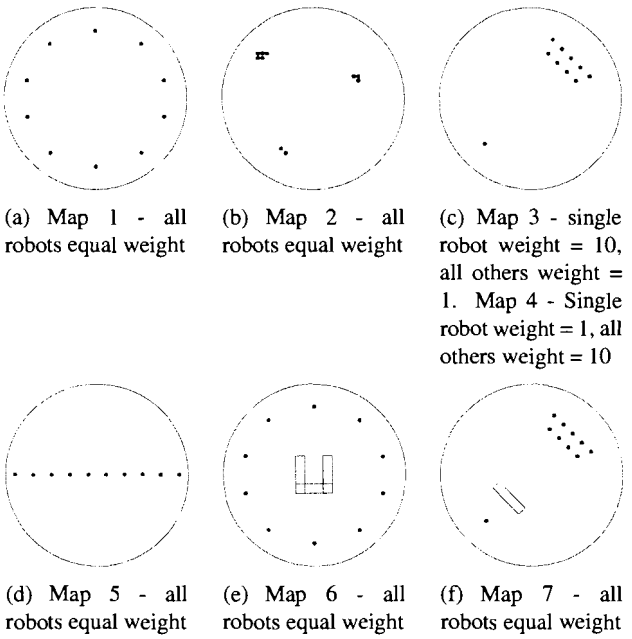
3.5.5 Experimental consistency

Each experiment uses an identical low-level robot controller. Only the high-level method for selecting the next robot goal position is changed between trials. To avoid any artifacts introduced by the presence or absence of communication overhead, the static experiments actually perform communication but simply discard the received messages.

3.5.6 Procedure

Each of the six experiments is performed with seven distinct initial configurations illustrated in Figure 3.2. A configuration consists of a map that specifies each robot's starting position and orientation, and the position of any environmental obstacles. A configuration also specifies each robot's locomotion cost weight. Configurations 3 and 4 use the same starting locations but have different locomotion costs. Maps 6 and 7 incorporate an obstacle to demonstrate each method's ability to cope with obstacles.

20 trials are performed on each configuration/method pair, for a total of $6 \times 7 \times 20 = 840$ trials. The total energy used in the trial is recorded and used to compute a mean and standard deviation.



(a) Map 1 - all robots equal weight

(b) Map 2 - all robots equal weight

(c) Map 3 - single robot weight = 10, all others weight = 1. Map 4 - Single robot weight = 1, all others weight = 10

(d) Map 5 - all robots equal weight

(e) Map 6 - all robots equal weight

(f) Map 7 - all robots equal weight

Figure 3.2: Experiment initial conditions.

The path taken by each robot is also recorded. There is a time limit on experiments (ten times the typical trial length) in case a trial fails to result in a rendezvous.

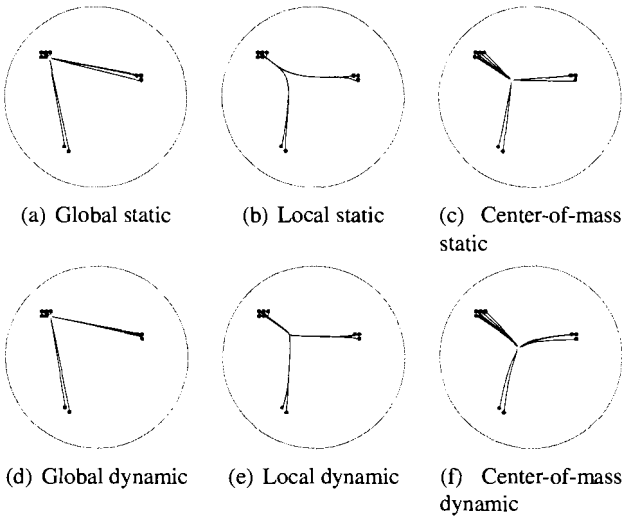
3.6 Results

3.6.1 Paths traversed

Figure 3.3 shows some example paths taken during the experiments. In each case, the global methods result in robots heading directly for the meeting location (Figure 3.3(a) and 3.3(d)) while the local methods tend to take a curved path (Figure 3.3(b) and 3.3(e)) as predicted in Section 3.4.2.

In the absence of obstacles (and spatial interference), all methods perform similarly regardless of whether the static or dynamic variant is used. The only exception is the local static method with collinear initial conditions, which fails to rendezvous as mentioned in Section 3.3.

In an environment with obstacles, all methods manage to achieve a meeting regardless of whether the static or dynamic variant is used.



(a) Global static (b) Local static (c) Center-of-mass static

(d) Global dynamic (e) Local dynamic (f) Center-of-mass dynamic

Figure 3.3: Typical paths taken on map 2.

Table 3.1: Mean total energy used in maps without obstacles. All standard deviations $< 5\%$.

	Dynamic			Static		
Map	C-o-mass	Global	Local	C-o-mass	Global	Local
1	149.93	149.99	151.56	149.92	149.92	152.79
2	119.78	105.21	115.21	113.79	105.25	119.00
3	154.32	77.56	80.49	86.12	77.28	81.48
4	923.74	477.89	503.18	531.24	476.97	530.11
5	95.58	97.36	97.98	95.54	95.51	Failed

3.6.2 Energy used

Table 3.1 shows the mean total energy used for each method/map combination. In each case without obstacles (Map 1-5) the global methods use less energy than the local methods. In cases with obstacles, this is not true. In cases where the center of mass coincides with the minimum energy cost location (as expected e.g. in Map 1) the center of mass method performs well, otherwise it performs poorly compared to the other methods.

Table 3.2: Mean total energy used in maps with obstacles. All standard deviations $< 6\%$.

	Dynamic			Static		
Map	C-o-mass	Global	Local	C-o-mass	Global	Local
6	293.58	288.56	256.75	284.46	283.23	256.25
7	1150.64	533.58	549.57	579.20	5.24.22	600.02

3.7 Discussion

3.7.1 Paths traversed

Figure 3.3 shows typical paths taken for each method using Map 2 (Figure 3.2(b)). Figure 3.3(a) and 3.3(d) show the robots meeting at approximately the optimal location. Figure 3.3(b) shows the robots also meeting at the optimal location, but the path traversed is an arc. This curve is produced by the robot following the local conjugate gradient. The local methods almost always result in curved paths that are slightly longer than those produced by the global methods.

In practice the cost of the longer path is traded off against the cost of the increased computation that may be required by the global method. It is unknown whether a robot that sits idle while computing the optimal meeting location then heading directly for it will consume less energy than a robot that begins moving immediately, but follows a longer path. This is a possible area of future research.

Figure 3.3(c) shows robots meeting at the center of mass. Notice that in this case, the center of mass does not coincide with the optimal meeting point and the energy usage is higher than the optimal methods. In the dynamic variant (Figure 3.3(f)) this is more pronounced, as the meeting point moves further away from the optimal location.

3.7.2 Static versus dynamic

The effect of dynamic updating can be seen clearly by comparing Figure 3.3(b) and 3.3(e). Updating position information results in robots 'sticking' together when they meet (Figure 3.3(e)). This is a desired result, since two robots r and s at the same location are equivalent to one robot of weight $r + s$. Further, each robot in such a group should conclude the same shortest path to the meeting point. In the comparison of 3.3(b) and 3.3(e) dynamic updating achieves this by shortening the curved path caused by the local method.

Dynamically updated robot positions can act as a benefit or hinderance. In our trials, it results in both improved and degraded cases. The degradation tends to be minimal while improvements tend

to be significant. One particular example of this is the collinear case using the local method, where without dynamic updates the robots never meet. In our trials, the naïve center of mass method never benefits from dynamic updates.

In an environment with obstacles, benefits of dynamically updating position information depend on the details of the environment. For example, in cases where an obstacle results in a shift of the optimal location, such an update is an asset. In cases where one obstacle results in a shift, then another results in a shift back, dynamic updating results in cyclic behaviour (and therefore higher energy usage). This is due to updates not yielding any real insight into the path that will need to be traversed, and is the expected result. However, one notable benefit of dynamic updating occurs when the meeting location chosen lies in an unreachable space (such as inside an obstacle). With the static methods, robots will perpetually attempt to reach an unreachable space. With the dynamic method, there is a chance that the optimal meeting location will shift away from the unreachable location. An algorithm which considers obstacles when computing the meeting location would be more suitable in this case.

3.7.3 Local versus global

Both methods achieve their intended goal: to identify a meeting location that can be reached at a minimum total cost. However, the methods exhibit different characteristics which make each more suitable in certain applications.

The global method performs well, but is computationally expensive. For a large number of robots computation may become prohibitive. This is especially true in the dynamic case. Every position update results in a recomputation of the optimal meeting location. On the other hand, the local method is computationally cheap, but does not yield a location but rather a direction to head in. Further, the path traversed tends to be slightly longer than that of the global method. The suitability of each method depends on the application. For small teams of robots, the global method should perform well, even the dynamic variant. For larger teams, the local method will avoid lengthy computation before moving. If a meeting is desired in a minimum amount of time, it may be the case that computation time with the global method exceeds the extra travel time resulting from the local method. In such a case, the local method would result in a rendezvous before the global method, despite the longer path.

3.8 Conclusion

We state the natural *robot rendezvous problem*, where multiple robots must meet at some location, chosen to optimize some utility function over all robots, and identify this as the single-facility location problem. We propose that this is the initial step in achieving meetings required for tanker based recharging and argue that the energy required to achieve a rendezvous is the minimum energy required to achieve traditional docking station recharging for all worker robots.

We implement a standard numerical solution and empirically evaluate its performance in a variety of scenarios. This data is then used to evaluate the performance of a novel behavioural heuristic method, which is shown to (i) produce global rendezvous; and (ii) incur travel costs only slightly greater than the global optimization method. The novelty here is in designing Algorithms 1 and 2 that create a local gradient that approximates the direction of the global optimum. These algorithms are very fast, with runtime growth linear with population size, and small constant per-robot cost. For some applications this may be preferred to an iterative numerical approximation technique with unknown runtime. Further, by iterating the local algorithm as the robot drives, we can naturally incorporate new information about robot locations, and thus cope with obstacles, robot locomotion failures, etc. without invalidating previous computation.

Some limitations of this work so far are in our assumptions of good global localization and reliable communication. The assumption of no robot collisions is not important, as this can be considered equivalent to the presence of obstacles, with which this method is shown to cope.

Consideration of the local heuristic method suggests that in the presence of unbiased error in global localization, these methods will cause the robots to converge until their mutual distances are within a small factor of the mean localization error. Given that global localization methods that give accuracy to less than a robot's sensor range are in everyday use, our method should be practical.

As for unreliable communications, we believe that occasional dropped messages will cause only small changes in individual robot behaviour, leading to graceful degradation in overall performance. On the other hand, a robot that loses communications completely during a run may be irrecoverably lost if the other robots encounter obstacles that cause the optimal rendezvous point to change.

Chapter 4

Frugal Feeding

The work elaborated in this chapter is based on the work by Yaroslav Litus, Richard Vaughan, and Pawel Zebrowski [20], presented at the IEEE International Conference on Robotics and Automation (ICRA 2007), Rome, Italy, April 10-14, 2007.

4.1 Introduction

This chapter introduces the 'frugal feeding' problem by relaxing the requirement for all robots to meet at one location. In order to achieve tanker based recharging, only the tanker needs to visit each worker robot. Further, since energy is a precious resource, doing so in an energy efficient manner is desirable. We seek to minimize the total amount of fuel spent driving robots to refueling rendezvous, so that the fuel available for useful work is maximized.

We present several *partial* methods to aid in solving the frugal feeding problem in practice. First, we outline a restricted locations case, where potential solutions are picked from a discrete set of candidate solutions. Following, we present the Nelder-Mead continuous numerical solution.

4.2 Problem definition

Given a set of original locations of worker robots and tanker robot, find a set of meeting points such that the tanker meets every worker in a way that minimizes the total energy spent on locomotion. By analogy to a mother animal attending her offspring we have named this problem the frugal feeding problem. It can be stated formally as follows:

Definition 1 (Frugal feeding problem). *Given tanker location $p_0 \in \mathbb{R}^d$, worker locations $r_i \in \mathbb{R}^d, i = 1..k$ and locomotion cost functions $C_i : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, i = 0..k$, find*

$$\min_{\pi, p_1, p_2, \ldots p_k} \sum_{i=1}^{k} \left(C_0(p_{i-1}, p_i) + C_{\pi(i)}(r_{\pi(i)}, p_i) \right) \tag{4.1}$$

Here $C_0(x, y)$ gives the cost of tanker relocation from x to y, $C_i(x, y)$ gives the corresponding cost for worker i, and $\pi : \{1..k\} \to \{1..k\}$ permutes the workers according to the order in which they are attended by the tanker.

Definition 1 could be amended to require the tanker to return to its original location after attending all workers, perhaps to refuel itself. This modification does not change the following analysis.

The problem has two components. One is combinatorial: finding the order in which robots should be attended; the other is analytical: finding the meeting points for the given order.

We will denote the solution points as p_i^* and corresponding permutation as π^*. The possibility of several robots being attended in one place is permitted, and captured by the possible coincidence of some meeting points p_i^*.

The Fermat-Torricelli problem (also called the Steiner-Weber problem) asks for the unique point x minimizing the sum of the distances to arbitrarily given points $x_1, \ldots, x_n$ in Euclidean d-dimensional space. Elaborating on the conventions in the Fermat-Torricelli problem literature [16], we name the location of solution points as follows.

Definition 2 (Special solution points). *If $p_i^* = r_j$ for some j, we call p_i^* a* worker absorbed *point. In this case worker i should either remain still and wait for the tanker to come to it, or move to the location of another worker. If $p_i^* = p_0$, we call p_i^* a* tanker absorbed *point. In this case the tanker should not move. Instead, worker i should move to the tanker's original location. Otherwise, p_i^* is not coincident with the starting point of any robot, and we call it a* floating *point. We show below that all of these are plausible contingencies.*

An instance of the problem is shown schematically in Figure 4.1, along with candidate solutions of each possible type.

The problem definition does not commit to any particular locomotion cost function. Cost functions could be complex to account for the presence of obstacles or other heterogeneities of the environment or robots. Unfortunately the nature of the cost function can make the complete problem very difficult to solve. Here we use the straightforward cost function:

$$C_i(x, y) = w_i ||y - x|| \tag{4.2}$$

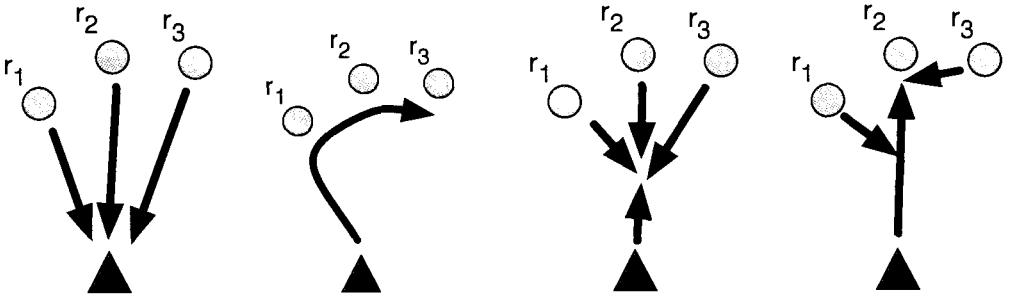


Figure 4.1: Schematic of the frugal feeding problem. Tanker robot (triangle) must rendezvous with worker robots (circles labeled r_1, r_2, r_3) in order. Four types of solution are possible: tanker absorbed, worker absorbed, floating, and a hybrid of these.

So the cost of relocating a robot is simply the weighted Euclidean distance between the origin and destination. The weight models the energetic cost of moving the robot some unit distance: a large, heavy robot would have a higher weight than a small, lightweight robot.

4.2.1 Analysis

The first observation to make is that solution points p_i^* can not lie outside the convex hull of $\{p_0, r_1, r_2, ...r_k\}$. This can be seen by considering a candidate meeting point outside the hull: replacing the candidate point with the closest point on the convex hull will unambiguously decrease the value of the goal function.

Special cases

Under certain conditions we can quickly find solutions without solving the general problem. We first show sufficient conditions for the worker absorbed case, i.e. where the meeting point for a worker is at that worker's starting location:

Lemma 1. *If $w_i \geq 2w_0$ then $p_i^* = r_i$. If $w_i > 2w_0$, then $p_i^* = r_i$ is the unique solution for p_i.*

Proof. The components of the objective function in (4.1) which depend on p_i are

$$g_i(p_i) = w_0||p_i - p_{i-1}|| + w_0||p_{i+1} - p_i|| + w_i||p_i - r_i|| \tag{4.3}$$

We can consider g_i in isolation. Point $p_i^* = \text{argmin } g_i(p_i)$ is the solution to the weighted Fermat-Torricelli problem formed by points p_{i-1}, p_{i+1}, r_i with corresponding weights w_0, w_0, w_i. As shown by Kupitz [16] there is a sufficient condition for the solution to be exactly at point r_i (the *absorbed case*) if points are non-collinear:

$$||w_0(\vec{u}(r_i, p_{i+1}) + \vec{u}(r_i, p_{i-1}))|| \leq w_i, \tag{4.4}$$

where $\vec{u}(x, y) = (y - x)/||y - x||$ gives the unit vector in direction from point x to point y. Now we find the upper bound of the left side of (4.4) to solve for the condition, sufficient for any pair of p_{i-1}, p_{i+1} which satisfies the non-collinearity condition. The left side can achieve value $2w_0$ only when $p_{i-1} = p_{i+1}$, thus for the non-collinear case the condition is $w_i \geq 2w_0$. We complete the proof by considering the collinear case. [4, p. 251] gives the sufficient condition for the collinear case of the Fermat-Torricelli problem. Point p_i^* is unique, if $W^- < W/2$ and $W^+ < W/2$, where W^- is the total weight of the points to the one side of p_i^*, W^+ is the total weight of the points to the other side, and W is the weight of the minimum point itself. If $W^- = W^+ = W/2$, then p_i^* is one of the many minimum points which comprise a closed segment and are defined by these equalities. Interpreting these conditions for our case we obtain the desired result. □

We now present sufficient conditions for the case where the tanker stands still and all workers are charged at the tanker location.

Lemma 2. *If $\sum_{i=1}^{k} w_i \leq w_0$ then $p_i^* = p_0$ for all i. If the inequality is strict, then this solution is unique.*

Proof. Let $C_a = \sum_{i=1}^{k} w_i||r_i - p_0||$ be the cost of a complete tanker absorbed solution, and $C(p_1, p_2, ...p_k) = \sum_{i=1}^{k} (w_i||r_i - p_i|| + w_0||p_i - p_{i-1}||)$ denote the cost of an alternative solution. We will show that for all values of $p_i, i = 1..k$ inequality $\Delta C = C_a - C(p_1, ...p_k) \leq 0$ holds.

$$\begin{aligned}
C_a - C(p_1, ...p_k) &= \sum_{i=1}^{k} w_i(||r_i - p_0|| - ||r_i - p_i||) - w_0 \sum_{i=1}^{k} ||p_i - p_{i-1}|| \\
&\leq \sum_{i=1}^{k} w_i||p_i - p_0|| - w_0 \sum_{i=1}^{k} ||p_i - p_{i-1}|| \\
&\quad (\text{let } ||p_j - p_0|| = \max_i\{||p_i - p_0||\}) \\
&\leq \sum_{i=1}^{k} w_i||p_j - p_0|| - w_0 \sum_{i=1}^{j} ||p_i - p_{i-1}|| \\
&\leq \sum_{i=1}^{k} w_i||p_j - p_0|| - w_0||p_j - p_0|| \\
&\leq 0
\end{aligned}$$

□

4.2.2 Complexity

Lemma 1 can be used to show that the frugal feeding problem is NP-hard because of its combinatorial component which finds the best order in which robots are visited. To do this we reduce the geometric salesman problem to the frugal feeding problem.

Theorem 2. *GEOMETRIC TRAVELING SALESMAN $\leq$ FRUGAL FEEDING*

Proof. Given an instance of the geometric traveling salesman problem (set of points $a_i, i = 0..n$ where a_0 is the original location of the salesman) we create the following frugal feeding problem. Assign $p_0 = a_0; w_0 = 1; r_i = a_i, w_i > 2$ for $i = 1..n$. According to Lemma 1, for any considered order of meetings (given by permutation π in (4.1)) optimal locations $p_i = r_{\pi(i)}$. This means that the set of solution points coincides with the set of robot locations, $\{p_i^*\} = \{r_i\}$. Thus the solution for the frugal feeding problem will be the minimum traveling path between all points r_i starting at location p_0 which is exactly the solution to the original problem. □

Theorem 2 proves that the combinatorial component of the frugal feeding problem is not easier than the geometric traveling salesman problem. Given that the latter is NP-hard [28] and no general scalable solution is available, we consider a search for efficient domain-specific heuristics to be more promising than attempts to find an exact solution. An interesting opportunity for future work is to find an algorithm which will solve the combinatorial component (meeting order) and analytical component (rendezvous locations) simultaneously. Literature on the traveling salesman problem is abundant, so depending on the particular initial conditions, a suitable heuristic could be selected. For example, if the number of robots is large and time is critical, heuristics presented in [30] could be used. A cheap and simple alternative ordering could be a nearest neighbour series, starting at p_0.

4.3 Restricted locations case

Assume that the locations where the tanker could attend to a worker are not arbitrary, but are limited to a fixed set of places. This models, for example, a list of coffee shops where a consultant can meet clients, or a list of air strips where a robot reconnaissance airplane could meet a ground-based tanker truck. The finite set of meeting places could also be a division of continuous space into a regular grid.

In this discrete version, the optimization problem (4.1) is extended with a constraint $p_i^* \in L$, where L is the set of possible meeting places. L could be a superset of the original robot locations

$\{r_i\} \cup \{p_0\}$. The naïve brute-force algorithm for solving the analytical part of the problem will take $O(|L|^k k)$ time ($|L|^k$ possible meeting point arrangements and $O(k)$ to calculate the cost value). However, it is possible to exploit the structure of the objective function and to show an algorithm with running time $O(|L|^2 k)$. This algorithm is based on the ideas of dynamic programming [3] and is presented in [20].

4.4 Continuous case

In the continuous version, the cost function in Equation 4.1 is altered in order to eliminate the combinatorial component of the frugal feeding problem. The cost function now becomes:

$$\min_{p_1,p_2,\ldots p_k} \sum_{i=1}^{k} \left(C_0(p_{i-1}, p_i) + C_i(r_i, p_i)\right) \tag{4.5}$$

It is possible to arrange this function in the form of a multidimensional minimization problem. In Chapter 5, we use the Nelder-Mead numerical minimization method to find points $q_1^*, q_2^*, ..., q_k^*$ which approximate the solution points $p_1^*, p_2^*, ..., p_k^*$ to (4.5). These points can then be prescribed to both tanker and worker robots as the meeting locations.

For a more complete description of numerical methods for the frugal feeding problem see Litus et al. in [20].

4.5 Conclusion

This chapter introduces and formally defines the frugal feeding problem. An analysis is performed and several special cases of the problem are identified. Finally, it is shown that this problem is NP-hard.

A discrete method, which produces optimal results in polynomial time, and the Nelder-Mead numerical method are described. In the next chapter, we present a perfectly scalable, distributed heuristic for achieving good quality multi-point rendezvous.

Chapter 5

Frugal Feeding Heuristic

The work elaborated in this chapter is based on the work by Yaroslav Litus, Pawel Zebrowski, and Richard Vaughan submitted to the IEEE Transactions on Robotics on June 11, 2007.

5.1 Introduction

This chapter presents a method whereby a simple controller, running in parallel on each robot, causes the robots to achieve good quality multi-point rendezvous thus approximating the solution to the analytical part of the frugal feeding problem. The method is perfectly scalable, in that both the effective computation time and time to task completion are linear in the population size.

It is assumed herein that the indexing of the workers corresponds to the partial order in which they are attended by the tanker. All robots know their own indices as well as the index of the next robot to be charged. Unlike the single-point rendezvous heuristic described in Chapter 3 and [44], here a robot needs to know only the locations of two other robots to calculate the direction of movement. We formally define a meeting as the event when robots come within distance s of each other.

Algorithm 1: frugal feeding heuristic

1. Update current location and movement cost of self, r_j, w_j. If information about the relevant robots is received, update it as well.

2. Check sufficient conditions described in Section 4.2.1. If satisfied, behave accordingly and go to step 6.

3. If $k = 1$ (only one worker in charging queue), then if $w_0 > w_1$ tanker stops, worker r_1 moves towards tanker. If $w_0 < w_1$, worker r_1 stops, tanker moves towards worker. If $w_0 = w_1$, tanker and worker may move towards each other, or one of them may stop and wait for the other. Go to step 6.

4. **Workers** ($j > 0$): If $j = k$, let $r_{k+1} = r_k$.

   If $j = 1$ (the next worker to be charged), set $\vec{D}_1 = w_0\vec{d}(r_1, r_0) + w_0\vec{d}(r_1, r_2)$.

   Otherwise, set $\vec{D}_j = w_0\vec{d}(r_j, r_{j-1}) + w_0\vec{d}(r_j, r_{j+1})$.

   Tanker ($j = 0$): Set $\vec{D}_0 = w_1\vec{d}(r_0, r_1) + w_0\vec{d}(r_0, r_2)$.

5. If $||\vec{D}_j|| < w_j$ then stop. Otherwise proceed in the direction $\vec{D}_j$.

6. Broadcast own position and movement cost. Once the worker robot is met and charged, tanker or charged worker should broadcast a corresponding message. Upon receipt, the robots decrease the robot counter k and own queue number j. The charged worker is not considered anymore in this instance of the problem.

7. Go to step 1.

5.2 Proof of correctness and run time bounds

The proof of correctness due to Yaroslav Litus and shown in Appendix A identifies the robots themselves as the current approximation to the solution points. Every robot tries to improve the quality of the solution by moving along the direction which decreases the corresponding part of the total cost function. If a robot is located at the minimizing point for the current configuration of robots, the robot stops. The proof shows the correctness of the frugal feeding heuristic by bounding the time it takes for robots to meet given robot sensor range s.

5.3 Experiments

A set of experiments is performed to demonstrate the frugal feeding heuristic and to empirically assess the efficiency of the paths it produces. Again, we consider only the analytical component of the frugal feeding problem: the order in which the tanker must visit each worker is fixed thereby eliminating the combinatorial component.

Further experiments are performed where the combinatorial component is solved with either brute force or a nearest neighbour algorithm. This is to demonstrate results for a complete solution to the frugal feeding problem. These example solutions are arbitrary and are intended as proof of concepts that at least some solution exists.

All experiments are performed with inter-robot collisions disabled where robots are invisible to each other and can occupy the same space. Experiments are then repeated with inter-robot collisions enabled. Non-collision experiments are performed in order to achieve results close to those predicted theoretically (without the influence of interference) and to validate the theory, while collision experiments are done in order to demonstrate algorithm suitability with interference present, and to gauge its effects.

All experiments are performed in a variety of environments, including ones with and without obstacles. Again, this is to demonstrate the frugal feeding heuristic method's ability to cope with obstacles and to gauge the effects.

Experimental set up is outlined in Section 3.5. Important differences are noted below.

5.3.1 Controller implementation

The experiment depends on two controllers: a robot controller and a processing controller. Even though the frugal feeding heuristic is intended to run in a distributed environment, we make use of a computation controller for experimental consistency. Communication between controllers is implemented with UDP datagrams.

Robot controller

This controller is responsible for the robots and their movements. The controller is principled on the subsumption architecture [7] which is used to prioritize tasks that a robot needs to perform. Since the tasks can all be assigned a static priority, this architecture works well.

The robot controller subsumption architecture consists of the following tasks:

1. communicate - update the central processor of the current location,
2. work - perform some task (the worker's main goal), and
3. seek way point - head for the way point prescribed by the tanker.

In our experiments, workers are assumed to have stationary tasks (such as digging a hole or taking measurements). This is done in order to have repeatable trials, since having workers move as part of their task would result in randomized starting locations.

Computation controller

This controller is responsible for aggregating robot data and prescribing the next way point for each robot. For this set of experiments, the only data considered are current robot location and locomotion cost. This controller performs the following tasks repeatedly:

1. check for new information from worker robots and update location data,
2. check if any meetings have been achieved, and update the list of meetings still required,
3. if no more meetings are required, terminate,
4. execute the chosen method on this data and store the result,
5. communicate this result (next way point) with each robot,
6. graph and log data and statistics.

For legacy reasons, in our implementation this computation controller is part of the tanker robot controller.

5.3.2 Implementation details

A note should be made about obstacle avoidance using the standard Player implementation of Borenstein's Vector Field Histogram method [5]. With visual inspection of trials, it can be seen that VFH does not always perform in an ideal way. In particular, should an obstacle cause a robot to turn and head away from a destination, VFH may not cause the robot to stop and turn around. Instead, the robot will head away from the destination for some time before changing course back towards it. This has an impact on energy consumed.

Another implementation detail deals with the following scenario. Consider a tanker R_0 with cost of movement C_0 and worker R_i with cost of movement C_i distance d apart. If $C_0 = C_i$, then where should the two robots meet? Meeting at any point which lies on the straight line between R_0 and R_i will result in the same total cost of movement. The problem arises when a numerical algorithm is asked to solve this problem. It will come up with a different point on this line every

time, resulting in erratic robot movements. For this reason, tanker cost of movement C_0 is changed to $C_0 + \epsilon$ for all experiments. This breaks the symmetry and causes the worker to do all traveling in the above scenario.

A more straightforward implementation detail is that, to avoid any artifacts from simulating a docking, we consider that robots have met when their mutual distance is less than a threshold of $s = 1$ meter. In a real robot implementation, we assume that another controller could take over to perform a sensor-based docking manoeuver.

Data recorded

Each trial records the following information:

- general trial information, such as time elapsed, total energy used, and number of computational cycles required,
- location of all robots, every second, which allows us to reconstruct the path taken by each robot, including temporal information,
- the goal point p_i' for each robot, every second. This allows us to visualize the information each robot has received which dictated the resulting path. It also allows us to visualize the results obtained from the processing controller, and
- elapsed computation time (real time) for every computation cycle. This allows us to examine computation time constants as well as rate of growth.

5.3.3 Frugal feeding task

Given some initial arrangement of robots in the environment, the task is for the tanker to meet with each worker robot exactly once, while minimizing the total system energy used on locomotion. For the sake of an accurate comparison, this task is further constrained such that the tanker must visit workers in a predetermined sequence (in order to keep the combinatorial portion of frugal feeding constant between experiments). This constraint is eliminated when performing experiments to demonstrate a complete solution for frugal feeding.

The tanker is deemed to have successfully met with a worker if the distance between it and the worker to be visited next is less than some threshold s. In these experiments s is 1 meter. It should be noted that coming within distance s of a worker out of order does not result in a meeting.

The metric used to evaluate the performance of each method is the total system energy E used during the experiment, where

$$E = \sum_{i=1}^{n} c_i d_i \tag{5.1}$$

and d_i is the length of the trajectory of robot R_i in a population of n robots.

Fixed order Nelder-Mead controller

This controller uses the Nelder-Mead numerical minimization algorithm [21] to find an approximation to the solution of the problem (4.5) for a predetermined visiting order π. The processing controller then prescribes the approximated solution $q_i^* \simeq p_i^*$ as the destination location r_i for robot R_i, $1 <= i <= n$ where n is the number of robots. The tanker R_0 is prescribed to visit each of q_i^* for $1 <= i <= n$ and wait at each location until the worker arrives.

In order to achieve robustness the solution is computed repeatedly with updated robot positions r_i. Each recomputation provides a new set of meeting locations q_i^*. This allows the system to recover in case a robot deviates from the chosen path because of an obstacle, interference, or navigation error. These deviations may make the original solution invalid presenting a new instance of the frugal feeding problem. Since it is difficult to determine if the deviation is significant enough to invalidate the present solution without comparing it with the solution for the current robot positions, recomputation is performed each time robot locations are updated.

Fixed order frugal feeding heuristic controller

Given the location of each robot R_i and a predetermined visiting order π, this controller implements the algorithm described in Section 5.1 to compute each robot's direction of travel. This direction is then used to determine a point p_i' which is given as the goal point for the VFH control program of robot R_i. Robot R_i then attempts to drive to p_i' while avoiding obstacles and results in R_i traveling in the direction originally intended by the frugal feeding heuristic. It should be noted that R_i will never reach p_i' as it is always a constant distance away from r_i. A similar technique is used to implement the stopping condition. Point p_i' is set to r_i and given to the VFH control program of robot R_i. This results in robot R_i stopping.

This computation is performed repeatedly with updated robot positions r_i. Every cycle, each newly generated goal location p_i' is communicated with robot R_i.

Brute force Nelder-Mead controller

This controller implementation constructs all possible worker visiting orders and applies the fixed order Nelder-Mead controller to each potential order. The ordering that achieves the lowest cost is used, and the controller proceeds as before with this order. The ordering is computed repeatedly along with the Nelder-Mead algorithm.

Nearest neighbour frugal feeding heuristic controller

This controller implementation constructs an ordering based on the nearest neighbour method. This ordering is then used as the ordering for the fixed order heuristic controller. The ordering is computed repeatedly along with the frugal feeding heuristic algorithm.

Distributed single point heuristic controller

This controller is based on the algorithm in Chapter 3. It is included for comparison with the other methods.

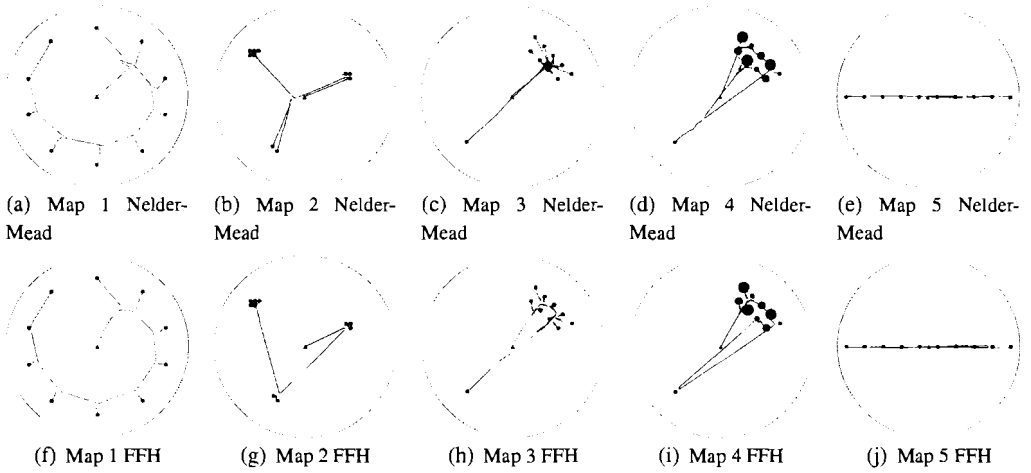
Procedure

All three controllers are used with nine distinct initial configurations, four of which contain obstacles. A configuration consists of the number of robots and a map that specifies each robot's starting position and orientation, along with any environmental obstacles. A configuration also specifies each robot's locomotion cost weight. Fixed order experiments are performed with ten worker robots, while variable order experiments are limited to five worker robots due to algorithm runtime restrictions.

All experiments are performed with collisions disabled, then repeated with collisions enabled. A minimum of 20 trials are performed on each configuration/method pair, for a total of more than 2000 trials. There is a time limit on experiments in order to prevent trials from continuing indefinitely. For example, if two robots have collided and are not able to move, all meetings may not be able to complete.

5.3.4 Fixed order results

Overall, both Nelder-Mead and the frugal feeding heuristic perform well in our fixed order experiments. Both methods achieve the goal of tanker visiting each worker, and both methods do so



(a) Map 1 Nelder-Mead (b) Map 2 Nelder-Mead (c) Map 3 Nelder-Mead (d) Map 4 Nelder-Mead (e) Map 5 Nelder-Mead

(f) Map 1 FFH (g) Map 2 FFH (h) Map 3 FFH (i) Map 4 FFH (j) Map 5 FFH

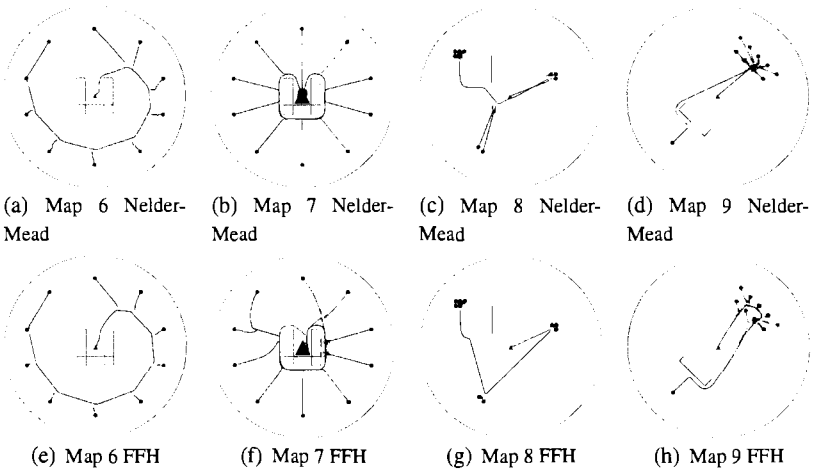
Figure 5.1: Typical paths taken for maps 1 through 5. Visiting order is fixed, collisions are disabled, and no obstacles are present.

Table 5.1: Mean and standard deviation of total energy used for rendezvous with no obstacles present and collisions disabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Nelder-Mead	117.08	4.55	74.40	6.30	65.04	1.43	103.36	3.68	54.28	0.75
FFH	105.00	0.17	61.89	0.29	60.89	0.30	82.53	0.11	52.09	0.07
Single point	150.60	0.26	123.27	0.31	66.60	0.19	131.56	0.98	93.70	0.44

using similar amounts of energy. Visual inspection of paths traversed shows robots moving in direct trajectories, agreeing on common meeting locations, and not participating in obvious sub-optimal behaviour. In our experiments, all trials completed successfully. See Appendix C for a complete list of results.

Nelder-Mead and the frugal feeding heuristic do not always dictate the same robot path. Figure 5.1(b) shows the path traversed using Nelder-Mead, while Figure 5.1(g) shows the same map using the frugal feeding heuristic. This difference is reflected in Table 5.1. A second notable difference can be seen in Figure 5.1(a) and 5.1(d). Notice the irregular paths taken by the workers. We conjecture this to be an artifact of using a numerical optimization method. Again, the effects are reflected with increased energy usage for Nelder-Mead in Table 5.1. Finally, an important difference which cannot be seen in the figures is that of timing. The Nelder-Mead method results in all robots learning their destination together and therefore traveling at once. The frugal feeding heuristic method produces



(a) Map 6 Nelder-Mead (b) Map 7 Nelder-Mead (c) Map 8 Nelder-Mead (d) Map 9 Nelder-Mead

(e) Map 6 FFH (f) Map 7 FFH (g) Map 8 FFH (h) Map 9 FFH

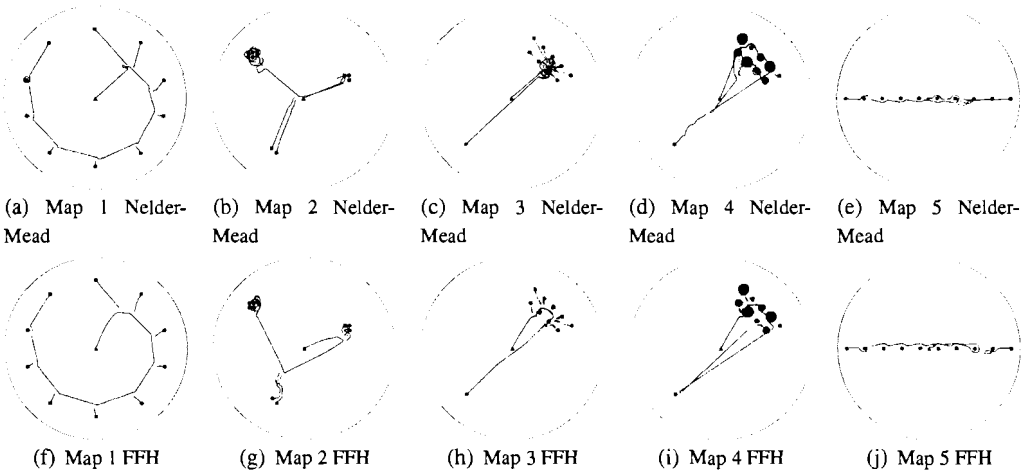
Figure 5.2: Typical paths taken for maps 6 through 9. Visiting order is fixed, collisions are disabled, and obstacles are present.

Table 5.2: Mean and standard deviation of total energy used for rendezvous with obstacles present and collisions disabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 6		Map 7		Map 8		Map 9	
	μ	σ	μ	σ	μ	σ	μ	σ
Nelder-Mead	122.46	11.92	274.93	0.14	78.11	9.65	69.88	1.31
FFH	105.15	0.14	321.50	3.02	62.26	0.54	68.06	1.13
Single point	283.17	6.54	275.94	0.17	149.26	12.15	71.30	0.17

'stop-and-go' behaviour. For example, in Figure 5.1(a) Nelder-Mead causes all the robots to move towards their meeting location at the beginning of the trial, while in Figure 5.1(f), workers do not approach their final meeting location until the tanker is near by. This emergent behaviour influences the duration of trials.

Table 5.1 shows that both Nelder-Mead and frugal feeding heuristic result in lower energy usage as compared with single point rendezvous, unless the rendezvous solution coincides with the frugal feeding solution. It must be noted, however, that the single point rendezvous results include the tanker. If an analogy is drawn that single point rendezvous is the best case tankerless recharging outcome, a tanker would not participate in any such rendezvous, and therefore result in lower energy costs. This is discussed in more detail in Section 6.2.



(a) Map 1 Nelder-Mead (b) Map 2 Nelder-Mead (c) Map 3 Nelder-Mead (d) Map 4 Nelder-Mead (e) Map 5 Nelder-Mead

(f) Map 1 FFH (g) Map 2 FFH (h) Map 3 FFH (i) Map 4 FFH (j) Map 5 FFH

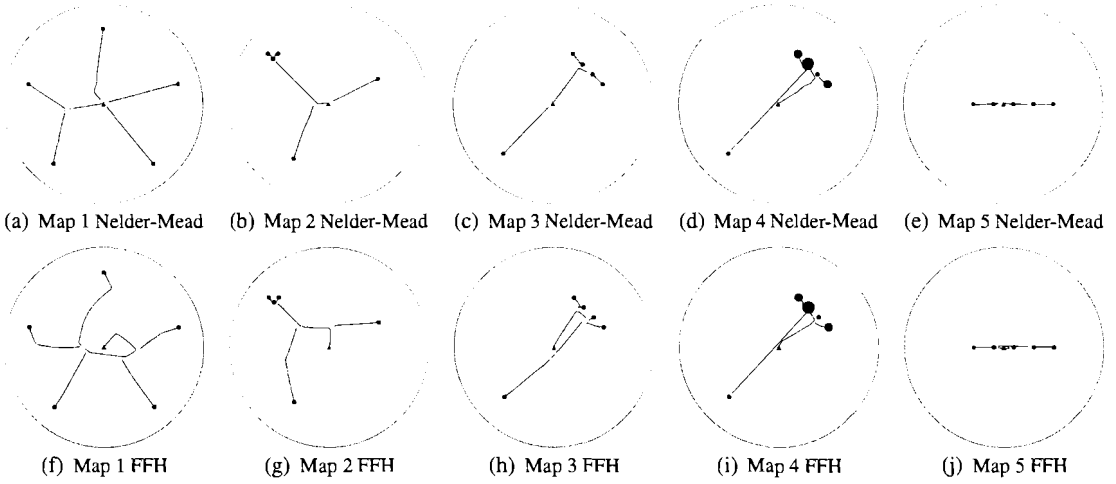
Figure 5.3: Typical paths taken for maps 1 through 5. Visiting order is fixed, collisions are enabled, and no obstacles are present.

Table 5.3: Mean and standard deviation of total energy used for rendezvous with no obstacles present and collisions enabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Nelder-Mead	121.86	6.59	132.01	17.08	117.40	26.67	117.47	9.77	61.57	3.62
FFH	101.06	21.07	136.34	24.13	59.45	1.37	85.25	0.20	56.95	1.07
Single point	222.64	69.45	328.86	108.20	278.69	73.45	460.67	186.61	235.05	72.05

Obstacles

Several experiments are performed with obstacles present to demonstrate method suitability in a more realistic environment. Figure 5.2 shows sample paths taken in the presence of obstacles. With the help of VFH, both methods do a suitable job of negotiating obstacles, although Figures 5.2(b) and 5.2(f) demonstrate the deficiency of using an obstacle avoidance method that suffers from the local minima problem. As robots take evasive action due to obstacles, they do not always choose the most energy efficient path. The result is robots taking the longer of two paths around an obstacle, which is not energy efficient.



(a) Map 1 Nelder-Mead (b) Map 2 Nelder-Mead (c) Map 3 Nelder-Mead (d) Map 4 Nelder-Mead (e) Map 5 Nelder-Mead

(f) Map 1 FFH (g) Map 2 FFH (h) Map 3 FFH (i) Map 4 FFH (j) Map 5 FFH

Figure 5.4: Typical paths taken for maps 1 through 5. Visiting order can change, collisions are disabled, and no obstacles are present.

Table 5.4: Mean and standard deviation of total energy used for rendezvous with no obstacles present and collisions disabled. There are 5 workers per experiment and visiting order varies to minimize energy usage.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Nelder-Mead	71.43	1.05	39.81	0.12	31.52	0.49	42.72	0.64	16.49	0.10
FFH	85.71	0.71	45.59	0.61	42.69	0.16	42.49	0.08	19.53	0.03
Single point	73.17	0.10	56.03	0.17	42.87	0.20	58.47	0.22	21.95	0.12

Interference

In our experiments, the degree to which interference degrades performance varies by map. For example, Figures 5.3(a) and 5.3(f) show paths which suffer minimally since robots do not approach each other. In contrast, Figures 5.3(c) and 5.3(h) show cyclic behaviour caused by robots vying for the same meeting position while avoiding collisions.

The effects of interference can be seen when comparing the results in Table 5.3 with Table 5.1. Map 1 energy usage exhibits a small mean total energy increase when interference is introduced. On the other hand, map 2 shows mean total energy usage nearly doubling.

Table 5.5: Mean and standard deviation of total energy used for rendezvous with obstacles present and collisions enabled. There are 5 workers per experiment and visiting order varies to minimize energy usage.

Method	Map 6		Map 7		Map 8		Map 9	
	μ	σ	μ	σ	μ	σ	μ	σ
Nelder-Mead	128.17	19.65	172.46	13.89	54.00	8.60	49.17	5.73
FFH	93.15	11.20	176.22	15.83	77.82	10.31	47.54	0.69
Single point	198.16	35.31	173.18	7.98	77.62	9.70	53.06	2.99

5.3.5 Variable order results

Our proposed solutions are paired with two ordering algorithms in an effort to demonstrate a complete solution. Figure 5.4 shows resulting paths for Nelder-Mead paired with brute force ordering and the frugal feeding heuristic paired with nearest neighbour ordering. Both lead to tanker successfully visiting each worker. Table 5.4 shows that brute force ordering leads to slightly lower energy use in some maps.

Collisions and interference

Table 5.5 presents the results for variable order experiments with collisions enabled in the presence of obstacles. These can be considered the most realistic of our experiments and suggest that both Nelder-Mead and the frugal feeding heuristic perform at least as well as single point rendezvous under most conditions in the presence of interference and obstacles.

5.3.6 Statistical analysis

A statistical analysis evaluating the statistical significance of the difference between Nelder-Mead and the frugal feeding heuristic was performed by Yaroslav Litus and is shown in Appendix B. The analysis accounts for environmental differences and does not assume normality of samples. It finds that Nelder-Mead takes on average 8.72 more units of energy to achieve all required meetings, which is less than 15% of total energy spending and is not very notable from a practical point of view.

5.3.7 Computation time

A special experiment was performed to compare the computation time used by Nelder-Mead and the frugal feeding heuristic for a larger number of robots. For each method, we vary the number of workers from 1 to 50 in steps of 5 and measure the run time. A reported run time is the time t

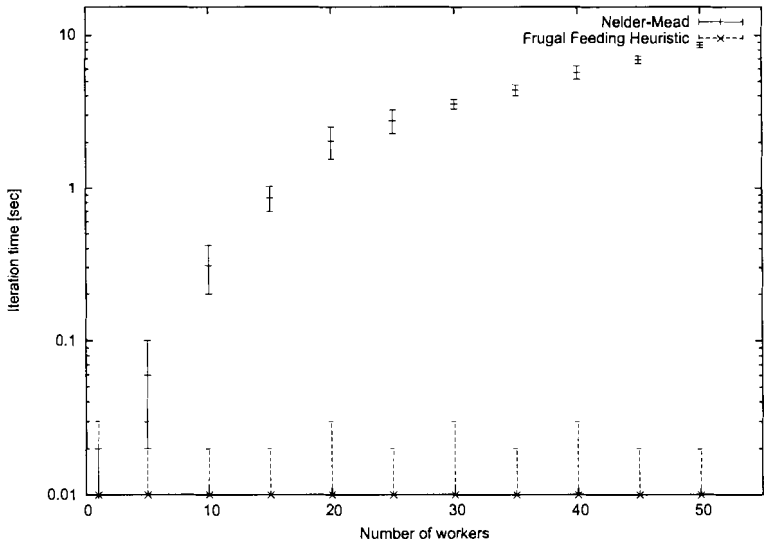


Figure 5.5: Experimental Computation Time

elapsed for one calculation cycle to finish. For example, a run time t of 2 seconds shows that it will take 2 seconds to determine the set of meeting points by Nelder-Mead or calculate the movement directions by the frugal feeding heuristic. Figure 5.5 shows how computation time changes as the number of robots increases.

It can be seen that computation times for the fixed order Nelder-Mead method increase monotonically. At 50 workers, each computation cycle takes 10 seconds. We believe that such delays can result in cyclical motion and sub-optimal paths in the presence of obstacles and interference.

The frugal feeding heuristic, to the contrary, shows no apparent increase in computation time as the number of workers grows. Theoretically, the time should grow linearly with a very small slope (the time required to sum two weighted unit vectors), however, this small increase is overshadowed by operating system process scheduling noise for the range of population size we consider. It should be noted that this experiment measures the total calculation time for all robots in the system. The actual per-robot direction calculation time is independent of population size, and thus the method is perfectly scalable.

5.4 Discussion

5.4.1 Nelder-Mead versus the frugal feeding heuristic

The statistical analysis in Section 5.3.6 argues that in our test environments, the difference in Nelder-Mead and frugal feeding heuristic performance is not significant in practice. Despite this, the two methods do have some notable differences which affect their suitability.

In our implementation of the Nelder-Mead numerical technique, repetitive computation with similar starting positions yields slightly different results each time. This is responsible for the emergence of two energy wasting behaviours. First, Nelder-Mead tends to result in worker robots slowly 'creeping' in the direction of a meeting location. This occurs because the numerical algorithm may dictate a location during one iteration, then dictate a slightly different location during the second iteration. This causes a robot to slowly move between successively dictated points, resulting in a path that is not necessarily a straight line. An example of this effect can be seen by examining the irregular worker robot paths in Figure 5.1(a).

A second emergent behaviour is due to the inherent instability of the frugal feeding problem. Given some environment, there may be multiple paths that lead to an optimal solution. Recomputation using Nelder-Mead will result in the prescribed solution jumping back and forth between the multiple optimal solutions. An example of this occurs when a tanker and one worker with equal weights try to decide where to meet. The Nelder-Mead method proposes a series of ever-changing meeting points along the line between the tanker and worker. More debilitating effects occur when the ever-changing paths cause robots to move in irregular paths.

Similarly to Nelder-Mead, the frugal feeding heuristic exhibits stop-and-go movement. This results from a robot R_i meeting its stopping condition briefly as the other robots being considered continue moving. Once these other robots move close enough to cause R_i to no longer meet its stopping condition, R_i quickly moves ahead to again meet its stopping condition. The result is a stop-and-go motion. While such motion has benefits in avoiding interference, Barilli et al. [2] suggests that it is not energy efficient.

Finally, it is possible to describe scenarios where the frugal feeding heuristic dictates a sub-optimal path. Consider the initial conditions shown in Figure 5.6 and the charging order R_1, R_2, R_3. In this case, the frugal feeding heuristic will result in tanker R_0 moving towards worker R_1 until worker R_2 (which is moving towards R_1) reaches R_1. At this point both R_1 and R_2 will move

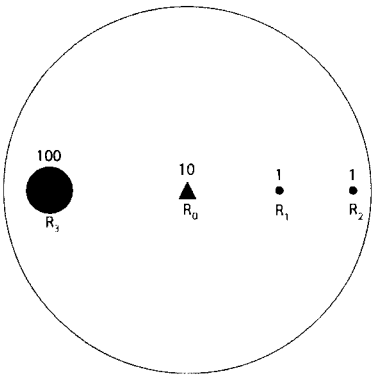


Figure 5.6: An example world where the frugal feeding heuristic results in a sub-optimal path. Locomotion cost is shown above robot.

towards R_0, while R_0 continues to move towards R_1 and R_2. After a meeting with R_1 and R_2 is achieved, R_0 begins moving back towards R_3. This is not the most energy efficient path in this scenario. A more efficient path would have robots R_1 and R_2 move to R_0, after which R_0 would move to R_3.

5.4.2 The effects of obstacles

While both Nelder-Mead and the frugal feeding heuristic are demonstrated in an environment with obstacles, nearly all of the paths in Figure 5.2 show that the obstacles are not considered by either algorithm when prescribing an energy efficient path. This is the expected result, as our chosen cost function does not consider obstacles directly. The result is an energy efficient path which tolerates obstacles, not an energy efficient path through an obstacle-filled environment. The frugal feeding problem in the presence of obstacles is an area of future research.

The stop-and-go movements resulting from the frugal feeding heuristic provide for stability in the presence of obstacles. Visual inspection suggests that some subset of robots is usually stationary. This results in minimal deviation from the chosen paths as robots are forced to change course. Nelder-Mead, on the other hand, results in more chaotic outcomes. Most robots tend to be moving at the same time, and when an obstacle is encountered, some robots choose evasive action that hinders the progress of a rendezvous. As an example, consider the hypothetical case where all robots are attempting to rendezvous inside an obstacle. With Nelder-Mead, upon seeing the obstacle, each robot decides to turn right. This results in robots circling the obstacle indefinitely. In contrast, the subset of stationary robots resulting from frugal feeding heuristic will often be enough to break the symmetry. More work is required in this area.

5.4.3 The effects of interference

Enabling inter-robot collisions is shown to result in increased mean total system energy usage. This is largely caused by robots circling around each other vying for the same space. Since cyclic behaviour wastes energy, special interference reducing provisions need to be made to reduce it, as suggested in [8, 38, 46–48].

Intuitively, maps which result in tanker absorbed solutions will suffer more from interference as robots attempt to rendezvous around the tanker. Most of our results exhibit this to some extent, while the single point rendezvous results can be seen to be most affected, as all 10 robots attempt to crowd around the tanker. As a basic attempt to mitigate this problem, our robots are programmed to leave the rendezvous area (without adding to total energy usage) once a meeting has occurred in an attempt to minimize such interference.

5.4.4 Variable order techniques

Brute force ordering is shown to produce lower mean energy usage than nearest neighbour ordering, but is very computationally expensive. In fact, we have to restrict our variable order set of experiments to 5 worker robots in order to have manageable computation times. Our results show that nearest neighbour can produce reasonable orderings in some environments, but can degrade under specific conditions. As an example, notice the cyclic path taken by tanker in Figure 5.4(j).

Continuous recomputation of the charging order has benefits and drawbacks. In experiments where robots are forced from their paths due to interference or obstacles, recomputation of the ordering is an asset as robots adjust to the new instance of the frugal feeding problem. On the other hand, in certain settings, recomputing the order causes robots to alter a chosen path before a meeting has been achieved, resulting in a detour.

5.5 Conclusion

In this chapter, we considered a practically inspired optimization problem of finding an energy-minimizing route for a heterogeneous robot system where a single service robot needs to meet a number of worker robots. This problem has two components, combinatorial (finding the best meeting order) and analytic (finding the best meeting places). In Chapter 4 we argued that the combinatorial component is NP-hard and here we propose a scalable distributed heuristic solution to the analytic component. By exploiting embodiment and the natural parallelism of the robot system

the heuristic achieves good results with very small computational demands.

Experiments are performed and theoretical results confirmed in simulation. Nelder-Mead is compared with the frugal feeding heuristic in a variety of environments to gauge their effectiveness at achieving multi-point rendezvous. Both methods perform well, with each emerging as the preferred method under certain conditions. Both are favorably compared to single point rendezvous results.

Additional experiments are performed by applying a brute force ordering technique to Nelder-Mead and a nearest neighbour ordering technique to the frugal feeding heuristic in an attempt to demonstrate a complete solution to the frugal feeding problem. Brute force ordering is empirically shown to produce better results at the cost of computation requirements. We demonstrate computation times in a separate experiment.

Several experiments are performed with obstacles and collisions enabled to gauge the effects of interference. In these experiments, robots depend heavily on VFH to avoid contact. Both Nelder-Mead and the frugal feeding heuristic are shown to cope with some level of obstacles and interference. Drawbacks are identified as both methods are deemed to merely tolerate obstacles instead of considering them when developing energy efficient motion plans.

Chapter 6

Discussion and Conclusion

6.1 The tanker approach: summary of contributions

Chapter 2 introduces the use of an energy distributing tanker robot in a system of autonomous worker robots. This approach is presented as a suitable alternative for recharging robot teams and is demonstrated in simulation. Several design attributes are explored, and general conclusions conjectured about the the properties of tanker based recharging. A design to physically achieve the coupling required to perform tanker based recharging is shown. Tanker to charging station coupling is demonstrated.

In Chapter 3 we move away from the ad-hoc techniques employed in Chapter 2 and attempt to structure tanker recharging so that it is more efficient and predictable. We introduce the notion of an energy efficient rendezvous in an attempt to answer the question "If we're concerned with energy efficiency, where is the best place to meet in order to recharge." We present two techniques for answering this question and discuss their merits. We demonstrate their suitability for achieving rendezvous in simulation.

Chapter 4 introduces the frugal feeding problem as the problem of finding the most energy efficient way for a tanker to meet with each worker. The problem is divided into two sub-problems: the order in which tanker should meet with each worker, and where each meeting should take place. The problem is analyzed for special cases and complexity. The ordering portion of frugal feeding is shown to be NP-hard. The chapter also outlines a partial discrete and numerical method for solving the frugal feeding problem. These solutions ignore the combinatorial portion of the problem and instead solve only the location component. The merits of each technique are discussed.

Chapter 5 presents a heuristic for solving the location component of frugal feeding. Simulation is used to demonstrate the heuristic and compare it to the Nelder-Mead numerical method presented in Chapter 4 as well as the rendezvous method in Chapter 3. The heuristic is then paired with an ordering heuristic and Nelder-Mead is paired with a brute force ordering technique in order to demonstrate two complete solutions to the frugal feeding problem. All solutions are then evaluated in several different environments: all combinations of obstacles present and absent, collisions enabled and disabled, and several initial robot arrangements. Final experiments are performed to show experimental running times for each method.

In addition to the author's supervisor, Richard Vaughan, our colleague Yaroslav Litus contributed significantly to some of the ideas in this thesis in the context of a collaborative project in the Autonomy Lab. Specifically, Litus formalized the generalized energy efficient rendezvous problem, proposed, formalized, and proved the convergence of the local solution, formalized the frugal feeding problem, and proposed, formalized, and proved the convergence of the frugal feeding heuristic. The remaining content and ideas are the author's own.

6.2 Tanker based recharging versus base station recharging

We have shown that tanker based recharging can have advantages over traditional docking station recharging. We argue that the division of labor which results from the introduction of a tanker can lower system complexity and cost while increasing modularity and productivity. Worker robots are not constrained by their requirements to maintain sufficient operational energy levels, and can therefore be less sophisticated. As an example, consider the utility of driving a ride-on-lawnmower to the gas station instead of delivering fuel in a gas can.

We set out to see the conditions under which the use of a tanker would make sense. We have shown that in several experimental scenarios, having a tanker based recharging scheme yields lower aggregate system energy usage. Further, since any solution to the frugal feeding problem makes provisions for the tanker absorbed subset of solutions, tanker based recharging should always be expected to perform at least as well as traditional base station recharging if the choice to use a tanker has been made and removing it from our system of robots is not an option.

However, the question of whether to initially include a tanker is not as straight forward. The introduction of a tanker into the system changes the rendezvous and frugal feeding outcomes. It is possible to construct a scenario where having a tanker increases aggregate system energy usage.

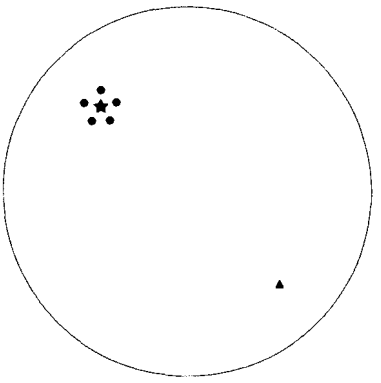


Figure 6.1: A world where tanker (triangle) based recharging is less efficient than traditional base station (star) recharging.

Consider the multi-robot system in Figure 6.1. In this example, having robots recharge at the charging station would use less energy than requiring the tanker to travel all the way to the charging station and then distributing energy. Any tanker based recharging scheme needs to take into account such a scenario. It is straightforward to see that if a tanker is initially placed at the charging station, tanker based recharging will yield results at least as good as base station recharging. On the other hand, if a scenario always results in tanker absorbed recharging, then base station recharging will perform as well as tanker based recharging.

6.3 Future work

We argue for the utility of tanker based recharging under certain conditions. However, before this recharging strategy can be used effectively, several remaining problems need to be addressed.

The visiting order for workers has been shown to be NP-hard. Since brute-force ordering will not scale well, a heuristic is needed to provide good ordering. The nearest neighbour ordering heuristic used in Chapter 5 may be a good first step, but degrades under certain conditions. It may also be possible to solve the frugal feeding problem concurrently with the ordering problem.

A second critical issue is that of timing. A practical tanker based recharging scheme will require the tanker to return to a charging station periodically to replenish its own fuel store. The decision of when to do this is non-trivial. Consider a tanker which has half of its energy remaining. If it drives by the charging station, should it stop to refuel? This question becomes harder once we consider current worker energy stores and their rates of discharge. If all workers are full, then stopping to top

up seems like a good idea, however, if a worker is about to run out of energy, then the situation is more urgent. A solution to this timing problem (including starvation) will likely draw from the body of work on resource scheduling problems.

Third, obstacles and interference should be accounted for, not simply tolerated. While our experiments show that our methods work in the presence of obstacles and interference, all claims of optimality are invalidated when a robot takes a detour to avoid an obstacle. Integrating a global path planner, such as [18], as part of the cost function may lead to energy efficient paths in the presence of obstacles, especially if we retain the assumption of having an environment map *a priori*. An approach to limiting the effects of robot-to-robot interference may be found in the RAGE project [8, 38, 46–48].

Finally, a more realistic cost metric would provide more convincing evidence of the usefulness of these techniques in the real world. Energy costs due to acceleration, changes in direction, and idling are not considered in this work. For example, trials performed using the frugal feeding heuristic often resulted in repetitive acceleration and deceleration. However, we know from [2] that this kind of motion should be avoided in the interest of energy efficiency. A good cost model should strive to eliminate such inefficiencies.

Appendix A

FFH: Proof of Correctness

This proof is due to Yaroslav Litus, based on the work by Yaroslav Litus, Pawel Zebrowski, and Richard Vaughan submitted to the IEEE Transactions on Robotics on June 11th, 2007.

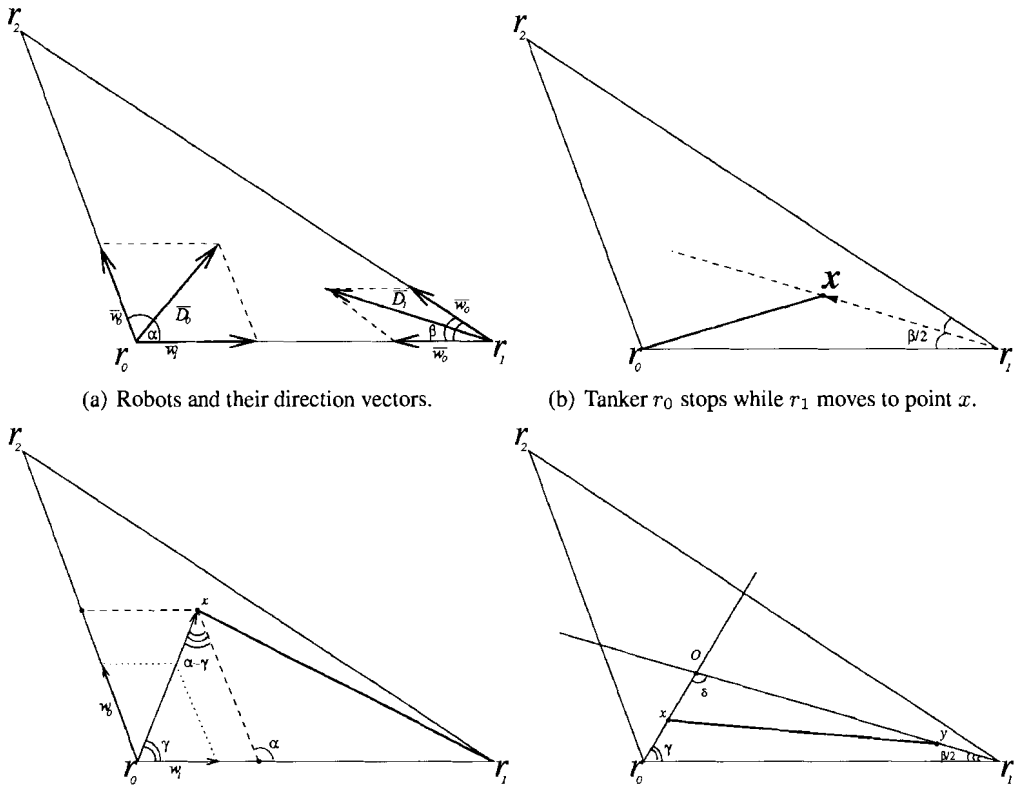
A.1 Proof of correctness and run time bounds

Formally, r_0 (the tanker) and r_1 (the next robot to be charged) move along the approximated antigradient of the cost function $g(p_1) = w_0||r_0 - p_1|| + w_0||p_1 - p_2|| + w_1||p_1 - r_1||$ using the current position of robot r_2 as an approximation to the unknown solution point p_2. The rest of the robots $r_j, j = 2, ..., k$ 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}.

Below we prove that the heuristic is correct and all of the robots are eventually charged. To do this we bound the time required for the robots to meet.

Theorem 3. *For any initial locations $r_j, j = 0, ..., k$ and meeting range s, if robots r_0, r_1 recalculate their movement direction $\vec{D}_j$ every time they travel distance $\epsilon < s/2$ then after at most $\frac{4||r_0 - r_1||^2}{\epsilon(s-2\epsilon)}$ iterations r_0 and r_1 will meet.*

Proof. Consider the situation depicted in Fig A.1(a). We need to show that after r_0 and r_1 stop or move some distance ϵ along the directions $\vec{D}_0$ and $\vec{D}_1$ as prescribed by the algorithm, distance $||r_0 - r_1||$ decreases significantly enough that in a finite time this distance is smaller than meeting range s. The proof proceeds in four steps. First, we show that r_0 and r_1 will never satisfy their stopping conditions simultaneously. This means that at least one of them moves. Second, we bound



(a) Robots and their direction vectors.

(b) Tanker r_0 stops while r_1 moves to point x.

(c) Tanker r_0 moves to point x, worker r_1 stops.

(d) Tanker r_0 moves to point x, worker r_1 moves to point y.

Figure A.1: Illustrating the frugal feeding heuristic correctness proof. r_0 is the tanker robot, r_1 is the worker robot the tanker should meet next, r_2 is the worker robot to meet after r_1.

the decrease of distance in one iteration in the case when r_0 stops and r_1 moves. Then we do the same for the case when r_0 moves and r_1 stops. Finally, we bound the decrease of distance for the case when both r_0 and r_1 move.

First we establish the conditions for r_0 and r_1 to stop. $\angle r_2 r_0 r_1 = \alpha, \angle r_2 r_1 r_0 = \beta$. Norms of direction vectors $||\vec{D}_0||^2 = w_1^2 + w_0^2 + 2w_1 w_0 \cos\alpha$, $||\vec{D}_1||^2 = 2w_0^2(1 + \cos\beta)$. r_0 stops when $||\vec{D}_0|| < w_0$, or $w_1^2 + w_0^2 + 2w_1 w_0 \cos\alpha < w_0^2$ which, given positive weights simplifies to

$$w_1 + 2w_0 \cos\alpha < 0; \text{(stopping conditions for } r_0) \tag{A.1}$$

Similarly, for r_1 the stopping conditions are $||\vec{D}_1|| < w_1$:

$$2w_0^2(1 + \cos\beta) < w_1^2; \text{(stopping conditions for } r_1) \tag{A.2}$$

Note, that since w_0 and w_1 are positive, (A.1) implies $\alpha > \pi/2$.

Both robot stop. Substituting $w_1^2 < 4w_0^2 \cos^2\alpha$ from (A.1) into (A.2) and simplifying gives

$$(1 + \cos\beta) < 2\cos^2\alpha, \tag{A.3}$$

As $\beta \leq \pi - \alpha$, and $\alpha > \pi/2$ we have $|\cos\beta| \geq |\cos\alpha|$. Plugging this into (A.3) and simplifying gives

$$2\cos^2\beta - \cos\beta - 1 > 0 \tag{A.4}$$

Solving this for $\cos\beta$ we get $\cos\beta \in (-\infty, -1/2) \cup (1, \infty)$ which means $\beta > 5\pi/6$. Since $\alpha > \pi/2$, $\alpha + \beta > \pi$ which violates the triangle sum of angles property. Thus, the system of inequalities (A.1) and (A.2) has no valid solutions. This means robots r_0 and r_1 can not stop simultaneously.

r_0 stops, r_1 moves. This situation is shown in Fig A.1(b). Initially the distance between robots is $l_t = ||r_0 - r_1||$. After r_1 moves to point x traveling distance ϵ, the distance between robots becomes $l_{t+1} = ||r_0 - x||$. Note, that $\vec{D}_1$ always bisects $\angle\beta$.

Using the law of cosines, $l_{t+1}^2 = l_t^2 + \epsilon^2 - 2l_t\epsilon\cos\beta/2$. Since $l_t > s$ and $\beta/2 < \pi/4$ (because $\alpha > \pi/2$) the following is valid:

$$\begin{aligned} l_{t+1}^2 &< l_t^2 + \epsilon^2 - 2l_t\epsilon\sqrt{2}/2 \\ &< l_t^2 + \epsilon^2 - \sqrt{2}l_t\epsilon \\ &< l_t^2 + \epsilon^2 - \sqrt{2}\epsilon s; \end{aligned} \tag{A.5}$$

This allows us to bound the decrease in the squared distance between robots in one iteration as:

$$l_t^2 - l_{t+1}^2 > \epsilon(\sqrt{2}s - \epsilon) \tag{A.6}$$

r_0 moves, r_1 stops. This situation is illustrated by Fig A.1(b). Here tanker r_0 moves to point x, decreasing the distance to the worker from $l_t = ||r_0 - r_1||$ to $l_{t+1} = ||r_1 - x||$. The distance tanker travels is $\epsilon = ||x - r_0||$.

By the law of cosines

$$l_{t+1}^2 = l_t^2 + \epsilon^2 - 2\epsilon l \cos\gamma \tag{A.7}$$

To bound γ we apply the law of sines, $w_0/\sin\gamma = w_1/\sin(\alpha - \gamma)$. Starting with $w_0 \sin(\alpha - \gamma) = w_1 \sin\gamma$ and doing trigonometric transformations we derive

$$\tan\gamma = \frac{w_0 \sin\alpha}{w_1 + w_0 \cos\alpha} \tag{A.8}$$

Since $\beta \leq \pi - \alpha$, $\cos\beta \geq \cos(\pi - \alpha)$. This inequality can be plugged into stopping condition (A.2) to get $2w_0^2(1 + \cos(\pi - \alpha)) < w_1^2$. The latter simplifies to $\cos\alpha > \frac{2w_0^2 - w_1^2}{2w_0^2}$. Plugging this into (A.8) and using $w_1 \leq 2w_0$ we obtain

$$\begin{aligned}
\tan\gamma &< \frac{w_0 \sin\alpha}{w_1 + \frac{2w_0^2 - w_1^2}{2w_0}} \\
&< \frac{2w_0^2}{2w_0 w_1 + 2w_0^2 - w_1^2} \\
&\leq \frac{2w_0^2}{w_1^2 + 2w_0^2 - w_1^2} \\
&= 1
\end{aligned} \tag{A.9}$$

This implies $\gamma \in [0, \pi/4]$. We use this bound with (A.7) to arrive at the same bound on squared distance as (A.5) describes. Thus, the bound (A.6) applies for this case as well.

Both robots move Fig A.1(d) shows robot r_0 moving to point x and robot r_1 moving to point y. The distance between robots changes from $l_t = ||r_0 - r_1||$ to $l_{t+1} = ||x - y||$. Both robots move equal distance $||x - r_0|| = ||y - r_1|| = \epsilon$. We will use additional notation $a = ||O - r_0||, b = ||O - r_1||$.

Using the law of cosines,

$$\begin{aligned}
l_{t+1}^2 &= (a - \epsilon)^2 + (b - \epsilon)^2 - 2(a - \epsilon)(b - \epsilon)\cos\delta \\
&= l_t^2 + 2\epsilon(1 - \cos\delta)(\epsilon - a - b)
\end{aligned} \tag{A.10}$$

We start by bounding δ. Note, that $\gamma + \beta/2 + \delta = \pi$. From (A.1), it follows that r_0 moves only when $w_1 + 2w_0 \cos\alpha \geq 0$. This implies $w_1 + w_0 \cos\alpha > 0$ (given positive weights). The latter could be used in (A.8) to get $\tan\gamma > 0$. Thus, if r_0 moves, $\gamma \in [0, \pi/2]$. Now, consider 2

cases. First, if $\beta \geq \pi/2$ then $\gamma + \beta/2 \leq \pi - \beta/2$ which implies $\delta \geq \pi/4$. Second, if $\beta < \pi/2$, $\gamma + \beta/2 < \pi/2 + \pi/4$. In this case, $\delta > \pi/4$. Thus, we bounded δ and

$$1 - \cos\delta > 1 - \frac{\sqrt{2}}{2} \tag{A.11}$$

Now we need to bound $\epsilon - a - b$. Applying the law of sines we get $l_t/\sin\delta = b/\sin\gamma = a/\sin(\beta/2)$. Thus,

$$\begin{aligned}
\epsilon - a - b &= \epsilon - \frac{l_t(\sin\gamma + \sin(\beta/2))}{\sin\delta} \\
&< \epsilon - \frac{s(\sin\gamma + \sin(\beta/2))}{\sin\delta} \\
&= \epsilon - s\frac{\sin\gamma + \sin(\beta/2)}{\sin(\gamma + \beta/2)} \\
&= \epsilon - s\frac{2\sin(\frac{2\gamma+\beta}{4})\cos(\frac{2\gamma-\beta}{4})}{2\sin(\frac{2\gamma+\beta}{4})\cos(\frac{2\gamma+\beta}{4})} \\
&= \epsilon - s\frac{\cos(\frac{2\gamma-\beta}{4})}{\cos(\frac{2\gamma+\beta}{4})} \\
&\leq \epsilon - s\cos\left(\frac{2\gamma-\beta}{4}\right) \\
&= \epsilon - s(\cos(\gamma/2)\cos(\beta/4) \\
&+ \sin(\gamma/2)\sin(\beta/4)) \\
&< \epsilon - s\cos(\gamma/2)\cos(\beta/4) \\
&< \epsilon - (\sqrt{2}/2)^2 s = \epsilon - \frac{s}{2}
\end{aligned} \tag{A.12}$$

Combining equations (A.10)-(A.12) we bound the decrease in squared distance:

$$l_t^2 - l_{t+1}^2 > (2 - \sqrt{2})\epsilon\left(\frac{s}{2} - \epsilon\right) > \frac{\epsilon}{2}\left(\frac{s}{2} - \epsilon\right) \tag{A.13}$$

Comparing (A.6) and (A.13) we see that the latter is a more conservative estimate of the squared distance decrease. Thus, if the initial distance between robots is L, the upper bound on the number of steps needed to meet is $\frac{4L^2}{\epsilon(s-2\epsilon)}$. □

Corollary 1. *For any initial locations $r_j, j = 0, ..., k$ and sensor range s, if robots recalculate their movement direction $\vec{D}_j$ every time they travel distance $\epsilon < s/2$ then after at most $k\frac{4C^2}{\epsilon(s-2\epsilon)}$ iterations all workers will be charged, assuming charging occurs instantaneously and $C = \max_{i,j} ||r_i - r_j||$.*

Appendix B

FFH: Statistical Analysis

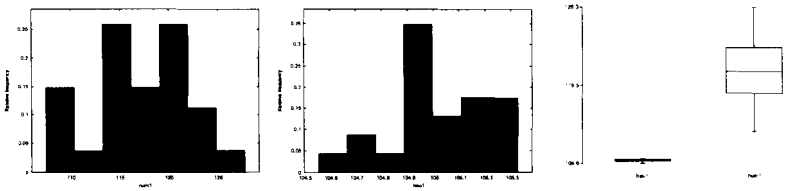
This statistical analysis is due to Yaroslav Litus, based on the work by Yaroslav Litus, Pawel Zebrowski, and Richard Vaughan submitted to the IEEE Transactions on Robotics on June 11th, 2007.

B.1 Statistical analysis

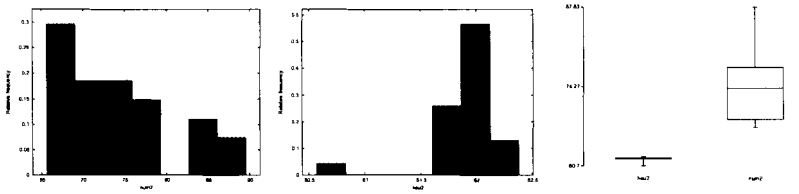
The goal of this statistical analysis is to evaluate the statistical significance of the difference between Nelder-Mead and the frugal feeding heuristic. Figure B.1 presents the histograms and box-plots of the data used in the analysis. All hypothesis testing results assume 5% significance level if not stated otherwise.

The number of observations for any particular map-method pair is at most 26. Because of this, it is difficult to conclude whether the difference between observations can be explained by normal noise. In fact, different normality tests give contradictory results on these samples. Therefore, even though all within-map t-tests reject the hypothesis that Nelder-Mead and the frugal feeding heuristic observations come from the same distribution, the validity of these tests could be questioned. We conduct non-parametric Kruskal-Wallis tests which do not require the normality of samples. For all 5 maps the hypotheses of equal performance of Nelder-Mead method and the frugal feeding heuristic are rejected.

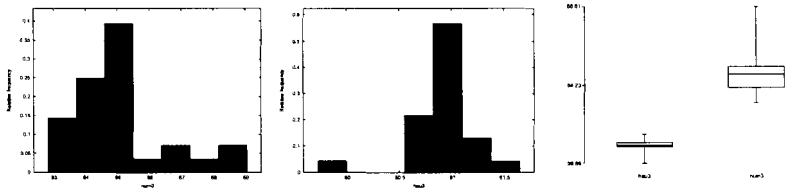
Given the above, we use all available observations to estimate the difference in performance by regressing the energy used for rendezvous on the dummy variable $Method2$ which represents the Nelder-Mead method. The Wald test and likelihood ratio test for group-wise heteroskedasticity reject the hypothesis that the noise variance is equal between maps. The test of joint significance of group means rejects the adequacy of the pooled OLS model in favour of a fixed effects alternative.



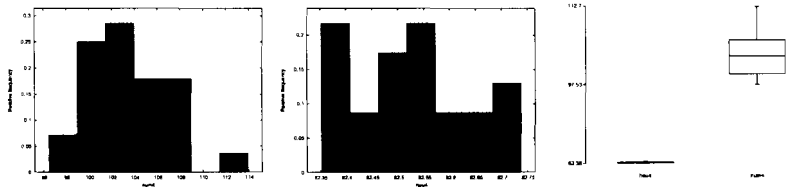
(a) Map 1 Nelder-Mead (b) Map 1 FFH (c) Map 1 Boxplot



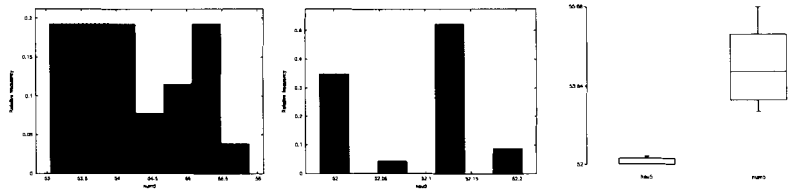
(d) Map 2 Nelder-Mead (e) Map 2 FFH (f) Map 2 Boxplot



(g) Map 3 Nelder-Mead (h) Map 3 FFH (i) Map 3 Boxplot



(j) Map 4 Nelder-Mead (k) Map 4 FFH (l) Map 4 Boxplot



(m) Map 5 Nelder-Mead (n) Map 5 FFH (o) Map 5 Boxplot

Figure B.1: Statistical representation of experimental results. The histograms show the distribution of energy losses for repeated experiments on 5 maps using either Nelder-Mead or the frugal feeding heuristic method. The boxplots present results obtained from both methods for every map. The left plot represents the frugal feeding heuristic, the right plot represents Nelder-Mead.

Thus the model of our choice is a fixed effects model corrected for group-wise heteroskedasticity.

Estimation results are presented in Table B.1. We observe significant fixed effects of each map (dummy variables *du_1, du_2, ..., du_5*) which is reasonable since each map has different theoretically possible minimum energy spending. The effect of $Method2$ is also significant which agrees with the result of the within-map tests. The value of the coefficient indicates that on average Nelder-Mead takes 8.72 more units of energy to converge. However, this distinction is less than 15% of the energy spending, thus it is not very notable from the practical point of view.

Table B.1: Estimation results

Model: WLS estimates using 251 observations
Included 5 cross-sectional units
Dependent variable: Energy
Weights based on per-unit error variances

$$\widehat{\text{Energy}} = \underset{(0.55481)}{106.811}\,\text{du\_1} + \underset{(0.71955)}{63.9341}\,\text{du\_2}$$

$$+\underset{(0.53371)}{58.3833}\,\text{du\_3} + \underset{(0.8597)}{89.1727}\,\text{du\_4} + \underset{(0.64332)}{48.6236}\,\text{du\_5}$$

$$+\underset{(0.50522)}{8.72431}\,\text{Method2}$$

(standard errors in parentheses)

Variable	Coefficient	t-statistic	p-value
du_1	106.811	192.5168	0.0000
du_2	63.9341	88.8531	0.0000
du_3	58.3833	109.3911	0.0000
du_4	89.1727	103.7252	0.0000
du_5	48.6236	75.5821	0.0000
Method2	8.72431	17.2682	0.0000

Statistics based on the weighted data:

Sum of squared residuals	240.319
Standard error of residuals ($\hat{\sigma}$)	0.990401
Unadjusted R^2	0.972120
Adjusted $\bar{R}^2$	0.971551
$F(6, 245)$	1423.79
Akaike information criterion	713.392
Schwarz Bayesian criterion	734.545
Hannan–Quinn criterion	721.905

Statistics based on the original data:

Mean of dependent variable	78.2138
S.D. of dependent variable	22.4821
Sum of squared residuals	5042.10
Standard error of residuals ($\hat{\sigma}$)	4.53652

Appendix C

FFH: Experimental Results

Table C.1: Mean total energy used for rendezvous with no obstacles present and collisions disabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	117.08	4.55	74.40	6.30	65.04	1.43	103.36	3.68	54.28	0.75
FFH	105.00	0.17	61.89	0.29	60.89	0.30	82.53	0.11	52.09	0.07
Single point	150.60	0.26	123.27	0.31	66.60	0.19	131.56	0.98	93.70	0.44

Table C.2: Mean total energy used for rendezvous with obstacles present and collisions disabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 6		Map 7		Map 8		Map 9	
	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	122.46	11.92	274.93	0.14	78.11	9.65	69.88	1.31
FFH	105.15	0.14	321.50	3.02	62.26	0.54	68.06	1.13
Single point	283.17	6.54	275.94	0.17	149.26	12.15	71.30	0.17

Table C.3: Mean total energy used for rendezvous with no obstacles present and collisions enabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	121.86	6.59	132.01	17.08	117.40	26.67	117.47	9.77	61.57	3.62
FFH	101.06	21.07	136.34	24.13	59.45	1.37	85.25	0.20	56.95	1.07
Single point	222.64	69.45	328.86	108.20	278.69	73.45	460.67	186.61	235.05	72.05

Table C.4: Mean total energy used for rendezvous with obstacles present and collisions enabled. There are 10 workers per experiment and visiting order is fixed.

Method	Map 6		Map 7		Map 8		Map 9	
	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	127.80	10.57	380.17	39.86	148.22	36.42	152.91	54.15
FFH	105.54	0.15	355.65	18.28	169.81	28.75	69.66	3.21
Single point	305.73	32.11	357.71	58.59	175.31	21.84	124.50	11.29

Table C.5: Mean total energy used for rendezvous with no obstacles present and collisions disabled. There are 5 workers per experiment and visiting order varies to minimize energy usage.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	71.43	1.05	39.81	0.12	31.52	0.49	42.72	0.64	16.49	0.10
FFH	85.71	0.71	45.59	0.61	42.69	0.16	42.49	0.08	19.53	0.03
Single point	73.17	0.10	56.03	0.17	42.87	0.20	58.47	0.22	21.95	0.12

Table C.6: Mean total energy used for rendezvous with obstacles present and collisions disabled. There are 5 workers per experiment and visiting order varies to minimize energy usage.

Method	Map 6		Map 7		Map 8		Map 9	
	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	106.81	0.61	164.79	0.12	43.35	0.29	39.01	0.36
FFH	82.69	0.11	164.71	0.10	61.42	7.78	46.30	0.18
Single point	172.03	0.58	165.16	0.13	69.27	2.95	47.55	0.19

Table C.7: Mean total energy used for rendezvous with no obstacles present and collisions enabled. There are 5 workers per experiment and visiting order varies to minimize energy usage.

Method	Map 1		Map 2		Map 3		Map 4		Map 5	
	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	72.13	1.34	47.67	4.43	42.14	4.13	45.80	2.61	17.87	0.52
FFH	86.65	1.00	61.15	12.31	44.13	1.56	44.18	0.29	20.77	0.07
Single point	73.15	0.15	60.24	0.94	47.69	2.99	73.46	7.48	23.23	0.17

Table C.8: Mean total energy used for rendezvous with obstacles present and collisions enabled. There are 5 workers per experiment and visiting order varies to minimize energy usage

Method	Map 6		Map 7		Map 8		Map 9	
	σ	μ	σ	μ	σ	μ	σ	μ
Nelder-Mead	128.17	19.65	172.46	13.89	54.00	8.60	49.17	5.73
FFH	93.15	11.20	176.22	15.83	77.82	10.31	47.54	0.69
Single point	198.16	35.31	173.18	7.98	77.62	9.70	53.06	2.99

Bibliography

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[2] A. Barili, M. Ceresa, and C. Parisi. Energy-saving motion control for an autonomous mobile robot. In *The 1995 IEEE International Symposium on Industrial Electronics*, pages 674–676, Athens, Greece, July 1995.

[3] Richard Bellman. *Dynamic programming*. Dover Publications, 2003.

[4] V. Boltyanski, H. Martini, and V. Soltan. *Geometric methods and optimization problems*. Kluwer Academic Publishers, 1999.

[5] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[6] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 540–545, Lausanne, Switzerland, September 2002.

[7] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1985.

[8] Sarah Brown, Mauricio Zuluaga, Yinan Zhang, and Richard Vaughan. Rational aggressive behaviour reduces interference in a mobile robot team. In *Proceedings of the International Conference on Advanced Robotics*, pages 741–748, Seattle, Washington, July 2005.

[9] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006.

[10] Zvi Drezner, editor. *Facility location. A survey of application and methods*. Springer-Verlag, 1995.

[11] Gregory Dudek and Nicholas Roy. Multi-robot rendezvous in unknown environments, or, what to do when you're lost at the zoo. In *Proceedings of the AAAI National Conference Workshop on Online Search*, Providence, Rhode Island, July 1997.

[12] S. Emami, G. Wyeth, M. Milford, and D. Prasser. Framework for the long-term operation of a mobile robot via the internet. In *Australasian Conference on Robotics and Automation*, Sydney, Australia, December 2005.

[13] Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, Gaurav Sukhatme, and Maja J. Matarić. Most valuable player: A robot device server for distributed control. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 1226–1231, Wailea, Hawaii, November 2001. IEEE.

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

[15] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam. Vision-based autonomous robot self-docking and recharging. In *ISORA 11th International Symposium on Robotics and Applications*, pages 1–8, Budapest, Hungary, July 2006.

[16] Y. Kupitz and H. Martini. Geometric aspects of the generalized fermat-torricelli problem. *Bolyai Society Mathematical Studies*, 6:55–129, 1997.

[17] Mark A. Lanthier, Doron Nussbaum, and Tsuo-Jung Wang. Calculating the meeting point of scattered robots on weighted terrain surfaces. In *Proceedings of the 2005 Australasian symposium on Theory of computing*, pages 107–118, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[18] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[19] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. In *Proceedings of 42th IEEE Conference on Decision and Control*, pages 1926–1931, Maui,Hawaii, 2003.

[20] Yaroslav Litus, Richard T. Vaughan, and Pawel Zebrowski. The frugal feeding problem: energy-efficient, multi-robot, multi-place rendezvous. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 27–32, April 2007.

[21] John H. Mathews and Kurtis K. Fink. *Numerical Methods Using Matlab*. Prentice-Hall Inc., 4 edition, 2004.

[22] Yongguo Mei. *Energy-Efficient Mobile Robots*. PhD thesis, Purdue University, December 2006.

[23] Yongguo Mei, Yung-Hsiang Lu, Y. Charlie Hu, and C.S. George Lee. Energy-efficient mobile robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 505–511, Orlando, Florida, USA, May 2006.

[24] Yongguo Mei, Yung-Hsiang Lu, C.S. George Lee, and Y. Charlie Hu. Energy-efficient motion planning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4344–4349, New Orleans, Louisiana, USA, April 2004.

[25] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[26] Illah Nourbakhsh, Judith Bobenage, Sebastian Grange, Ron Lutz, Roland Meyer, and Alvaro Soto. An affective mobile robot educator with a full-time job. *Artificial Intelligence*, 114(1–2):95–124, 1999.

[27] A. Oh and K. Taylor. Autonomous battery recharging for indoor mobile robots. In *Proceedings of Australian Conference on Robotics and Automation*, August 2000.

[28] C. H. Papadimitriou. Euclidean TSP is NP-complete. *Theoretical Computer Science*, 4:237–244, 1977.

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

[30] Gerhard Reinelt. Fast heuristics for large geometric travelling salesman problems. *ORSA Journal on Computing*, 4(2):206–217, 1992.

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

[32] Neil C. Rowe. Obtaining optimal mobile-robot paths with nonsmooth anisotropic cost functions using qualitative-state reasoning. *The International Journal of Robotics Research*, 16(3):375–399, 1997.

[33] Neil C. Rowe and Yutaka Kanayama. Near-minimum-energy paths on a vertical-axis cone with anisotropic friction and gravity effects. *The International Journal of Robotics Research*, 13(5):408–433, 1994.

[34] K. Schlude. From robotics to facility location: Contraction functions, weber point, convex core. Technical Report 403, ETHZ, Computer Science, 2003.

[35] Milo Silverman, Dan M. Nies, Boyoon Jung, and Gaurav S. Sukhatme. Staying alive: A docking station for autonomous robot recharging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1050–1055, Washington D.C., May 2002.

[36] S. L. Smith, M. E . Broucke, and B. A. Francis. Curve shortening and the rendezvous problem for mobile autonomous robots. *IEEE Transactions on Automatic Control*, 52(6):1154–1159, June 2007.

[37] Z. Sun and J. Reif. On energy-minimizing paths on terrains for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3782–3788, Taipei, Taiwan, September 2003.

[38] Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme, and Maja J. Matarić. Go ahead, make my day: Robot conflict resolution by aggressive competition. In *Proceedings of the 6th International Conference on Simulation of Adaptive Behaviour*, pages 491–500, Paris, France, August 2000.

[39] Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme, and Maja J. Matarić. Lost: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-Robot Systems*, 18(5):796–812, October 2002.

[40] W. G. Walter. *The Living Brain*. W.W. Norton, New York, 1963.

[41] Guiling Wang, Mary Jane Irwin, Piotr Berman, Haoying Fu, and Tom La Porta. Optimizing sensor movement planning for energy efficiency. In *Proceedings of the 2005 international symposium on Low power electronics and design*, pages 215–220, New York, NY, USA, 2005. ACM Press.

[42] J. Wawerla, G. Sukhatme, and M. Matarić. Collective construction with multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2696–2701, Lausanne, Switzerland, October 2002.

[43] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 47–53. ACM Press, 1998.

[44] Pawel Zebrowski, Yaroslav Litus, and Richard T. Vaughan. Energy efficient robot rendezvous. In *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pages 139–148, May 2007.

[45] Pawel Zebrowski and Richard Vaughan. Recharging robot teams: A tanker approach. In *Proceedings of the International Conference on Advanced Robotics*, pages 803–810, Seattle, Washington, July 2005.

[46] Yinan Zhang. Spatial interference reduction for multi-robot systems using rational and team-based aggression. Master's thesis, Simon Fraser University, July 2006.

[47] Mauricio Zuluaga. Rage amongst the machines: A biological approach to spatial interference in multi-robot systems. Master's thesis, Simon Fraser University, September 2005.

[48] Mauricio Zuluaga and Richard Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2798–2805, Edmonton, Alberta, August 2005.