USING SPATIAL EMBEDDEDNESS AND PHYSICAL EMBODIMENT FOR COMPUTATION IN MULTI-ROBOT SYSTEMS

by

Yaroslav Litus

M.A., National University of Kyiv-Mohyla Academy, 2002

Specialist, National Technical University of Ukraine, 2001

B.Sc., National Technical University of Ukraine, 1999

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

SIMON FRASER UNIVERSITY

Spring, 2011

APPROVAL

Name:	Yaroslav Litus
Degree:	Doctor of Philosophy
Title of Thesis:	Using spatial embeddedness and physical embodiment for computation in multi-robot systems
Examining Committee:	Dr. Greg Mori Chair
	Dr. Richard Vaughan, Associate Professor Computing Science, Simon Fraser University Senior Supervisor
	Dr. Robert Hadley, Professor Computing Science, Simon Fraser University Supervisor
	Dr. Andrei Bulatov, Associate Professor Computing Science, Simon Fraser University Supervisor
	Dr. Alexandra Fedorova, Assistant Professor Computing Science, Simon Fraser University Internal Examiner
	Dr. Owen Holland, Professor of Cognitive Robotics, University of Sussex External Examiner
Date Approved:	January 24, 2011

Declaration of
Partial Copyright Licence

Abstract

This thesis contributes to the understanding of computational capabilities of multi-robot systems by viewing them as devices in which mechanical and electronic components jointly perform computation. We show that a multi-robot system can use physical embodiment and spatial embeddedness as computational resources for two types of problems: (i) a continuous optimization problem of achieving optimal joint robot team configuration, and (ii) a combinatorial problem of sorting robots.

In the continuous problem domain we describe a general approach for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and time to approach some optimal and initially unknown configuration. We provide examples of the application of this general method by solving two non-trivial problems of multi-robot coordination: energy-efficient single point and multiple point rendezvous.

In the combinatorial problem domain we demonstrate a multi-robot system controller that sorts a team of robots by means of its continuous movement dynamics. Between-robot rank comparisons suggested by traditional discrete state sorting algorithms are avoided by coupling neighbors in the order in a Brockett double bracket flow system.

To Squamish SAR

Acknowledgments

I would like to thank my friend and senior supervisor Dr. Richard Vaughan for his patient mentorship and guidance. Many thanks to colleagues from Autonomy Lab for their support and especially to my academic brother Dr. Jens Wawerla for sharing the road and asking hard questions.

I am very grateful to my supervisors Dr. Robert Hadley and Dr. Andrei Bulatov for helping me to complete this work, and to my examiners Dr. Alexandra Fedorova and Dr. Owen Holland whose encouraging prodding allowed me to significantly improve this document.

Special thanks to all my teachers whose efforts took me to this point and will take me further, and to all my friends for their friendship. Thanks to my family for their love and support.

Last and most, thanks to Jenya for everything.

Contents

Approval ii

Abstract iii

Dedication iv

Acknowledgments v

Contents vi

List of Tables x

List of Figures xi

1 Introduction 1

1.1 Overview 1

1.2 Contributions 2

1.3 Thesis structure 2

2 Related work 4

2.1 Architectures 4

2.2 Using a physical body to compute 8

2.2.1 Coevolution of morphology and control 8

2.2.2 Morphological computation 9

2.2.3 Computing by self-assembly 13

2.2.4 Collision based computing 15

2.2.5 Computing in multi-agent systems 17

2.3 Robot path planning . . . 18
2.3.1 Combinatorial methods . . . 19
2.3.2 Sampling-based methods . . . 19
2.3.3 Feedback planning . . . 20
2.3.4 Multi-robot planning . . . 22
2.4 Summary . . . 24

3 Gradient optimization with embodied approximation 25
3.1 Introduction and related work . . . 25
3.2 Distributed gradient optimization with embodied approximation . . . 26
3.3 Discussion . . . 29

4 Rendezvous 30
4.1 Introduction . . . 30
4.2 Problem Characterization and related work . . . 31
4.3 Solutions . . . 32
4.3.1 Centralized (global) solution . . . 32
4.3.2 Local solutions . . . 32
4.4 Experiments . . . 35
4.4.1 Setup . . . 35
4.4.2 Results . . . 37
4.5 Discussion . . . 39
4.5.1 Paths Traversed . . . 39
4.5.2 Static Versus Dynamic . . . 40
4.5.3 Global Versus Local . . . 40

5 Frugal Feeding Problem 42
5.1 Problem description and characterization . . . 42
5.1.1 Related work . . . 44
5.1.2 Analysis . . . 45
5.1.3 Complexity of combinatorial component . . . 47
5.2 Restricted locations case . . . 48
5.2.1 Algorithm: Formal presentation . . . 49
5.2.2 Algorithm: Illustrative example . . . 50

5.3 Continuous case . . . 51
5.3.1 First order method . . . 51
5.3.2 Second order method . . . 51
5.4 Demonstration of proposed numerical algorithms and restricted locations method . 52
5.4.1 Simulation setup . . . 52
5.4.2 Results . . . 54
5.5 A distributed heuristic for the Ordered Frugal Feeding Problem . . . 57
5.5.1 Definition of Ordered Frugal Feeding . . . 57
5.5.2 Algorithm . . . 57
5.5.3 Intuitive explanation . . . 58
5.5.4 Proof of correctness and run time bounds . . . 59
5.6 Experimental evaluation . . . 62
5.6.1 Comparison with Nelder-Mead method . . . 62
5.6.2 Computation time . . . 68
5.6.3 Reintegrating ordering . . . 70
5.6.4 Estimating quality of solutions . . . 71
5.7 Summary . . . 73

6 Robot sorting with a smooth controller 75
6.1 Introduction . . . 75
6.2 Related work . . . 76
6.3 Brockett smooth sorter . . . 77
6.4 Application to robot sorting . . . 79
6.4.1 Theoretical considerations . . . 79
6.4.2 Implementation . . . 80
6.4.3 Demonstration . . . 82
6.5 Discussion . . . 84
6.6 Summary . . . 86

7 Conclusion 88
7.1 Summary . . . 88
7.2 Future work . . . 89

Bibliography 91

List of Tables

4.1 Mean Total Energy Used In Maps Without Obstacles. All Standard Deviations $< 5\%$ 39

4.2 Mean Total Energy Used In Maps With Obstacles. All Standard Deviations $< 6\%$ 39

5.1 Total Energy Used For Rendezvous. All Standard Deviations $< 10\%$. (Table by P. Zebrowski) 65

5.2 Estimation results 69

5.3 Experiment 3: Total Energy Used For Rendezvous. All Standard Deviations $< 10\%$. (Table by P. Zebrowski) 70

5.4 Statistics of approximation factor bounds 72

6.1 Parameters used in Stage simulation 83

List of Figures

4.1 Experimental initial conditions 38
4.2 Typical paths taken (Map 2) 38

5.1 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. (Figure by R. Vaughan) 44
5.2 Illustrative example. (a) Grid world with rectilinear distances. Tanker is denoted as T, workers as $R1, R2, R3$, possible meeting locations as A, B, C. (b) Construction of maps for problem solution. (c) Function, used in the selection of the minimum cost value. 49
5.3 Experimental results. Solution quality is plotted against compute time for each solution method, each of three settings (maps). Maps 1-3 differ only in locomotion cost weights on the single tanker w_0 and five worker $w_i | i > 0$ robots. (Figure by P. Zebrowski) 55
5.4 Rendezvous solutions discovered. Tanker is at the solid spot on figure (a), other points show workers. Circles denote meeting places. Tanker route is shown with solid lines, workers routes are shown with dashed lines. (Figure by P. Zebrowski) . 56
5.5 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. 59
5.6 Typical paths taken for maps 1 through 5. Fixed meeting order. Triangle represents tanker, black disks represent workers, with disk radii proportional to worker weight (movement cost). Gray circles represent meeting range. (Figure by P. Zebrowski) . 65

5.7 Statistical representation of experimental results. Histograms show the distribution of energy losses for repeated experiments on 5 maps using either Nelder-Mead or Frugal Feeding Heuristic methods. Boxplots present results obtained from both methods for every map, left plot representing Frugal Feeding Heuristic, right plot representing Nelder-Mead. 67
5.8 Experimental Computation Time (Figure by P. Zebrowski) 68
5.9 Typical paths taken for maps 1 through 5. Variable meeting order. Triangle represents tanker, black disks represent workers, with radius proportional to worker weight (movement cost). Gray circles represent meeting range. (Figure by P. Zebrowski) 70

6.1 Robots queueing for recharging at the station "C" located near the wall. The share of remaining charge is shown for every robot. Sorting robots increases the probability that all robots can charge before running out of energy. 76
6.2 Dynamics of sorting system (6.3-6.4) initialized with $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. Four components of vector b (vertical axis) are plotted against time (horizontal axis). In 3 500‘ simulation steps the system converges to $b = (6.088, 2.976, 1.999, 0.937)$. 78
6.3 State diagram of the robot sorting controller 81
6.4 Robot trajectories for different initial conditions (rows). $x(t)$ graphs plot x coordinate (vertical axis) against time. $y(x)$ graphs plot y coordinate (vertical axis) against x-coordinate. Disks denote ends of trajectories on $y(x)$ graphs. 87

Chapter 1

Introduction

1.1 Overview

Until late 1930s, computing devices built by humans were mechanical. A set of mechanical parts moved restricted by the kinematics of the mechanism; and this movement constituted computation. With the advance of electric and electronic technology the role of mechanics in computing devices decreased and finally almost disappeared. However, mobile robots and multirobot teams necessarily retain mechanical components. A multirobot system can be seen as a set of mechanical parts, restricted in their movement by the physical properties of the system and control algorithms which govern the robots. This thesis contributes to the understanding of the computational capabilities of such a mechanical-electronic system. We show that a multirobot system can use physical embodiment and spatial embeddedness as useful computational resources for solving two types of problems: a continuous optimization problem of finding and moving to an optimal joint robot team configuration and a combinatorial problem of sorting robots.

In the continuous problem domain we describe a general approach for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and/or time, to approach some optimal and initially unknown configuration. Our approach relies on using embodiment and spatial embeddedness as a surrogate for computational resources, permitting the reduction or elimination of communication or shared memory for conventional parallel computation. Intermediate stages of the gradient descent process are manifested by the locations of the robots, instead of being represented symbolically. Each point in the space-time evolution of the system can be considered an approximation of the solution, which is refined by the agents' motion in response to sensor measurements. We provide examples of the

application of this general method by solving two non-trivial problems of multi-robot coordination: energy-efficient single point and multiple point rendezvous.

In the combinatorial problem domain we demonstrate a multi-robot system controller that sorts a team of robots by means of its continuous movement dynamics. Between-robot rank comparisons suggested by traditional discrete state sorting algorithms are avoided by coupling neighbors in the order in a Brockett double bracket flow system. The robots are reactive agents with information exchanged between robots by means of relative position sensing. A multi-robot simulation with non-holonomic driving, noisy sensor data, collision avoidance and sensor occlusions demonstrates that Brockett double bracket flow system can withstand perturbations introduced into the ideal dynamics by physical limitations of real robots.

Summarizing, this thesis shows how a multi-robot system can use embodiment and spatial embeddedness as computational resources for solving both continuous and combinatorial problems. Apart from the engineering value of this approach, it gives some synthetic proof of the general applicability of embodiment as a part of the computational apparatus.

1.2 Contributions

This thesis contributes to understanding of computational capabilities of multi-robot systems in the following ways:

- It shows that physically embedded agents can use embodiment and spatial embeddedness as a surrogate for computational resources to run parallel gradient descent algorithms.
- It develops and analyzes multi-robot systems that solve non-trivial continuous optimization tasks by combining computational power of a very simple embedded computer with the physical dynamics of interacting robots.
- It presents the first robotic system that solves a combinatorial problem (sorting) by means of its own continuous dynamics.

1.3 Thesis structure

The following chapter reviews the body of work related to the topic of this thesis. First, Section 2.1 outlines robot architectures focusing on their relation to the embodiment. Section 2.2 surveys previous work relevant to the ideas of using a physical body for computing in unconventional ways.

Since this thesis describes solutions to several robot path planning problems, a review of path planning methods is given in Section 2.3.

Chapter 3 presents a general approach for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and/or time, into some optimal and initially unknown configuration. Our approach relies on using embodiment and spatial embeddedness as a surrogate for computational resources, permitting the reduction or elimination of communication or shared memory for conventional parallel computation. Intermediate stages of the gradient descent process are manifested by the locations of the robots, instead of being represented symbolically.

We illustrate this approach by giving solutions to two non-trivial realistic optimization tasks from the robotics domain. This work is in the context of large-scale distributed systems such as animal colonies and multi-robot systems, which work together to solve complex tasks. We study two different versions of energy-efficient robot-robot rendezvous, useful for for recharging or refueling, or as a component of various other tasks. Chapter 4 addresses the problem of finding a single rendezvous point for a group of robots that minimized their locomotion costs of reaching that rendezvous point. Chapter 5 tackles the more difficult problem of finding an energy efficient joint motion plan for a group of worker robots that need to rendezvous with a single dedicated service robot.

Chapters 3-5 show how taking into account embodiment and spatial embeddedness can assist in running computations, described by continuous processes (or discretization of such) on a multi-robot system. Chapter 6 shows how a discrete prolem, usually solved by discrete algorithms can be solved by a multi-robot system that does not use a conventional discrete computer. It describes the first robotic system that solves a combinatorial computational problem by means of its own continuous dynamics.

The thesis concludes by summarizing our contribution and offering several directions for future work.

Chapter 2

Related work

2.1 Architectures

Despite the essential physicality of robots which are artificial agents designed to do valuable work in the real world, embodiment of robots was not in focus for a long time. Robots were designed to follow the traditional Sense-Plan-Act cycle inspired by the Sense-Think-Act of classical Artificial Intelligence where the thinking part was done by symbol manipulation (Newell and Simon, 1976). Such an approach had a very limited success because of the failure of robots to model complex world dynamics and thus operate outside of stringently structured environments.

A new paradigm was introduced by Brooks (1986) who presented the subsumption architecture for robot control based on a hierarchy of control layers running in parallel and influencing each other through a loose coupling. This work was in part based on the realization that complex behaviour does not necessarily require complex internal mechanisms and can be an interaction of a simple agent with a complex environment (Simon, 1969; Breitenberg, 1984). Successful early implementations of robots built according to the subsumption architecture (Brooks, 1990) boosted the research in behaviour-based robotics which nowadays is a part of the mainstream. Brooks (1990, 1991) suggested that the philosophy behind behaviour-based robotics should be applied to the phenomenon of intelligence in general. The author lists situatedness and embodiment as the key ideas behind behaviour-based robotics and new Artificial Intelligence. The first key idea, situatedness, can bring an agent a computational advantage:

> A situated agent must respond in a timely fashion to its inputs. Modeling the world completely under these conditions can be computationally challenging. But a world in

> which it is situated also provides some continuity to the agent. That continuity can be relied upon, so that the agent can use its perception of the world instead of an objective world model...*The world is its own best model.* (ibid., p. 583)

The author also advocates the use of deictic (Agre and Chapman, 1987) representations which express entities in terms of their relationships to an agent instead of representing entities in the world independently. For example, for a reader of current document a deictic representation of this document would be "the document I am reading" instead of its full citation. Deictic representations allow to "trade off computational depth for computational width" (ibid., p.583) reducing the computational path from sensors to actuators.

The second key idea, embodiment, is important for two reasons. First, only an embodied agent can demonstrate an intelligent behaviour by acting in the real world. Second, embodiment provides the means of physical grounding for the internal processing of the agent. Without physical grounding meaning does not exist for an agent. Also, embodiment introduces real world factors which are difficult or impossible to simulate. Some aspects of this realism can make design of the agent difficult (like aspects related to perception and action) while another can simplify it (e.g. symmetry breaking is often done "for free" in the real world).

As robot tasks get more complex it becomes increasingly difficult to create a robot controller without resorting to representations. Interaction between layers of subsumption architecture become too intricate to manage, especially when a robot can have many conflicting goals (Russell and Norvig, 2003, p. 933). Arkin and Mackenzie (1994) address limitations of subsumption architecture:

> Although this method has provided successful demonstrations of autonomous behaviour, it is lacking in its ability to incorporate world knowledge and alterations in user intent based upon changes in environmental circumstances and internal conditions. Even though early models of subsumption architecture did provide for the integration of world models, it is our position that a better approach involves a synthesis of hierarchical planning and reactive control.

The authors describe *hybrid architectures* that perform this synthesis by merging classical (deliberative) and behavioural (reactive) philosophies. An example of this architecture has three layers. At the highest level a mission planner module plans (by conventional means) the overall structure of the robot mission by decomposing the complex task into simpler subtasks. At the next level navigator plans generates a set of via points for the particular subtask. This is again done using conventional

planning techniques. Finally, a pilot works on a particular pair of via points selecting and tuning reactive behaviours from the behavioral repertoire of the robot. These behaviours perform the actual execution of the plan that was constructed by the previous (deliberative) levels of the control system. As execution unfolds, configuration and activity levels of concurrently acting behaviours change dynamically without resort to representations. Reactive execution level can trigger re-planning if it is not possible to reach current goal. New representational knowledge about the world becomes available to the deliberative levels during the plan execution, thus sensing feeds both lower reactive and higher deliberative modules.

Three-layer hybrid architecture or its modifications is employed by most modern robot control systems (Russell and Norvig, 2003, Ch. 25). In this architecture a reactive layer with tight sensor-action coupling and short decision cycle controls robot on the low level. A deliberate layer generates global plans to solve complex tasks. This layer uses models which can be predefined or learned from robot experience. Deliberate layer decision cycle is long (in the order of minutes) due to the complexity of planning. Executive layer interfaces reactive and deliberate layers. It transforms directives generated by deliberate layer into sequences of behaviours for the reactive layer. Executive layer also integrates sensor data and transforms it into representation with which deliberate layer can operate. The length of decision cycle of executive layer falls is in the order of seconds. Thus, three layer architecture bridges traditional planning ideas with the understanding of robot control based on embodiment.

On a more philosophical level Pfeifer (1996) lists embodiment along with situatedness, autonomy and self-sufficiency as the aspects of the "Complete agent principle" for design of autonomous agents. The author argues that only analysis and synthesis of complete agents can lead to understanding of cognition and intelligence. The author lists nine other principles and in later work (Pfeifer et al., 2005) capitalizes embodiment for all of them:

> However, there is a nontrivial meaning of embodiment, namely that there is a tight interplay between the physical and the information theoretic aspects of an agent. The design principles all directly or indirectly refer to this issue, but some focus specifically on it, such as the principle of sensory-motor coordination (embodied interaction with the environment induces sensory-motor patterns), the principle of cheap design (proper embodiment leads to simpler and more robust control), the redundancy principle (proper choice and positioning of sensors leads to robust behaviour), and the principle of ecological balance (capitalization of the relation between morphology, materials, and neural control). (ibid., p. 6)

Clark (1999) acknowledges importance of considering embodiment in cognitive science, but warns against radical embodiment stances that denounce internal representations, computation, functional decomposition and other traditional cognitive science notions (see Chemero (2009) for the recent presentation of this philosophy). Clark summarizes his reservations as follows:

> The major challenge for the vision of 'radical embodiment' described here lies with the class of 'representation-hungry' problems and the phenomena of off-line, abstract, and environmentally-decoupled reason. (Clark, 1999, p. 350)

In the field of machine consciousness that bridges robotics and cognitive science Holland (2007) advocates a strongly embodied approach to development of conscious machines, but finds attempts to avoid representations and deliberations wrong:

> The reaction to the perceived failure of these [logic-based] methods in robotics was behaviour-based robotics, which abandoned internal modelling altogether, and relied instead on sensing the environment directly; the baby of planning and deliberation went out with the bathwater of knowledge-based methods, and simulation as a general method, ubiquitous in engineering, was often decried in favour of working with 'real robots'. (ibid., p. 99)

Ziemke (2007) compares two different accounts of embodiment used in machine consciousness. Both accounts agree that an embodied agent not only should possess a physical body, but also interact with the environment through its sensory and motor system. The point of disagreement is whether the agent should also perceive its own homeostatic regulation. Sensorimotor approaches claim that an agent involved in complex sensorimotor interaction with its environment and endowed with the capabilities to model itself and its interactions with the environment can have conscious experience. An alternative view is that perception of the homeostatic regulation inside agent's body is necessary for having consciousness. This alternative view effectively prohibits non-living agents from having consciousness as they lack the complexity of autopoiesis (Varela et al., 1974) present in higher organisms.

In this thesis we follow a sensorimotor account of embodiment. Robots in all systems we consider actively interact with their environment (other team members) using their sensorimotor apparatus. This interaction becomes a crucial part of computation performed by the agents. We believe that high quality simulation of robot sensing and motion captures the essence of embodiment as seen by sensorimotor approaches. In such simulation a robot controller should be seamlessly transferrable between a simulated robot and a real one. This quality of simulations is easy to achieve in

our experiments where robot motion amounts to movement in two-dimensional space and sensing requires measuring relative bearing and distances to other robots. The usual drawbacks of simulations (inaccurate models and lack of noise in sensor data) are not critical for our simple systems. Therefore, we opt for using simulations in our experiments as they allow to perform large number of experiments and use team sizes that are difficult to achieve using real robots.

Summarizing this section, embodiment based approaches have enriched robotics and cognitive science dislodging traditional views but not replacing them.

2.2 Using a physical body to compute

2.2.1 Coevolution of morphology and control

Bongard and Paul (2001) show how morphological variability can play a crucial role in evolution of embodied agents. The authors conduct a set of experiments on the artificial evolution of a virtual biped agent. The agent has a waist and two upper and lower leg links. Hip joints have controlled roll and pitch while knee joints have only controlled pitch giving the agent six degrees of freedom with limited ranges. The agent is equipped with two haptic sensors in the feet and six proprioceptive sensors and torsional actuators for each degree of freedom. Control is performed by a three-layer neural network with a fully connected recurrent hidden layer. Network weights are optimized by a genetic algorithm favoring agents who can walk the furthest distance in a fixed direction subject to certain gait plausibility constraints. The authors compare two modes of artificial evolution. In the first mode only control network weights evolve, while in the second mode three morphological parameters are added to the genome: the radii of the lower legs, upper legs and waist thereby increasing the dimensionality of the search space by three. Contrary to the expectation that this increased dimensionality will degrade search, the second evolutionary mode consistently showed better performance. The authors suggest that this better performance is not explained by the evolution discovering superior morphologies, it is instead due to extradimensional bypasses (Conrad, 1990) which make it is easier to climb the fitness hill in a higher dimensional space:

> ...evolution of variable morphology agents does not perform better because evolution is able to discover a "good" morphology: rather, the addition of morphological parameters transforms the topology of the search space through which the evolving population moves, creating connections in the higher dimensional space between separated adaptive peaks in the lower dimensional space. (Bongard and Paul, 2001, p. 407)

The authors make a strong case for this explanation by tracking the evolution of the agent morphology. They found that at the end of an evolutionary run most fit individuals have morphologies identical or very close to the one with which the run was started. However, during the evolution there are periods where the morphology significantly departs from the initial value. This corresponds to use of an extradimensional bypass to get to an adaptive peak in the control parameters subspace. The authors also show that not every set of morphological parameters offers extradimensional bypasses. If instead of radii of the legs and waist evolution controls the placement and dimensional parameters of block masses attached to legs and waist, no gain from joint evolution of morphology and control is observed. The authors offer two potential explanations. First, agents with fixed blocks have dynamics which may make an adaptive landscape of control parameters more rugged than in the fixed length width case. Second, weight parameters are controlled by 8 variables versus 3 variables in the fixed width case which corresponds to a larger difference between dimensionalities of the control subspace and the joint control and morphology subspace.

2.2.2 Morphological computation

The phenomena observed in the work on morphological variability inspired the first work to explicitly connect computation and embodiment in robotics (Paul, 2004). The author introduced the concept of *morphological computation* by noting that robot morphology can "subsume a computational role" (ibid., p.33). The author describes a robot with two binary signal controlled actuators: $M1$ which is a rotating wheel which can propel the robot, and an arm $M2$ that can lift the wheel off the ground so the robot does not move even when the wheel rotates. The robot receives two binary inputs: A and B. A simple one-layer perceptron network calculates the disjunction $A \vee B$ and outputs the result to $M1$ so the wheel rotates if at least on of the two input bits is non-zero. Another one-layer perceptron network calculates the conjunction $A \wedge B$ and outputs the result to $M2$ so the wheel is lifted if none of the input bits is zero. A quick analysis of the behaviour of this robot shows, that the robot will move iff $A \oplus B = 1$. Therefore, the locomotion control of this robot (dubbed "XOR Robot" by the author) is a linearly inseparable "Exclusive OR" function which can not be performed by a single-layer perceptron. An interaction between moving components of the robot body (an arm and a wheel) performs some of the computation. The computation part performed by the robots morphology can be formulated as "the robot moves if the wheel rotates and the arm is not lifted", or formally $M1 \wedge \neg M2$:

This clearly shows that through its morphology, a robot body can perform a quantifiable

> computation which reduces the computation required in the controller[1]. (ibid., p.34)

The author describes another robot which has two binary signal controlled wheels each of which can propel the robot. In the same spirit, this "OR Robot" morphologically performs the disjunction of the control signals, since it moves iff. one of them is non-zero. Despite the simplicity of these examples, they indeed demonstrate explicit computations. The author shows how a part of a complex neural controller with feedback could be substituted by a morphological computation performed by the XOR Robot using a velocity sensor which can digitize the result of the morphological computation so it could be used further down the control schematic.

Another observation made in the paper connects motor and computational functions. The author states that a single action can serve both motor and computational purposes at the same time, thereby making possible the duality of functions in an embodied agent. A human counting a pile of coins on a desk while sliding them into the drawer one by one illustrates this concept. The duality here is in the coin sliding action which serves the computational function of separating counted coins from not counted and the physical function of relocating the coins to the drawer. Therefore, the counting here is done not so much *while* sliding, but more *by* sliding.

The following quote summarizes the findings of this paper which started the field of morphological computation:

> It has been known that interaction of loosely coupled sensorimotor processes with the dynamics of the body gives rise to *emergent behaviours* which are not explicitly represented in the controller... By showing here that the body can perform a computational role, the basis of such emergence can begin to be understood. In some cases at least it can be understood that the body provides the "computational glue" between loosely coupled sensorimotor processes. (ibid., p.38)

Pfeifer et al. (2006) discuss the relation of morphological computation and neural processing. The authors describe several biological and robotics examples where they claim morphological computation takes over some of the required control tasks and reduces the complexity of agent control. In the first system the morphology and neural controller of the *Eyebot* robot with multiple narrow direction visual sensors inspired by compound eyes of the insects is optimized by an evolutionary algorithm to keep the constant distance from an obstacle. Best-fitted exemplars show arrangements of sensors similar to the arrangement of ommatidia in flying insects: the density of cells is larger

[1] To be fair to neural networks we should note, that the morphological computation in this robot could have been traded for adding another layer with a single neuron into the controller (footnote by the author of this thesis)

at the frontal part of an agent than at the lateral. This allows the agent moving at constant speed to perform parallax compensation since a stationary point light source will "switch" between different ommatidia at a constant rate even though it will travel faster through the visual field when at the side of the agent than when at the front. If the arrangement of visual sensors is homogeneous, complex neural circuitry will be needed to perform parallax compensation, so the authors claim non-homogeneous arrangement described above performs morphological computation.

The second system described in the paper is a quadruped robot robustly performing fast locomotion with no sensory feedback. The robot has a rigid body with four legs with one servo motor for each leg placed at the shoulder. Each leg consists of two limbs connected by a passive elastic joint. The controller simply ensures sinusoidal positional oscillations of motors. Surprisingly, this robot is capable of maintaining several stable periodic gates without any feedback. The robot exploits its complex intrinsic dynamics and self-stabilizes in case of small perturbations due to very little friction on the feet. The locomotion control task here is said to be performed by a morphological computation which uses the robot morphology, the elasticity and stiffness of robot "muscles", the amplitude and frequency of motor actuations and environmental factors such as gravity, friction and shape of the ground surface.

Another example of morphological computation used for locomotion is the swimming fish robot *Wanda*. Having only one degree of freedom actuator (a waggling tail) the robot can swim forward, turn left, right, up and down. This is achieved by the proper use of the dynamics of interaction between the actuated elastic tail fin and water. The turning angle is controlled by the zero-point of the tail wiggling movement, while the wiggling frequency and amplitude control the robot speed. Slow speed makes the robot sink, fast speed combined with a turn makes the fish to move upwards. Again, the authors claim that in this robot the complexity of actuation and control is shifted to morphological computation.

The final robotic example described in the paper is the 13 degrees of freedom 5 finger Yokoi hand. Some parts of the hand are built from elastic, flexible and deformable materials: tendons are elastic while between-finger coupling and finger tips are deformable. If the hand is closed, fingers will automatically come together because of the anthropomorphic morphology. This allows solving a complex task of grasping with a simple control scheme which closes the hand. Properties of the hand will lead to the automatic self-adaption of the fingers position to the shape of the object the hand is grasping. In the absence of these properties the grasping task will require prior analysis of the object shape and actuation planning or using the feedback of the pressure and bending sensors or both. However, in authors opinion morphological computation takes over most of the complexity

of this task and allows the use of a simple and imprecise control scheme.

The computation performed by the morphology in all of these systems is identified much more vaguely than in systems described by Paul (2004). In the latter morphology performed a precisely specified transformation of the state variables that was a part of the computation required by the control system (e.g. disjunction in the "OR robot"). Therefore, a particular transformation is performed by the robot actuators instead of the conventional control system. In the systems described by Pfeifer et al. (2006) it is not clear what exactly is the information processing performed by the morphology. Indeed, the morphological choices described by the authors allow to use computationally simple control systems. However, an argument that complexity was absorbed by morphology and it is morphology that now *performs* a part of computation is not convincing. It would be possible, but similarly unnatural to claim that it is easier to control a three-wheeled bike than a two-wheeled version because a three-wheeled morphology computes stabilization control inputs required by a two-wheeler. Simpler dynamics of a three-wheeler seems to be a better explanation.

Nevertheless, in the examples presented by Pfeifer et al. (2006) complex agent behaviours are achieved by a combination of simple controllers and appropriate morphologies. The authors suggest to use the same approach for analysis of living neural systems. These systems can not be studied with no regard to the details of their embedding into an agent and the specifics of the agent interaction with the environment.

The authors finish the paper by raising an issue of the quantification of morphological computation. Since it seems that morphological computation has varying degrees of complexity in different agents, it would be useful to be able to measure the "amount" of computation performed by the morphology.

Lundh (2007) outlines an approach to such a quantification of morphological computation for the case of perception system. The author considers two information presentation processes, one that happens at the receptor site (sensor) where an external stimulus is fed into the perception system, another happens at the exit from the perception system where information is fed into the central computing unit (brain). Information presentation processes can be modeled in many ways, the only requirement is that there is some norm defined for a process (the author calls it "capacity norm"). The author defines a threshold, such that if capacity norm difference of two presentation processes at the entrance to the central computing unit is below this threshold, the system can not distinguish between these two processes. This threshold is inspired by the limit of conscious separation of different stimuli processes. The amount of morphological computation performed by the perception system can be seen as the amount of computation needed to be done on the information presentation

process at the input of the perception system to make it indistinguishable from the output of the perception system. In particular, the author suggests as a measure the minimum number of arithmetic operations needed to compute a functional that performs such an indistinguishable transformation of the input presentation process.

A different approach is suggested by Polani et al. (2006) who embraces information-theoretical view of embodied systems control. He suggests modeling the agent-environment interaction as a causal Bayesian network of perception-action cycles and to consider information flows that arise in such a system. The resulting quantification will take form of the family of probability distributions which could be further analyzed. The question of morphological computing precise meaning and its quantification does not yet have definite answers. It hardly seems that researchers will agree on some single quantification approach, the choice will rather depend on a particular application.

2.2.3 Computing by self-assembly

One of the lines of research relevant to using embodiment and spatial embeddedness is computing by self-assembly. In self-assembly systems multiple elements selectively form mechanical connections with each other based on the local information about the elements at the potential connection site. Therefore, a complex structure emerges from local interactions without centralized control. Research in this area owes a lot to the early work of Wang (1961) who studied a special case of the tiling problem using equal size square tiles with colored edges[2]. A question is to determine whether copies of a given set of tiles can be used to fill an infinite plane so that the colors of the adjacent tiles match. Wang (1961) conjectured that this problem is decidable and proposed an algorithm which was based on the flawed assumption that the only way to tile a plane is to do it periodically, that is with a repeating pattern. Later Wang (1963) proved that the problem is in fact undecidable by describing a construction of a set of tiles whose fitting in the plane corresponds to the evolution of a given Turing machine and input. For any machine a corresponding set of tiles fills the plane if and only if a Turing machine does not halt on that input. Such a reduction from a halting problem to a domino problem immediately implies undecidability of the latter. Berger (1966) confirms this result and also provides an aperiodic set of tiles which fills the plane without forming repeated pattern (i.e. there is no invariant under translation). Wang's undecidability result means that a system of assembling tiles is capable of Turing-universal computation. Universality of this sort inspired later work on self-assembly computing.

[2]Now these tiles are called *Wang tiles* or *Wang dominoes*

Adleman (1994) describes a computation of Hamiltonian path in a graph performed by a DNA molecules acted upon by standard biochemical protocols and enzymes. Nodes were encoded by DNA sequences and edges were encoded by oligonucleotides. The procedure amounted to creation of multiple copies of DNA molecules representing possible paths through the graph, selection of the paths which originate from the source node and terminate in the destination node, selection of the paths that enter exactly n vertices, and selection of the paths which visit every vertex at least once. Remaining molecules, if they exist, represent a sought Hamiltonian path. The paper reports a successful experiment with a 6 node, 14 edges graph. Due to the vast number of molecules available (the paper reports approximately 3×10^{13} copies of oligonucleotide for every edge) creation of Hamiltonian path molecule is very likely if such path exists. The author predicts difficulties with scaling of the DNA computation approach due to the increasing probability of erroneous bindings for larger graphs. However, because of the massive parallelism of DNA computing it can potentially outperform electronic supercomputers in number of operations per second. Another benefit comes from the superior energy efficiency of molecular computer in comparison with electronic machines which fall short of the theoretical thermodynamic efficiency limit by the order of 10^{10}. Nevertheless, the flexibility of electronic machines seems to be difficult to reach by one-dimensional DNA self-assembly.

Winfree et al. (1998) makes a step forward by offering a "two-dimensional crystalline forms of DNA that self-assemble from synthetic DNA double-crossover molecules"(ibid., p.1). Interactions between the structural units could be programmed allowing to form desired periodic patterns similar to the periodic Wang tiles. A way to create aperiodic crystals will make it possible to perform universal algorithmic computation with DNA molecules.

Winfree (1998) presents the Tile Assembly Model which extends the Wang tile model by introducing connection strength for every label of Wang tiles. A tile binds if and only if all the edge labels match the labels of the neighbors and the total connection strength of all binding edges exceeds a certain threshold. Connection strength models the strength of molecular binding while connection threshold models the reaction temperature. Hence, molecules bind if the binding strength overcomes thermal dissociation. Computation is performed by starting with a seed configuration which may consist of a single tile and allow the system to grow by connecting matching tiles. The Tile Assembly Model preserves Turing-universality of the Wang tiling simultaneously allowing to prevent dead-end growing sequences by proper use of the connection strength. Winfree (2003) mentions three-dimensional periodic DNA tiling as an open problem and states that a high error rate of DNA self-assembly is the major obstacle for executing complex algorithms. Therefore, developing error

correction schemes seems to be of a great importance to the field.

A notion of embodiment could help to attribute the computing power of self-assembly systems to the particular elements and processes. A current stage of a computation process, including the input is embodied in the arrangement of the assembling elements. The computation process is discrete in that bonding or removal of a single element could be seen as a computation step. A program of such a computer is embodied in a mechanism responsible for selective bonding based on local properties of the environment. Note, that this mechanism includes means of delivery of assembly elements to bonding sites. Depending on a system this could vary from thermal movement of molecules in a chemical reaction to self-actuated locomotion of elements in self-assembling robot systems (Hamann and Wörn, 2007).

2.2.4 Collision based computing

In a seminal paper Fredkin and Toffoli (1982) describe *reversible* and *conservative* computation. Traditional models of computation operate with non-bijective functions, effectively losing information during computation. For example, conjunction $z = x \wedge y$ does not allow recovery of the values of x and y from z after the operation is performed. Thus, a device which realizes such a conjunction so that the value of even one of the operands is lost is an irreversible device. After the operation is performed there is less information in the system than before. Thermodynamics laws dictate that an erasure of information should be accompanied by the thermalization of the system. Hence, this necessary energy dissipation constitutes a persistent obstacle on the way of increasing computer performance and decreasing its size. However, the authors stress, irreversibility is only a property of macroscopic physical processes, while microscopic processes are reversible. Their main message is that computation does not require irreversibility and can be based on invertible physics. The authors offer a logical model of a conservative and reversible circuit based on two primitives: the *Unit wire* and the *Fredkin gate*.

The Unit wire is a "storage-transmission primitive whose intuitive role is to move one bit of information from one point of space-time to another separated by one unit of time". The authors argue, that signal storage and signal transmission are a single phenomenon whose interpretation depends on the reference frame. For example, in a reference frame attached to a Christmas card "transmission" from Canada to Ukraine the card serves as a "storage" since information does not change its spatial position. Likewise, in the example given by the authors a note left on the desk to serve as a "storage" and be read next morning will travel large distance from the point of view of

an observer at rest with the solar system as a whole effectively becoming a "transmission" for this observer. This vision is important for systems where transmission of information between different elements of the system is substituted for the direct access of these elements to the information stored in the configuration of the system (Litus and Vaughan, 2008).

The Fredkin gate is a Boolean $\{0,1\}^3 \rightarrow \{0,1\}^3$ function working in the following way. One of the inputs serves as a control bit and two as data bit. Control bit is not changed by the function, i.e. the output value of the control bit always equals the input value. If the control bit is 1 then two data bits are unchanged, otherwise the values of the data bits are swapped at the output. Therefore, a Fredkin gate does not change the signals, it simply reroutes them. In this sense it is a conservative transformation, the number of the raised bits at the input always equal the number of the raised bits at the output. It is also clear that this function is a bijection (the inverse of the function is the function itself) making this transformation invertible. The Fredkin gate is functionally complete, i.e. any logical function could be represented as a composition of multiple Fredkin gates. Thus, a conditional rerouting of signals performed by the Fredkin gate combined with Unit wire signal storage/transmission and delay provided by Unit wire give sufficient building blocks for a Turing-universal computer. The authors give an example of one possible physical implementation for such a device: a Billiard Ball model. This idealized system is comprised of a set of identical balls and a set of reflector walls which interact in perfectly elastic collisions. The presence of the ball at a certain location corresponds to binary 1 while its absence corresponds to binary 0:

> Intuitively, we show that by giving the container a suitable shape (which corresponds to the computer's *hardware*), and the balls suitable initial conditions (which corresponds to the *software* - program and input data), one can carry out any specified computation. (Fredkin and Toffoli, 1982, p. 239)

The billiard ball model serves as a proof of concept for conservative computing. Though it presents an interesting example of unconventional computing, it is impractical. Likewise, there are no cost-effective implementations of reversible or nearly reversible logic devices available at the moment.

The Billiard Ball model inspired research in so-called collision-based computing (Adamatzky, 2002) which does not necessarily preserve the original ideas of reversibility and conservation. A recent work describes a computing system based on the collisions of the wave fragments in the reaction-diffusion Belousov-Zhabotinsky chemical medium (Adamatzky, 2004). The Belousov-Zhabotinsky (BZ) reaction is a chemical reactions which does not reach thermodynamic equilibrium for a long time. Instead, the BZ reaction is a non-linear chemical oscillator which shows periodic

changes in the concentration of certain ions in the presence of bromate oxidant. Chemical concentration waves can propagate through the reaction medium. These waves could be initiated by local excitation in a thin layer reaction solutions. Adamatzky uses a photosensitive BZ medium as a computing system by interpreting the traveling waves initiated by local application of light as traveling particles similar to balls in the Billiard Ball model. He achieves non-conservative universal computation in a realistic simulation of BZ reaction by implementing logical interaction gates by collision of traveling concentration waves. Since the BZ medium does not have any preexisting structure the author calls this computing system "architectureless". A later work Asai et al. (2005) describes a silicon electronic analog of a reaction-diffusion processor that computes a Voronoi diagram while Adamatzky et al. (2004) presents a robot controlled by a BZ chemical medium.

2.2.5 Computing in multi-agent systems

Research on morphological computation discussed above demonstrates the ways for a single robot to use embodiment for computing. They involve computation processes performed by the parts of the physical body of an agent interacting with each other and with environment. Coupling between bodies of the agents in multi-robot system is much weaker than between body parts of the single agent. The absence of the spatial continuity in the multi-agent system makes it conceptually different from the single agent. Can the idea of using sensors and actuators to compute be applied to the multi-agent case?

Hamann and Wörn (2007, 2008) present two relevant systems. The first gives an approximated solution to Euclidean Steiner Tree Problem (Hwang et al., 1992): for a given set of points (nodes) on the Euclidean plane find an interconnecting network with minimum total length of the edges possibly including additional nodes to decrease the total length. These additional nodes are called Steiner points. In the proposed embodied solution a set of immobile seeds is initially placed on the plane at the points to be connected by a tree. Then a heterogeneous group of robots with local sensing capabilities move randomly on the plane connecting to seeds and already connected robots following a set of locally defined rules. At a certain point in this attachment process a second stage is triggered and the robots which are connected to only one robot disconnect and leave. This effectively leaves connected only the robots which form paths between seeds. At the end of the disconnection process every robot has a connection degree of two or three. The robots which have a degree of three represent the Steiner points in the solution. In the final stage connections between seeds and Steiner points are straightened by releasing the robots between them which gives the approximation

to the sought Steiner tree.

The second application is a variation on the Majority Problem (also called the Density Classification Task) initially proposed in the field of cellular automata (Gács et al., 1978). The task is to perform a majority voting in a one-dimensional cellular automaton, i.e. a set of two-state cells arranged in an array should assume the state of the majority of cells in the set basing transitions only on the states of the cells in the local neighborhood. In the system proposed by the authors a set of two-state robots moving randomly in a bounded environment needs to assume the state of the majority. The robots engage in collision-avoidance behaviour and after five encounters with other robots assume the state of the majority of the robots met. The authors report an error rate of 18 % for the initial state ratios in the most difficult range of 0.5 ± 0.05 independent of the number of robots for numbers above 100.

Payton et al. (2001) introduces a concept of "world embedded computation" and describe a robot swarm system that runs an analogue of Dijkstra's shortest path algorithm in a world embedded manner. Robots spread themselves in the environment maintaining connected visibility graph and then find the shortest path between two points by exchanging local messages. The path is delivered to a user of the system by every robot turning towards the next robot on the shortest path to the goal and displaying a visible arrows. The user then follows these arrows towards the goal starting from any point from which a robot is visible. The system is robust in a sense that if some number of robots is destroyed, the remaining swarm members rearrange themselves so to restore connectivity of the visibility graph. A more conventional approach would be to build a map of environment and then perform shortest path search using that map. In Payton's system robot manifest locations in the world and steps of Dijkstra algorithm are executed through local message exchange, thereby embedding computation in the world.

2.3 Robot path planning

The main distinction of robot path planning (motion planning) from most other forms of planning is the continuous state space in which robots operate (LaValle, 2006, Part 2). Execution of a plan generates a continuous path in the *configuration space* of the robot (the set of states which robot can assume) (Lozano-Perez, 1983). Configuration space can be restricted due to the presence of static or moving obstacles and the mechanical properties of the robot. In case of a multi-robot team every robot creates a mobile obstacle for the rest of the team.

The traditional approach to motion planning is to discretize the configuration space and to apply discrete planning algorithms to the resulting problem. Two classes of methods can be used to discretize the configuration space: sampling based methods and combinatorial methods.

2.3.1 Combinatorial methods

If the configuration space has low dimensionality it may be possible to use combinatorial methods that use exact representation of the problem. For example, cell decomposition methods follow this approach by discretizing configuration space into a number of continuous regions (cells) (Brooks and Lozano-Perez, 1983). Decomposition is performed in such a way that path planning inside every cell is trivial. Therefore, the initial problem is reduced to the discrete graph search problem. Skeletonization methods reduce dimensionality of the problem by creating a one-dimension representation of the free configuration space. One of such representations is Voronoi graph (Choset, 1996). Path planning in one-dimensional representation can also be solved as a discrete planning problem. Combinatorial methods are complete as they always find a plan if it exists or correctly decide that the problem has no solution. However, they are computationally expensive and quickly become infeasible as the dimensionality of the problem grows.

Probably the most famous implementations of the classical approach to motion planning was Shakey the robot, developed at Stanford Research Institute in 1966-1972 (Fikes et al., 1972; Nilsson, 1984). The robot used a logical model of its world and robot actions expressed in the formal language based on a predicate calculus. The high-level action plan was generated using the STRIPS planner (Fikes and Nilsson, 1971). Lower level motion planning was done by discretizing the world with a finite grid and building a visibility graph on that grid.

2.3.2 Sampling-based methods

Sampling-based motion planning methods avoid construction of the explicit and complete problem representation. Instead the search is performed by sampling configuration space. These methods sacrifice deterministic completeness for speed, but in practice allow to solve high dimensional problems for which combinatorial planning methods are prohibitively slow. Various solutions are used to generate and use samples. Randomized potential fields methods (Latombe, 1991) introduces a potential function which provides an estimate of the distance to the goal configuration by combining attractive term (metric distance to the goal) and repulsive term (penalizing proximity to obstacles). This values of the potential function are used to create a gradient field in the configuration space in

which a variant of a gradient descent with a backtrack is performed. This method can escape local minima, but has a lot of parameters that need to be tuned.

The Ariadne's clew algorithm (Mazer et al., 1993) complements search with an exploration step in which an addition of a new configuration to the search graph is attempted. The new configuration should be easily connected to the randomly selected vertex of the graph and should be globally optimal with respect to the distances to the other vertices of the graph. Genetic algorithms that require problem-specific parameters tuning are used to solve this difficult optimization problem. A similar method that uses an exploration step is described by Snchez and Latombe (2001). This method is based on bidirectional search and improves the planning efficiency, however still suffers from the need for problem-specific tuning.

Several sampling-based methods that do not require parameter tuning are based on the idea of rapidly exploring random trees (Lavalle and Kuffner, 2000). A search tree is constructed incrementally with gradually increasing resolution. The limit of this construction densely covers the search space. Unlike space filling curves the tree arranges many short paths instead of a single long path. Bidirectional search ideas can be used simultaneously constructing two trees rooted in the initial and goal configurations (Kuffner and LaValle, 2005). Some methods build more than two trees at the same time in order to parallelize exploration of problematic (narrow) regions (Strandberg, 2004).

All sampling-based methods mentioned so far are aimed at finding a plan for a single planning query (initial-final configuration pair). For the case where multiple planning queries are expected roadmap methods use the preprocessing stage to create a topological graph (roadmap) that facilitates quick planning for future initial-goal queries (Kavraki et al., 1996). Visibility roadmap (Simeon et al., 2000), which attempts to create a small graph with a good coverage is one of the most popular roadmap methods.

2.3.3 Feedback planning

The open loop solutions described above may face difficulties in real world applications for several reasons. Imperfect information about the world leads to errors in configuration space representation or samples that may result in a plan that is impossible to execute. As a special case, the initial configuration may not be known. Also, the robot may deviate from the planned trajectory during the plan execution and some means of returning to the trajectory or replanning may be necessary. In a dynamic environment future world state is difficult to predict, so by the time the robot reaches a certain point on the planned trajectory changes in the world may render the original plan wrong

even if robot was following the plan precisely. Decision-theoretical planning attempts to explicitly model uncertainty of world states (Boutilier et al., 1999) and is not discussed here.

Another way to cope with uncertainty is to include feedback mechanisms into planning thereby allowing robot to select an action in various states in which it finds itself. A common approach is to define a potential function that assigns a potential value to every state based on its distance to the destination state and other properties (e.g. proximity to the obstacles). A potential function that is defined for every state is called a navigation function. Planning can be performed by finding a path that follows the direction of the fastest decrease of navigation function.

These ideas were presented to robotics most notably by Khatib (1986). The goal of his approach is to shift some burden of motion planning from the slow higher levels of control hierarchy to the faster lower levels thus improving real-time navigation. The author used manipulator collision avoidance to present his approach and summarized his ideas as follows:

> The manipulator moves in a field of forces. The position to be reached is an attractive pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.

Therefore, a controlled robot can be viewed as a particle moving in an artificial potential field that exerts forces that can be computed as antigradients of the potentials. Thus a robot performs a gradient descent in this field as it moves towards the destination. In such a system high level control is responsible for providing a set of intermediate goals instead of a complete collision-free path. Low level control performs obstacle avoidance in real time as the data from sensors arrive during the robot motion. The ideas of artificial potential fields are further developed in Rimon and Koditschek (1992).

Koren et al. (1991) identified a set of practical limitations of potential field methods: cyclic behaviour due to local minima; inability to pass through a narrow passage between close obstacles; and oscillatory behaviour in the obstacle proximity. The authors claim that the main reason behind these shortcomings is excessive data reduction during the summation of repulsive forces from various obstacles. They suggest an alternative "Vector Field Histogram" method (Borenstein and Koren, 1991; Ulrich and Borenstein, 1998) that incorporates more information about local obstacles by building a one-dimensional polar histogram of obstacle density in different sectors around the current position of the robot. Sectors with density values below certain threshold comprise the possible candidate directions from which one is chosen based on the target direction. Speed is selected based on the obstacle proximity so that robot moves slow near obstacles. Special care is taken to detect cyclic behaviour, however once detected, the problem is delegated to the global planner that comes up with

a new set of via points based on the available obstacle information.

Though technically different from potential field methods, vector field histogram methods follow the same philosophy of not generating a complete motion plan in advance and instead generating a set of via points and performing local reactive planning as robot follows through this set. Such an approach may even violate some stringent definitions of planning as a set of intended actions to achieve the goal is not available before the execution of plan starts.

It is interesting to consider how far it is possible to take the idea of reactive planning. Mataric (1997) argues that purely reactive (stateless) control systems are applicable to problems that allow complete specifications at design-time. This limits the flexibility of a purely reactive system as it is incapable of storing any information during execution time. Schoppers (1987) describes a purely reactive control system based on a decision tree generated during pre-processing time. Such a tree accounts for all states a robot might find itself in while navigating to a particular goal thus making initial state irrelevant. Due to poor scaling this approach is impractical for complex worlds and robots. Mataric (1990, 1994) slightly departs from pure reactivity when describing robot Toto with a behaviour-based control system capable of navigation, map learning, and path-finding in an unknown environment. The robot uses internal representations to store information about the world as it learns it, however instead of static representations used in traditional planning system, the map is created as an integral dynamic part of a robot behaviour network. The authors calls these procedural and behavioral representations *active representations*. Path planning is performed by spreading activation from the selected goal behaviour through all of the nodes in the behavioral network. During activation spreading a shortest path from the current map behaviour (position) to the goal is found and a sequence of actions to reach the goal is performed by spreading activation from the current map behaviour towards the goal.

2.3.4 Multi-robot planning

If a group of robots operates in the same environment, every robot acts as an obstacle for every other member of the team. Path planning for multiple robots has to take this phenomenon of *spatial interference* into account to avoid collisions between team members.

It is possible to treat the whole team as a single robot with multiple mechanically decoupled sensors and actuators. In this case combinatorial and sampling-based methods described above are applicable. However, the dimensionality of configuration space for such a "robot" will grow linearly with the size of the robot team. Therefore, combinatorial methods scale poorly as their complexity

is exponential in space dimensionality. Better performance can be expected from sampling-based methods, but they also become overwhelmed as the team size grows. Therefore, special methods for multi robot planning should be considered

Decoupled planing solves the problem in two stages. First, a (partial) plan is produced for every robot independently ignoring spatial interference with other robots. Second, the interaction between plans is considered to resolve conflicts and create a joint collision-free motion plan for the whole team. In prioritized planning Erdmann and Lozano-Perez (1986); van den Berg and Overmars (2005) an order is imposed on a robot team and path planning proceed sequentially starting with a robot with the highest priority. A plan is produced for this robot ignoring rest of the team. Plans for lower priority robots are produced by treating the higher priority robots as moving obstacles with known trajectories. This approach is not complete as the plans for higher priority robots can not be revised even if they make the goal unreachable for lower priority robots.

A more powerful approach is proposed by (O'Donnell and Lozano-Pérez, 1989). Again, the path of every robot is completed independently ignoring the spatial interference. Then, a joint *coordination space* is considered where every dimension describes a point on the path of the particular robot. Sets of points in the coordination space where robots collide are treated as obstacles. In such a way the problem of trajectory coordination can be solved as a scheduling problem. Robots that have to pass through the same part of the space are separated in time. This is achieved by planning a collision-free path in the coordination space from the initial to goal configuration by conventional search methods.

Reactive planning philosophy can be applied to collision avoidance. Robots ignore other members of the team until they detect that a collision is imminent. Then a local collision avoidance manoeuvre is executed by the robots that are about to collide. Once spatial interference is resolved, robots continue working ignoring other team members.

The reactive approach serves as a basis for models of flocking behavior (Reynolds, 1987; Olfati-Saber, 2006; Chazelle, 2009). These models are based on three rules: separation (repulsion from neighbours which are too close), alignment (matching speed and direction with the average of neighbours), and cohesion (moving towards the average position of the neighbours). Flocking models are similar in spirit with the algorithms elaborated in this thesis as all of them achieve complex global dynamics using relative state sensing. However, while our algorithms solve precisely specified computation problems by means of their system dynamics, the goal of flocking algorithms is the dynamics with certain properties per se. This places flocking algorithms and embodiment based multi-robot computation into separate domains.

Vaughan et al. (2000) describe a biologically inspired interference resolution method which is capable to handle challenging one-robot-size gaps scenarios. In these scenarios robots attempt to simultaneously pass through a narrow opening which does not have space for a drive-around collision avoidance. The algorithm is based on aggressive displays performed by animals. Once there is a need to avoid collision, robots randomly chooses an *aggression level*. These levels are compared through stereotypic display behaviour and the robot with highest aggression level uses the space in question first while lower aggression robots back up to let the winner through. No communication between robots is necessary.

Berg et al. (2008) follow a more formal approach and define a reactive method that guarantees oscillation-free collision avoidance. Every robot senses the relative velocities and shapes of robots in close proximity and computes a *combined reciprocal velocity obstacle* - a set of velocities which will lead to a collision with one of other robots. A robot then chooses a velocity vector which lies outside of this velocity obstacle (if possible). The choice attempts to simultaneously minimize the departure from the desired velocity vector and maximize the time to collision in cases where velocity obstacle includes all possible vectors. An extension of this method allows to optimize the sets of possible collision-avoiding velocities (van den Berg et al., 2009).

2.4 Summary

The body of literature reviewed in this chapter shows that promotion of physical body from the mere source and sink of data for a dedicated computing module to an active participant in information processing led to advances in robot control systems. We cited works that use these advances for single robots. We also described several multi-robot systems that employ embodiment to solve combinatorial problems using discrete algorithms.

The following chapters further these ideas in a novel way by solving continuous and combinatorial problems using embodiment, spatial embeddedness and continuous dynamics of multi-robot systems. We begin by describing a general approach for developing decentralized distributed gradient descent optimization algorithms for teams of embodied agents that need to rearrange their configuration over space and/or time, into some optimal and initially unknown configuration.

Chapter 3

Distributed gradient optimization with embodied approximation

3.1 Introduction and related work

In Section 2.1 we reviewed Brooks' approach to architecture of intelligent agents. This approach attempts to avoid internalizing the world but rather to sense and react to it directly whenever possible. This chapter makes explicit an inversion of this approach, in which where the world is used to externalize a computational process (Litus and Vaughan, 2008).

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

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

approximately in the direction of the local antigradient of the global cost function.

Gradient-based formation control and navigation have been widely used in robotics (Zelek, 1999; Tanner and Kumar, 2005). The idea of embodied computation was recently presented in (Hamann and Wörn, 2007), however contrary to the authors' claim we believe that some globally defined algorithms can be implemented by embodied computation. Our approach is related to the "information surfing" technique described by Bourgault et al. (2002). Finally, Loizou and Kumar (2007) have shown biologically plausible navigation and tracking algorithms which involve using other agents locations to calculate a local gradient. To our knowledge, this chapter is the first to explicitly present embodied approximation as a method to implement parallel gradient optimization algorithms.

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

3.2 Distributed gradient optimization with embodied approximation

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

If the function J has certain mathematical properties (e.g. is continuously differentiable) then a good approximation to an optimal configuration can be found in a centralized way by using one of many gradient optimization algorithms. It will start with some initial approximation and iteratively change it in constant or decreasing steps in the direction of (approximated) antigradient. Details of

these particular algorithms are irrelevant for the present discussion. This centralized solution will need a central processor with amount of memory of the order of the size of the state vector s as well as means to communicate s^* to all agents once an acceptable approximation is found. Once all agents know their component of s^* they can proceed directly toward it. If J is parameterized by initial configuration of agents S_0 then the means of communicating or sensing this configuration by the central processor are required.

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

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

Algorithm 1 Gradient optimization with embodied approximation

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

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

This approach makes the agents themselves serve as a shared memory for the parallel computation they perform where the information about the current approximation is embodied in physical configuration of the agents. Indeed, an agent sensing configurations of another robot corresponds to the read operation in conventional shared memory. Changing the configuration corresponds to the write operation to the conventional shared memory. Note that this world-embedded memory does not require the access synchronizations of conventional shared memory as the configuration of one agent can be simultaneously sensed by several agents while that agent is moving. Once the values of necessary components are available, gradient direction ∇J_i or an approximation to it can be calculated and agent can move in the antigradient direction thus improving the approximation of component s_i and global state s. Agent relocation becomes a part of a computation process, creating a parallel computer comprised of the agent's processors and the physical bodies of agents. Computation is performed both inside the agent processors (computing movement directions) and outside of processors in the environment in which agents are embedded (changing spatial configurations). The algorithm continues until convergence is reached which is detected in a way specific for the particular problem and algorithm employed.

3.3 Discussion

A general approach for development of decentralized distributed gradient descent optimization algorithms for teams of embodied agents is presented. This approach uses embodiment and spatial embedding as computational resources which allow us to reduce or eliminate communication or shared memory requirements for parallel computation. Spatial configuration of agents serves as an embodied representation of the current approximation to the global solution which is accessed by means of sensing and refined by agents moving along local antigradient directions.

Admittedly the extent to which embodied approximation can substitute communication or shared memory is limited by the properties of sensors and environment. Limited sensor range and occlusions will limit the number of relevant state vector components that could be accessed by sensing. If this can not be remedied by communication, agents will not have enough data to compute their gradient step directions. Attempts to ignore this issue by using old values of unaccessible state vector components may lead to wrong results. Depending on the particular scenario, convergence to the extremum can be delayed or even prevented either due to oscillations or convergence to the wrong point. To avoid this, at least sporadic access to up-to-date information through sensing or communication is necessary.

Lack of access to relevant components will not be an issue in optimization problems that by their nature need local and limited exchange of information. For other problems some communication is necessary to run distributed optimization algorithms, but embodied approximation may still significantly reduce the need for inter-agent communication and should be considered as one of the available resources.

Sensor noise presents another obstacle as it will distort the data used in calculation of antigradient directions. The influence of this issue on the convergence will depend on the particular cost function and the nature of the noise. Conventional noise management techniques can be used in cases where noise degrades performance of the system.

In the following chapter we will use the embodied approximation approach to address two nontrivial optimization tasks of energy-efficient path planning for mutli-robot teams. The resulting distributed algorithms show good results in experimental evaluation. These algorithms use very simple mechanisms that result in energy efficient group behaviour. In both cases, computationally complex optimization tasks are solved using bearing-only sensing that we suggest is biologically affordable.

Chapter 4

Rendezvous

The work elaborated in this chapter is based on the work by Zebrowski et al. (2007), presented at the Fourth Canadian Conference on Computer and Robot Vision in May 2007. Y. Litus contributed to this work by performing mathematical analysis of the problem, researching related literature and suggesting local heuristic methods.

4.1 Introduction

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

If fuel is a precious resource, as is common in real world applications, then an important measure of system efficiency is the ratio of fuel expended by the workers in doing work to the total energy expended by all robots. In this chapter we examine the problem of finding a single best meeting location given the locations and movement costs of all robots. Two methods, *Global* and *Local* are described and their properties compared. Local method is an application of the embodied approximation approach described in the previous chapter. Both methods are tested in simulation. The experimental setup which is used in the majority of our experiments is discussed.

The rendezvous problem is a natural beginning for tanker based recharging. It is straightforward to implement and allows all robots to meet in order to achieve recharging. The problem also has the benefit of providing a tankerless recharging benchmark. Consider n robots in some environment

and some optimal 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.

4.2 Problem Characterization and related work

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

$$p^* = \arg\min_p \sum_{i=1}^{n} c_i||p - r_i|| \tag{4.1}$$

Problem (4.1) is known very well 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 (Gueron and Tessler, 2002) for the case $n = 3$ and (Kupitz and Martini, 1997) for the general case). Effective numerical algorithms exist (Rosen and Xue, 1991). Interestingly, it is also possible to describe the solution using a mechanical interpretation as a system of idealized strings, pulleys and weights (Polya, 1968).

We will now briefly state the properties of the solution point. First, p^* obviously can not 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 (4.1) is strictly convex, which ensures the uniqueness of p^* if $n \geq 3$. Finally, it is possible to prove the following theorem (Kupitz and Martini, 1997).

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

$$||\sum_{j=1}^{n} c_j \frac{r_j - r_i}{||r_j - r_i||}|| > r_i, i \neq j \tag{4.2}$$

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 (4.2) 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.

4.3 Solutions

4.3.1 Centralized (global) solution

Since in our setting robots know each other's locations, the straightforward way to rendezvous is for each robot to find p^* using a numerical method and then move towards it. Alternatively, if communication is possible, one robot can calculate p^* and broadcast it 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 the next section we propose a fast local heuristic method which allow 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.

4.3.2 Local solutions

In this heuristic *local* method, each robot's physical trajectory approximates a gradient descent towards point p^* on the total movement cost function adaptive 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 illustrates the ideas of using embodied approximation for distributed gradient descent presented in the previous chapter.

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

$$\vec{d}(a,b) = \frac{b-a}{||b-a||} \tag{4.3}$$

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 $\vec{D} = \sum_{i|x \neq r_i} c_i \vec{d}(x, r_i)$ (the sum of weighted unit vectors).
3. If $x = r_i$ for some i (the robot is at some robot's original location), set $c = c_i$, otherwise set $c = 0$.
4. If $||\vec{D}|| < c$ then stop. Otherwise proceed in the direction $\vec{D}$.
5. Goto 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 $\vec{D}_j = \sum_{i \notin A} c_i \vec{d}(r_j, r_i)$.
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. Goto 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). This follows from the fact that vectors D and D_j point in the direction of the conjugate gradient of the total energy cost function. Thus, the current positions of the robots serve as current solution approximations in a first order numerical gradient descent optimization of the cost function. The absorbed case is accounted for by the stopping condition. Convergence of the first order method is proven in Wang (1975). We do not expect much gain in calculating the movement vector using second order approximations and believe that more complex calculations will not yield worthwhile reductions in total trajectory length. However, exploring this option may be one further direction of this research.

The local approach has important benefits in comparison with 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).
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.

4.4 Experiments

4.4.1 Setup

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.

World and Simplifying Assumptions Experiments are performed in simulation using the well-known Player/Stage robot control and simulation system (Vaughan, 2008). 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-200 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 intentionally 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 (Zuluaga and Vaughan, 2005), 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.

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 d. In these experiments d 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

$$E = \sum_{i=1}^{n} c_i d_i \tag{4.4}$$

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

Controller Implementation The experiment depends on two controllers: a robot controller and a processing controller. Communication between controllers is implemented with UDP datagrams. The next sections describe the controllers.

Low-Level Robot Controller 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 (Borenstein and Koren, 1991).

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 4.3.1. 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 4.1. This method achieves this using the Nelder-Mead algorithm (Mathews and Fink, 2004). Nelder-Mead returns the minimum cost location p^*. The processing controller then prescribes this as the location each robot should go to. In the dynamic variant p^* is recomputed periodically using the latest robot positions. In the static variant, the minimum cost location p^* is computed exactly once.

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.

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.

Procedure Each of the six experiments is performed with seven distinct initial configurations illustrated in Figure 4.1. 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 start locations but different weights. 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. 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.

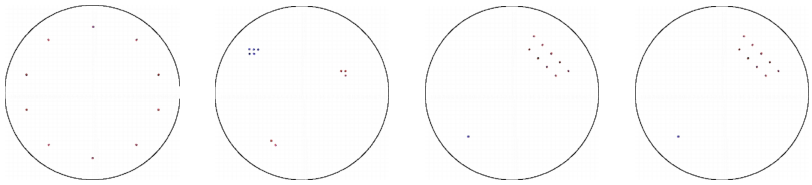
4.4.2 Results

Paths Traversed

Figure 4.2 shows some example paths taken during the experiments. In each case, the global methods result in robots heading directly for the meeting location (Figure 4.2(a), 4.2(d)) while the local methods tend to take a curved path (Figure 4.2(b), 4.2(e)) as predicted in Section 4.3.2.

In the absence of obstacles (and spatial interference), all methods achieve rendezvous 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 4.2.

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

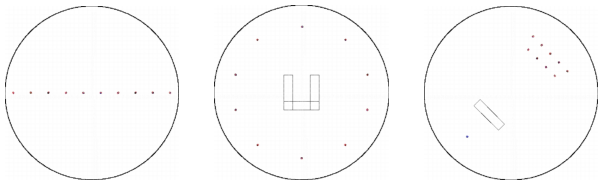


(a) Map 1 - All robots equal weight

(b) Map 2 - All robots equal weight

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

(d) Map 4 - Single robot weight = 1, all others weight = 10

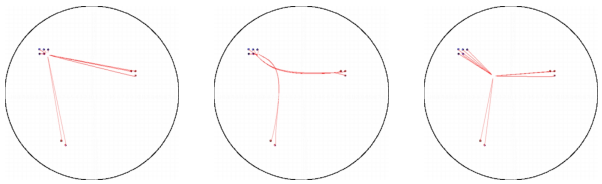


(e) Map 5 - All robots equal weight

(f) Map 6 - All robots equal weight

(g) Map 7 - All robots equal weight

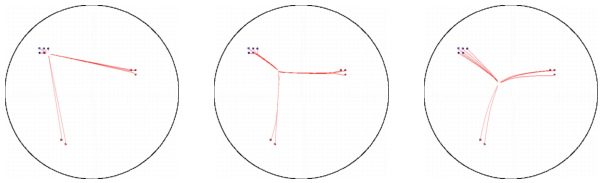
Figure 4.1: Experimental initial conditions



(a) Global static

(b) Local static

(c) Center-of-mass static



(d) Global dynamic

(e) Local dynamic

(f) Center-of-mass dynamic

Figure 4.2: Typical paths taken (Map 2)

Table 4.1: 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

Table 4.2: 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	524.22	600.02

Energy Used

Table 4.1 shows the mean total energy used for each method/map combination. In each case without obstacles (Map 1-5) the global methods use on average 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 on average well, otherwise it performs poorly compared to the other methods.

4.5 Discussion

4.5.1 Paths Traversed

Figure 4.2 shows typical paths taken for each method using Map 2 (Figure 4.1(b)). Figures 4.2(a) and 4.2(d) show the robots meeting at approximately the optimal location. Figure 4.2(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. Application will dictate the choice between longer path and longer computation time.

Figure 4.2(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 4.2(f)) this is more pronounced, as the meeting point moves further away from the optimal location.

4.5.2 Static Versus Dynamic

The effect of dynamic updating can be seen clearly by comparing Figures 4.2(b) and 4.2(e). Updating position information results in robots "sticking" together when they meet (Figure 4.2(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 4.2(b) and 4.2(e) dynamic updating achieves this by shortening the curved path caused by the local method.

Dynamically updating robot positions can act as a benefit or hindrance. 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.

4.5.3 Global Versus Local

The global method finds good rendezvous locations at the cost of longer computation time. The local method is computationally cheaper (as there is no need to compute a complete solution), 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 could be preferrable as it avoids computing a complete solution before moving starts. 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.

Chapter 5

Frugal Feeding Problem

The work elaborated in this chapter is based on the work by Litus et al. (2007) presented at the 2007 IEEE International Conference on Robotics and Automation and the work by Litus et al. (2009) published at the IEEE Transactions on Robotics, Feb. 2009. All work in this chapter was done by Y. Litus with the exception of running Stage experiments described in Section 5.6.1 (Y. Litus provided the heuristic controller code for these experiments), and generating some of the figures.

5.1 Problem description and characterization

In this chapter we will relax a single-place rendezvous assumptions used in the previous chapter. We still seek to minimize the total amount of fuel spent driving robots to refuelling rendezvous, so that the fuel available for useful work is maximized. However, now we allow different worker robots to meet with the tanker robots in different locations.

The problem can be formulated as follows: given a set of original locations of worker robots and tanker, find the set of meeting points such that the tanker meets every worker and that minimizes the total energy spent on locomotion. Alternative objective functions include minimizing total rendezvous time, or a trade-off between energy and time cost. Here we address energy efficiency only. By analogy to a mother animal attending her offspring we call this problem the "Frugal Feeding Problem". If we impose a total order in which worker robots must be met and charged, (perhaps based on urgency or some other priority scheme) we obtain the "Ordered Frugal Feeding Problem" considered later in this chapter.

Definition 1 (Frugal Feeding Problem)**.** *Given tanker location* $p_0 \in \mathbb{R}^d$*, workers 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*

$$\min_{\pi, p_1, p_2, \ldots p_k} \sum_{i=1}^{k} \left(C_0(p_{i-1}, p_i) + C_{\pi(i)}(r_{\pi(i)}, p_i) \right) \tag{5.1}$$

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 tanker.*

Definition (5.1) 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), another is analytical (finding the meeting points for the given order).

We model locomotion costs as the weighted Euclidean distance between the origin and destination. The weight models the energetic cost of moving the robot some unit distance: a massive robot would have a higher weight than a small, lightweight robot:

$$C_i(x, y) = w_i \|y - x\|, \tag{5.2}$$

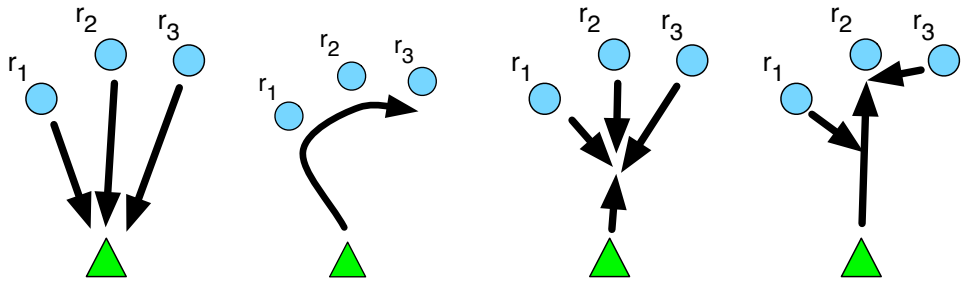
where w_i is the weight of robot i.

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

The Fermat-Torricelli problem (also called the Steiner-Weber problem) asks for the unique point x minimizing the sum of 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 (Kupitz and Martini, 1997), we identify some special cases of solution points as follows.

Definition 2 (Special cases for solution)**.** *If* $p_i^* = r_j^0$ *for some* $j > 0$*, we will call* p_i^* **worker absorbed**. *If* $p_i^* = r_0^0$*, we call* p_i^* **tanker absorbed**. *Otherwise, we call* p_i^* **floating**.

If the meeting point for worker i is worker absorbed, worker i should either remain still and wait for the tanker to come to it, or it should move to the location of worker j ($i \neq j$). If a meeting point is tanker absorbed for worker i, the tanker should not move, and the worker should drive to the original tanker location. Floating meeting points do not coincide with any original robot location. Finally, there may exist hybrid complete solutions which combine absorbed and floating meeting points (see Fig 5.1).



(a) Tanker absorbed (b) Worker absorbed (c) Floating (d) Hybrid

Figure 5.1: 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. (Figure by R. Vaughan)

5.1.1 Related work

The Ordered Frugal Feeding Problem can be addressed as a specific case of the continuous multiple facility location problem (Drezner, 1995), where points $r_0^0, r_1^0, \ldots, r_n^0$ serve as the existing facilities, points $p_1, \ldots, p_n$ are new facilities and the costs are set appropriately.

A variety of numerical methods exist for tackling this generalized problem. However, these numerical methods typically require centralized computation and do not scale well. The main contribution of this chapter is introducing a simple and provably correct scalable decentralized heuristic which is observed to produce multi-point rendezvous with locomotion cost close to that of an optimal Ordered Frugal Feeding solution in a number of realistic experimental settings.

The Weiszfeld algorithm (Weiszfeld., 1937) serves as a basis to many continuous multiple facility location problem algorithms (see Drezner (1995) for a thorough review). Li (1995) improves the slow convergence order of the Weiszfeld algorithm and identifies good convergence properties that hold in non-degenerate cases. Projected direction methods (Calamai and Conn, 1980) offer an alternative to the Weiszfeld approach, however they are arguably less suitable for large scale computation(Li, 1995). Most of the parallel methods for facility location problems focus on discrete cases. However, Rosen and Xue (1991) offer a parallel algorithm for continuous single facility location problem. The theoretical foundation for parallelization of gradient-like algorithms is given in Tsitsiklis et al. (1986). Since as facility location problem the Ordered Frugal Feeding Problem can have degenerate instances (e.g., when more than one meeting point is absorbed by the same worker), convergence of Weiszfeld-inspired methods is not guaranteed.

Related robotics research includes studies of single-point rendezvous problems. Usually authors consider the setting where robots have incomplete information about locations of each other which makes it difficult to coordinate and agree on a single meeting point (Dudek and Roy, 1997; Cortes et al., 2006; J. Lin and Anderson, 2003; Schlude, 2003). For example, Ando et al. (1999) describes and proves the correctness of a distributed rendezvous algorithm for robots with limited visibility. Lanthier et al. (2005) present an algorithm for finding the meeting point which minimizes the maximum individual travel costs to a single meeting point on a weighted terrain. Smith et al. (2007) describe a scheme which makes the convergence process more organized in a certain mathematical sense.

In previous sections, we described a distributed heuristic for a simple single-point rendezvous that minimizes total locomotion costs. For the Ordered Frugal Feeding Problem we have given two custom numerical algorithms that give approximate solutions, and also a complete polytime discrete solution for the case when meeting points are restricted to a fixed set of locations and arbitrary locomotion cost functions in (5.1) We also describes general Frugal Feeding Problem where the order of recharging is not fixed and is also subject to optimization. Reduction from the Euclidean Travelling Salesman Problem (ETSP) to general Frugal Feeding problem shows that the finding optimal charging order is NP-hard.

5.1.2 Analysis

The first observation to make is that solution points p_i^* can not lie outside the convex hull of $\{r_0^0, r_1^0, r_2^0, \ldots, r_k^0\}$. 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. Also, it is easy to prove the convexity of the objective function defined by (1,5.8) (see, e.g.(Boltyanski et al., 1999, p. 239) for the idea of the proof).

Under certain conditions we can quickly find solutions without solving the general problem. We first show the 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 $p_i^* = r_i^0$. If $w_i > 2w_0$, then this solution is unique.*

Proof. The components of the objective function in (1), which depend on p_i are

$$g_i(p_i) = w_0\|p_i - p_{i-1}\| + w_0\|p_{i+1} - p_i\| + w_i\|p_i - r_i^0\| \tag{5.3}$$

We can consider g_i in isolation and show that $g_i(p_i) \geq g_i(r_i^0)$ for $p_i \neq r_i^0$.

$$\begin{aligned} g_i(p_i) = w_0\|p_i - p_{i-1}\| + w_0\|p_{i+1} - p_i\| + w_i\|p_i - r_i^0\| \geq \\ w_0\|p_i - p_{i-1}\| + w_0\|p_{i+1} - p_i\| + 2w_0\|p_i - r_i^0\| = \\ w_0(\|p_i - p_{i-1}\| + \|p_i - r_i^0\| + \|p_{i+1} - p_i\| + \|p_i - r_i^0\|) \geq \\ w_0(\|p_{i-1} - r_i^0\| + \|p_{i+1} - r_i^0\| = g_i(r_i^0). \end{aligned}$$

Here the first inequality follows from the lemma statement and second inequality is a pair of triangle inequalities. Note, that if the inequality in the lemma statement is strict then the first inequality in above is also strict. In this case r_i^0 will indeed be unique optimal meeting point for robots 0 and i. □

Now we show the 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 $p_i^* = r_0^0 = p_0$ for all i. If the inequality is strict, then this solution is unique.*

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

$$\begin{aligned} \Delta C = \sum_{i=1}^k w_i(\|r_i^0 - p_0\| - \|r_i^0 - p_i\|) - \\ w_0 \sum_{i=1}^k \|p_i - p_{i-1}\| \leq \\ \text{(triangle inequality)} \\ \sum_{i=1}^k w_i\|p_i - p_0\| - w_0 \sum_{i=1}^k \|p_i - p_{i-1}\| \leq \\ \text{(let } \|p_j - p_0\| = \max_i\{\|p_i - p_0\|\}) \\ \sum_{i=1}^k w_i\|p_j - p_0\| - w_0 \sum_{i=1}^j \|p_i - p_{i-1}\| \leq \\ \text{(series of triangle inequalities)} \\ \sum_{i=1}^k w_i\|p_j - p_0\| - w_0\|p_j - p_0\| \leq 0 \end{aligned}$$

The case with strict inequality is argued similarly. □

5.1.3 Complexity of combinatorial component

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 TRAVELLING SALESMAN* $\leq$ *FRUGAL FEEDING*

Proof. Given an instance of the geometric travelling 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 (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 travelling 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 travelling salesman problem. Thus, a search for efficient domain-specific heuristics looks 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.

Based on the similarity between objective functions we conjecture that the order given by the solution to the geometric travelling salesman problem set on the points $p_0, r_1, ...r_n$ may often be a reasonable approximation to the optimal order π^*. A possible heuristic then would be to use this order and avoid the expansion of the permuted orderings. At the time of writing we do not have results to evaluate this approach. Literature on the travelling 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 Reinelt (1992) could be used. A cheaper and simpler, but further from optimal alternative ordering could be a nearest-neighbour series, starting at p_o.

In the remaining analysis we will assume that the order π is given. For the practical experiments we use brute force by iterating through all possible orderings, solving the analytical component for each of the orderings and using the ordering which resulted in the best solution.

As mentioned above, the analytical component of the problem is a specific case of the facility location problem, where points $p_0, r_1, ...r_n$ serve as the existing facilities, points $p_1, ..., p_n$ are new facilities and the costs are set appropriately. Because the gradient and Hessian of the cost function are not defined at the locations of existing facilities, defining gradient-based numerical methods is not trivial. Moreover the instance of the facility location problem we consider could be degenerate at the optimal solution, since more then one new facility may coincide with the same existing facility. This makes finding a good numerical algorithm even more difficult.

Below we present three methods to solve the analytical component of the frugal feeding problem. We start from the case where locations of p_i are restricted to some finite set and proceed with two custom numerical methods for the general case with cost functions, described by (5.2).

5.2 Restricted locations case

Assume that the locations where the tanker could attend workers 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 (5.1) 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 naive brute-force algorithm for solving the analytical part of the problem will take $O(|L|^k k)$ time ($|L|^k$ possible meeting points arrangements and $O(k)$ to calculate the cost value). However, it is possible to exploit the structure of the objective function and to provide the algorithm with running time $O(|L|^2 k)$. This algorithm is based on the ideas of dynamic programming (Bellman, 2003). First we describe the algorithm formally and then we consider an example which illustrates our approach.

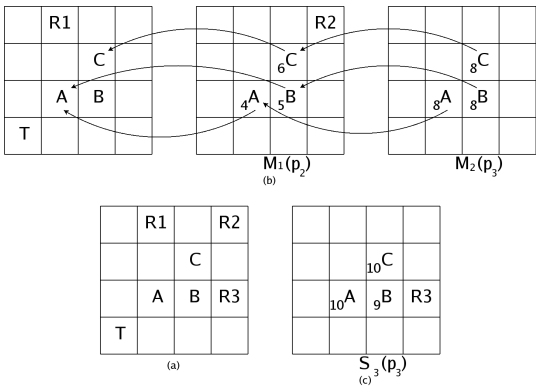


Figure 5.2: Illustrative example. (a) Grid world with rectilinear distances. Tanker is denoted as T, workers as $R1, R2, R3$, possible meeting locations as A, B, C. (b) Construction of maps for problem solution. (c) Function, used in the selection of the minimum cost value.

5.2.1 Algorithm: Formal presentation

Consider the following reformulation of the problem (for a given ordering of robots):

$$\begin{aligned} &\min_{p_k}[C_k(r_k,p_k) + \\ &\min_{p_{k-1}}[C_0(p_{k-1},p_k) + C_{k-1}(r_{k-1},p_{k-1}) + \\ &\min_{p_{k-2}}[C_0(p_{k-2},p_{k-1}) + C(r_{k-2},p_{k-2}) + ... + \\ &\min_{p_1}[C_0(p_1,p_2) + C_1(r_1,p_1) + C_0(p_0,p_1)]...]]] \end{aligned} \tag{5.4}$$

The equivalence of the analytical part of (5.1) and (5.4) is the corollary of the obvious equality $\min_{x,y} f(x,y) = \min_x[\min_y f(x,y)]$.

Now consider the following sequence of functions:

$$\begin{aligned} S_1(p_1,p_2) &= C_0(p_1,p_2) + C_1(r_1,p_1) + C_0(p_0,p_1); \\ S_i(p_i,p_{i+1}) &= C_0(p_i,p_{i+1}) + C_i(r_i,p_i) + S_{i-1}(p_i), \\ &\text{for } i = 2..k-1; \\ S_k(p_k) &= C_k(r_k,p_k) + S_{k-1}(p_k) \end{aligned} \tag{5.5}$$

Problem (5.4) could be solved by sequentially building the mappings

$$\begin{aligned} M_1(p_2) &= (\min_{p_1} S_1, \arg\min_{p_1} S_1); \\ M_i(p_{i+1}) &= (\min_{p_i} S_i, \arg\min_{p_i} S_i), \text{ for } i = 2..k-1; \end{aligned} \tag{5.6}$$

$$M_k = (\min_{p_k} S_k, \arg\min_{p_k} S_k);$$

Each mapping M_i is built by considering every value in L as a parameter and for each value of parameter every value in L is considered as a possible solution to the minimization problem. If minimization does not yield a unique solution, we arbitrarily select one of the minimizing points. Thus, it takes $O(k|L|^2)$ steps to build all maps. M_{k1} gives the minimum value for (5.4) and M_{k2} gives point p_k^*. The rest of the points could be found as $p_i^* = M_i(p_{i+1}^*)_2$.

Note that this algorithm does not depend on the particular form of the movement cost functions.

5.2.2 Algorithm: Illustrative example

Consider the setting described in Fig.5.2a. The tanker is located at point T, and three workers are located at $R1$, $R2$, and $R3$. All robots can move into the 4-connected adjacent cells only, and the locomotion cost for tanker and workers is the rectilinear (Manhattan) distance between the points, measured in number of cells moved. The set of possible meeting places is restricted to three points $L = \{A, B, C\}$. We will solve the problem by building a sequence of maps illustrated in Fig.5.2(b),(c).

The first map $M_1(p_2)$ gives the answer to the question: "for the given location p_2 of meeting with robot $R2$, what is the best place p_1 to meet robot $R1$ in terms of the cost of moving tanker from T to p_1 and then to p_2 plus the costs of moving $R1$ to p_1?" The map also provides corresponding minimal costs. For example, if robot $R2$ is to be met at point B, then the best place to meet $R1$ is A and associated costs are 5 (Fig. 5.2b). If there are several locations which yield the same minimum cost, any of them could be used without affecting the quality of the solution.

Using $R1$ we can build the second map $M_2(p_3)$ which again answers the question "for the given location p_3 of meeting with robot r_3 what is the cheapest place p_2 to meet $R2$ in terms of the accumulated minimum cost of the tanker arriving at p_2 (this is given by $M_1(p_2)$), relocating to point p_3 plus the costs of moving $R2$ to p_2."

Finally, using M_2 we can build the function $S_3(p_3)$ which shows the minimum total cost for each possible point for charging p_3 (Fig. 5.2c). This cost is the accumulated minimum cost of the tanker getting to p_3 which is given by $M_2(p_3)$ plus the cost of moving $R3$ to p_3. By minimizing S_3 we conclude, that the best place for charging r_3 is B. Now we can roll back the maps and find the rest of the locations. The best place to charge $R2$ is given by M_2. Thus, $p_2^* = M_2(p_3^*) = M_2(B) = B$. Similar, $p_1^* = M_1(p_2^*) = A$. The solution to the problem is (A, B, B) and the minimum cost is 9.

5.3 Continuous case

From now on we will use the energy cost functions, described by (5.2). We introduce some notation to describe the algorithms below. The desired convergence precision is ϵ. The objective function is $F(p_1, p_2, ...p_k)$. Define $f((s_i), (c_i), x) = \sum_i c_i||x - s_i||$, $g((s_i), (c_i), x) = \sum_{x \neq s_i} c_i \nabla ||x - s_i||$, $G((s_i)(c_i), x) = \sum_{x \neq s_i} c_i \nabla^2 ||x - s_i||$, $\lambda((s_i), (c_i), x) = \sum_{x \neq s_i} c_i / ||x - s_i||$.

Both methods are based on the generalizations of algorithms for the Fermat-Torricelli problem with proved convergence (see Wang (1975); Xue (1987) cited by Rosen and Xue (1991)).

5.3.1 First order method

We describe one step $s(x)$ of this iterative method. It is used recursively to obtain a sequence of solution approximations $x_{t+1} = s(x_t)$ until $||x_t - x_{t-1}|| < \epsilon$.

At iteration t given a current approximation to the solution $p_i^t, i = 1..k$

1. Let $S(i) = (p_{i-1}, p_{i+1}, r_i), W(i) = (w_0, w_0, w_i)$ for $i = 1..k-1$; $S(k) = (p_{k-1}, r_k), W(k) = (w_0, w_k)$
2. Next approximation $p_i^{t+1} = P(S(i), W(i), p_i)$

Here $P(S, W, p)$ denotes the step of the first-order Wang acceleration of the Weiszfeld algorithm for the Fermat-Torricelli problem (Rosen and Xue, 1991). Given the set of points s_i with weights c_i, $i = 1..m$ and current approximation x we can calculate $F((s_i), (c_i), x)$ as follows:

1. Compute $g' = g((s_i), (c_i), x)$. Compute $\sigma = ||x - s_j|| = \min_i\{||x - s_i||\}$. If $\sigma > 0$, goto step 2, else goto step 3
2. If $g' = 0$ then return x, else return $x - g'/\lambda((s_i), (c_i), x)$
3. If $||g'|| \leq c_j$ then return x, otherwise return $x - [g'/\lambda((s_i), (c_i), x))][(||g'|| - c_j)/||g'||]$

5.3.2 Second order method

The second order method we propose is a Newton minimization procedure. We exploit the fact that the Hessian of the objective function has a block-diagonal structure. Because of this it is possible to calculate the Newton direction separately for each of solution point approximations p_i. If the Hessian for some point is singular, or does not exist because p_i is absorbed, then the first order step is used to calculate the direction. After all directions are calculated, the line search is performed

to calculate the length of the step. This method is based on a second order Xue method for the Fermat-Torricelli problem (Rosen and Xue, 1991).

Initially set $\delta_i = \delta$, where δ is a cutoff parameter. At iteration t given current approximation to the solution $p^t = (p_i^t), i = 1..k$ and current values of cutoff parameters δ_i^t

1. Let $S(i) = (p_{i-1}, p_{i+1}, r_i), W(i) = (w_0, w_0, w_i)$ for $i = 1..k-1$, $S(k) = (p_{k-1}, r_k), W(k) = (w_0, w_k)$

2. For $i = 1..k$ calculate direction and new value of cutoff parameter

$$(\delta_i^{t+1}, d_i) = Q(S(i), W(i), \delta_i^t, p_i)$$

3. Next approximation $p^{t+1} = p^t + \alpha d$,where α is a largest number in $\{1, 1/2, 1/4, ...\}$ such that $F(p^t + \alpha d) \leq F(p^t)$

Here $Q(S, W, \delta_i^t, p_i)$ calculates the direction for a particular component of the solution approximation and a new cutoff parameter value. Given a set of points s_i with weights c_i, $i = 1..m$ and current approximation x we can calculate $Q((s_i), (c_i), \delta, x)$ as follows:

1. Compute $g' = g((s_i), (c_i), x)$. Compute $\sigma = ||x - s_j|| = \min_i\{||x - s_i||\}$. If $\sigma = 0$, then set $\delta^{+1} = \delta$, goto step 4. If $\sigma \geq \delta$, then set $\delta^{+1} = \delta$, goto step 3.

2. (Cutoff step) If $f((s_i), (c_i), s_j) \leq f((s_i)(c_i), x)$ then return $(\delta^{+1}, s_j - x)$

3. Compute $G' = G((s_i), (c_i), x)$ If G' is singular, goto step 4. Otherwise solve system $G'd = -g'$ for d and return (δ^{+1}, d)

4. If $||g'|| \leq c_j$ then return $(\delta^{+1}, 0)$, otherwise return $(\delta^{+1}, -[g'/\lambda((s_i), (c_i), x))][(||g'|| - c_j)/||g'||])$

5.4 Demonstration of proposed numerical algorithms and restricted locations method

5.4.1 Simulation setup

We performed a series of simulations to demonstrate the methods described above. Three different settings are considered, each in a 2-dimensional continuous world with a tanker and five robots

placed as shown in Fig.5.4. The standard locomotion cost function (5.2) is used. The three settings differ only in the weights in the cost functions. In Fig.5.4 the robot start positions are indicated by solid black dots, with the size of the dot representing the relative locomotion cost. The first setting (Map 1) has the weight of the tanker $w_0 = 20$, the worker robots have weights $w_i = 1$. In the second setting (Map 2) $w_0 = w_i = 1$. In the last setting (Map 3) $w_0 = 1, w_i = 2$. Map 1 should have solutions which are tanker absorbed, since the weights satisfy the conditions of Lemma 2. Map 2 could possibly have some floating solution points, and Map 3 satisfies the sufficient conditions in Lemma 1, thus all of the solution points should be worker absorbed. Therefore, these three settings cover three major classes of solutions. Though we can use the sufficient conditions provided in Section 5.2 to find the solutions for Map 1 and Map 3 at once, we use these maps to see how well the methods will perform in capturing the absorbed cases.

As explained in Section 5.1.3 the combinatorial component of the problem is tackled by iterating through all possible orders of visits. Investigating faster methods is left as future work. Here we compare 5 different ways to solve the analytical part of the problem. These are:

1. *Discrete100* covers the embedding square of all robots with a 100 point (10 by 10) uniform rectangle grid and applies the discrete method described in Section 5.2.

2. *Discrete400* uses the discrete method on a 400 point (20 by 20) grid.

3. *Nelder-Mead* is a simplex optimization method which works by querying the value of the objective function in the vertexes of the simplex and updates that simplex accordingly (Mathews and Fink, 2004, Ch. 8). Thus, it does not require any special properties of the objective function to operate. For this method the desired precision of optimization is set to 10^{-4} and maximum number of iteration is 2000.

4. *First Order* is the method described in Section 5.3.1 with the desired precision set to 1.

5. *Second Order* is the method described in Section 5.3.2 with the desired precision set to 10^{-1} and cutoff parameter $\delta = 10$.

These first and second order precision parameter values are selected experimentally to give comparable running times for the numerical methods in all three settings, since we are interested in finding a method which will find good solutions quickly in all three representative settings. All three numerical methods start from the same solution approximation, which is $p_i^0 = r_{k+1-i}$. An alternative approximation could be to set each meeting place to the weighted average between original

tanker location and original worker location, $p_i^0 = (w_0 p_0 + w_i p_i)/(w_0 + w_i)$. We plan to explore the influence of the starting point on the convergence in our future work.

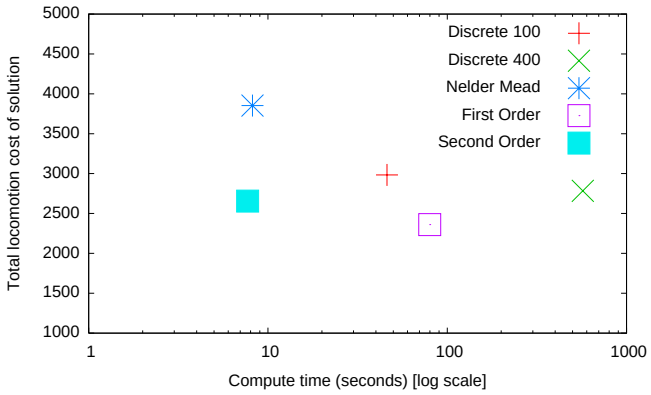
5.4.2 Results

Figure 5.3 shows the results of running each of the five methods in three simulation settings. The first thing to note is the increase in run-time between the 100 point and 400 point grids for the Discrete method, consistent with the complexity estimates presented in Section 5.2. Run-time grows quadratically in the size of the grid. Thus, for practical purposes the discrete method can be recommended only if the number of *a priori* allowed meeting locations is small, or, similarly, the required precision is low so that the search space can be covered by a coarse grid. The fact that the discrete method failed to find the absorbed solutions for Map 1 and 3 is evident from the cost of the solution not being minimal between all methods. The reason for this is that the grid nodes failed to capture the appropriate points. Thus, if a discretization is used to find the solution, the original locations of workers and robots should be included in the set of possible locations L even if the search grid does not capture them.

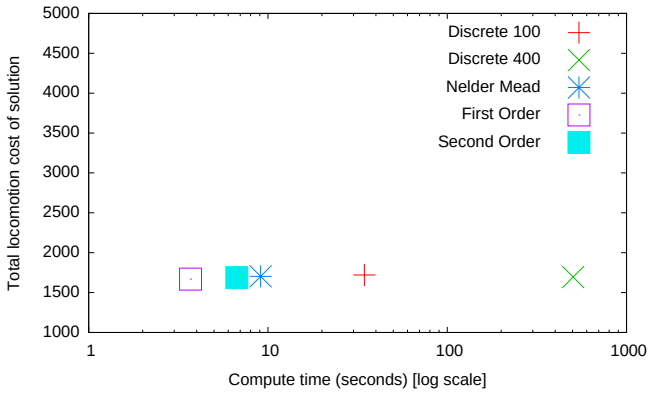
Fig.5.4a,b,c shows the solutions found by the numerical methods on Map 1, which is constructed to be a tanker absorbed case. The first order method finds the best-quality solution but takes a relatively long time for the method to converge. The second order method converges faster, but the solution it finds is worse. However, a change of the required convergence precision of second order method to 10^{-4} increases the time to 16.2s while beating the first order solution quality with the value 2346.71. This confirms our expectation that the second order method should be able to find a high-quality solution faster than the first order method. The Nelder-Mead method converges quickly on this map, however the quality of the solution it found is relatively poor.

Map 2 reveals an interesting result. The first order method manages to find a better solution (Fig.5.4(e)) in less time than the second order method takes to find an inferior solution (see Fig.5.4(f)). Though the difference between the quality of solutions is only 1.2%, it shows, that there is potential for improvement of the second order procedure. Inspection of the sequence of approximations that the second order method produced for this Map shows that many steps had a singular Hessian, thus the method resorts to the first order direction calculation. One of the potential improvements is to take special care of these points. Nelder-Mead performed better on Map 2 than on Map 1, still taking a relatively short time to run (Fig.5.4(d)).

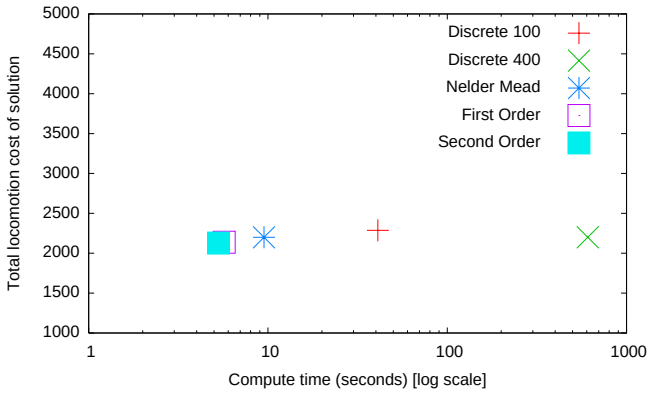
Map 3 shows that all methods find the same order of tanker/worker meetings. Due to the cutoff



(a) Map 1: $w_0 = 20, w_i = 1 | i > 0$ (Tanker absorbed)



(b) Map 2:$w_0 = w_i = 1$



(c) Map 3: $w_0 = 1, w_i = 2 | i > 0$ (Worker absorbed)

Figure 5.3: Experimental results. Solution quality is plotted against compute time for each solution method, each of three settings (maps). Maps 1-3 differ only in locomotion cost weights on the single tanker w_0 and five worker $w_i | i > 0$ robots. (Figure by P. Zebrowski)

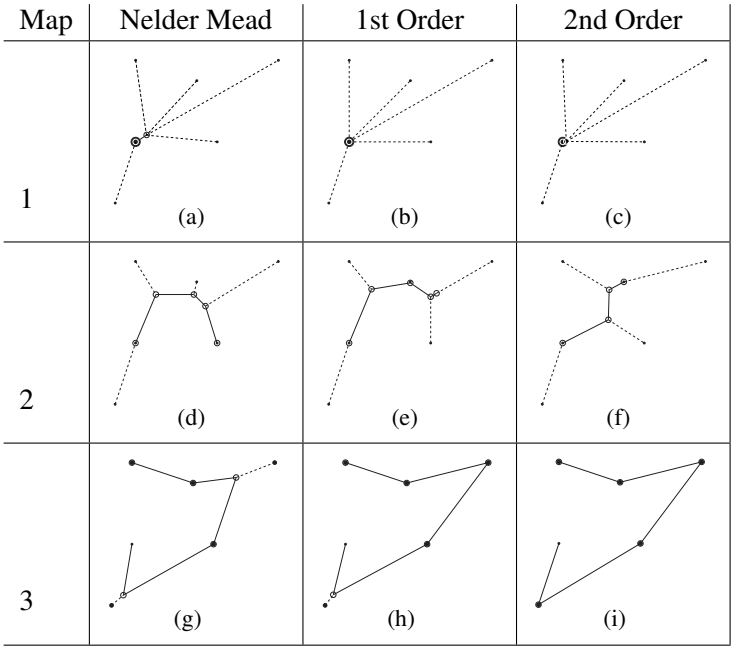


Figure 5.4: Rendezvous solutions discovered. Tanker is at the solid spot on figure (a), other points show workers. Circles denote meeting places. Tanker route is shown with solid lines, workers routes are shown with dashed lines. (Figure by P. Zebrowski)

procedure used in the second order method, it is able to converge precisely to the locations of workers (Fig.5.4(i)), providing the best solution. The first order method fails to put one of the meeting places at the location of the worker. (Fig.5.4(h)). The Nelder-Mead method fails to place two of the meeting places at the best location (Fig.5.4(g)).

Between-map comparison of the numerical methods suggests that the second order method seems to be the safest choice for practical purposes. However, since the first order procedure performs well for the floating case, it could be of practical use when absorbed cases are not likely (based on consideration of the initial conditions). This issue requires more investigation and is left as future work. Finally, the Nelder-Mead method (provided by a multitude of numerical method code libraries) could be used if the quality of the solution is not critical, and convenient implementation is preferred.

5.5 A distributed heuristic for the Ordered Frugal Feeding Problem

5.5.1 Definition of Ordered Frugal Feeding

The Ordered Frugal Feeding Problem can stated formally as follows:

Definition 3 (Ordered Frugal Feeding Problem)**.** *Given original tanker location $r_0^0 = p_0 \in \mathbb{R}^d$, original worker locations $r_i^0 \in \mathbb{R}^d, i = 1, \ldots, k$ and locomotion cost functions $C_i : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}, i = 0, \ldots, k$ find*

$$\min_{p_1, p_2, \ldots, p_k} \sum_{i=1}^{k} \big(C_0(p_{i-1}, p_i) + C_i(r_i^0, p_i)\big); \tag{5.7}$$

$$C_i(x, y) = w_i \|y - x\| \tag{5.8}$$

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 w_i is a weight of robot i.

5.5.2 Algorithm

We present a method whereby a simple controller, running in parallel on each robot, causes the robots to achieve low-cost multi-point rendezvous thus approximating the solution to the Ordered Frugal Feeding Problem (see Algorithm 2). The method is scalable, in that both the total computation time and time to task completion are linear in the population size. Importantly, the per-robot, per-timestep cost is constant, so the method works for arbitrary population sizes since there is no need to increase computational capabilities of robots as the team size grows.

This algorithm requires every worker to know its own weight and the tanker weight, whether or not it is the head of the current charging queue, and the direction toward the next worker in the queue and the previous worker (or tanker if the robot is at the head of the queue). The tanker needs to know its own weight, the weights of the first two robots in the queue and the directions toward them. As a robot is met and charged, it is removed from the queue, and this update is broadcast to all robots. All robots have first order dynamic control. The algorithm is described and analyzed in discrete time but can be reformulated for continuous time. These requirements can be met by a variety of engineering solutions. Locomotion cost (weight) of self can either be pre-set or measured by the robot by sensing its own physical parameters and the properties of the environment. The knowledge of team member weights and own queue position can be acquired by means of communication. Since locomotion costs do not change often, the communication requirements are modest. Measuring directions to

other robots can be performed, for example, using a camera, or lidar, or stereophonic hearing. These sensors can also perform the robot identification task. The advantage of the algorithm is that distance to other robots (which is more difficult to obtain than direction) is not required.

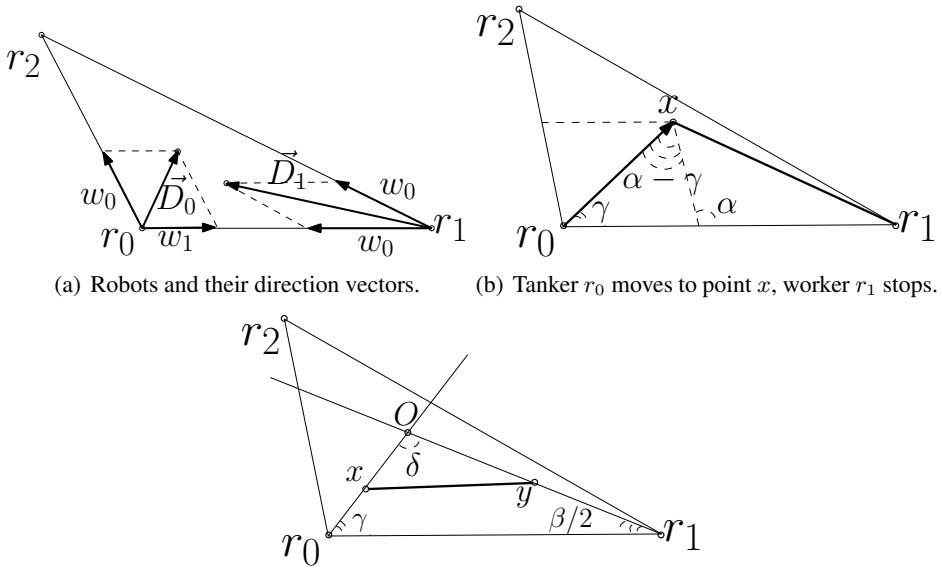
We define a **meeting** as the event when robots come within distance s of each other. We denote the current location of robot i as r_i.

Algorithm 2 Distributed Heuristic

1: $i \leftarrow 1$
2: define $\vec{d}(x, y) = (y - x)/\|x - y\|$
3: **while** $i \leq k$ **do**
4: **if** $\|r_0 - r_i\| < s$ **then**
5: tanker charges worker i; $i \leftarrow i + 1$
6: **next iteration**
7: **if** $i = k$ (only one robot in queue) **then**
8: the lighter of robots 0 and k goes toward the other
9: **end**
10: **for all** $j \in \{0, i, i + 1, \ldots, k\}$ **do** {in parallel}
11: $$\vec{D}_j \leftarrow \begin{cases} w_i\vec{d}(r_0, r_i) + w_0\vec{d}(r_0, r_{i+1}) & \text{if } j = 0 \\ w_0(\vec{d}(r_i, r_0) + \vec{d}(r_i, r_{i+1})) & \text{if } j = 1 \\ w_0(\vec{d}(r_j, r_{j-1}) + \vec{d}(r_j, r_{j+1})) & \text{if } i < j < k \\ w_0\vec{d}(r_k, r_{k-1}) & \text{if } j = k \end{cases}$$
12: **if** $\|\vec{D}_j\| < w_j$ **then**
13: robot j stops
14: **else**
15: robot j proceeds in the direction $\vec{D}_j$

5.5.3 Intuitive explanation

This approach is based on the embodied approximation philosophy described in Ch.3. Instead of operating with the model of the world and searching for the complete solution, each robot uses the position of itself, its queue predecessor and successor as the current approximation to the solution points. Every robot tries to improve the global solution quality by moving in the direction which decreases the part of the total cost function that concerns itself and its neighbors. If the robot finds itself located at the minimizing point for the current local configuration of robots, the robot stops. More formally, robots 0 (the tanker) and 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 2 as an approximation to the unknown solution point p_2. The



(a) Robots and their direction vectors.

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

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

Figure 5.5: 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.

rest of the robots $j, j = 2, ..., k$ move along the approximated antigradient of the cost functions $f_j(p_j) = w_0\|p_j - p_{j-1}\| + w_0\|p_j - p_{j+1}\| + w_j\|r_j - p_j\|$ using r_{j-1}, r_{j+1} as approximations for the unknown solution points p_{j-1}, p_{j+1}. Parallel computation of movement directions in step 11 and simultaneous movement of all robots in steps 12-15 provides the scalability of this algorithm. Unlike the single-point rendezvous heuristic described in previous chapter, here the robot needs to know the direction to only two other robots to calculate its movement direction.

5.5.4 Proof of correctness and run time bounds

Below we prove that Algorithm 2 is correct and all of the workers are eventually charged. To do this we bound the time required for the tanker to meet the first worker in the queue. Since the workers are removed from the queue after being charged, the same bound will apply consecutively to every worker becoming a head of the charging queue.

Theorem 3. *For any initial locations $r_j^0, j = 0, \ldots, k$ and meeting range s if robots $0, 1$ recalculate their movement direction $\vec{D}_j$ every time they travel distance $\epsilon < s/2$ then after at most $\lceil \frac{4\|r_0^0 - r_1^0\|^2}{\epsilon(s-2\epsilon)} \rceil$ iterations robots 0 and 1 will meet.*

Proof. Consider the situation depicted in Fig 5.5(a). We need to show that after robots 0 and 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 0 and 1 will never satisfy their stopping conditions simultaneously before meeting, i.e. at least one of them moves. Second, we bound the decrease of the distance in one iteration in case when 0 stops and 1 moves. Then we do the same for the case when 0 moves and 1 stops. Finally, we bound the decrease of distance for the case when both 0 and 1 move.

First we establish the conditions for 0 and 1 to stop. $\angle r_2 r_0 r_1 = \alpha, \angle r_2 r_1 r_0 = \beta$. Norms of direction vectors $\|\vec{D}_0\|^2 = w_1^2 + w_0^2 + 2w_1 w_0 \cos\alpha$, $\|\vec{D}_1\|^2 = 2w_0^2(1 + \cos\beta)$. Robot 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

$$w_1 + 2w_0 \cos\alpha < 0; \text{(stopping conditions for } r_0) \tag{5.9}$$

Similarly, for robot 1 the stopping conditions are $\|\vec{D}_1\| < w_1$:

$$2w_0^2(1 + \cos\beta) < w_1^2; \text{(stopping conditions for } r_1) \tag{5.10}$$

Note, that since w_0 and w_1 are positive, (5.9) implies $\alpha > \pi/2$.

Both robots stop Substituting $w_1^2 < 4w_0^2 \cos^2\alpha$ from (5.9) into (5.10) and simplifying gives

$$(1 + \cos\beta) < 2\cos^2\alpha, \tag{5.11}$$

As $\beta \leq \pi - \alpha$, and $\alpha > \pi/2$ we have $|\cos\beta| \geq |\cos\alpha|$. Plugging this into (5.11) and simplifying gives

$$2\cos^2\beta - \cos\beta - 1 > 0 \tag{5.12}$$

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 (5.9) and (5.10) has no valid solutions. This means robots 0 and 1 can not stop simultaneously.

0 **stops,** 1 **moves** Initially the distance between robots is $l_t = \|r_0 - r_1\|$. After 1 moves to point x travelling ϵ, the distance between robots becomes $l_{t+1} = \|r_0 - x\|$. Note, that $\vec{D}_1$ always bisects $\angle\beta$.

Using 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:

$$l_{t+1}^2 < l_t^2 + \epsilon^2 - 2l_t\epsilon\sqrt{2}/2 < \tag{5.13}$$

$$l_t^2 + \epsilon^2 - \sqrt{2} l_t \epsilon < l_t^2 + \epsilon^2 - \sqrt{2}\epsilon s;$$

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

$$l_t^2 - l_{t+1}^2 > \epsilon(\sqrt{2}s - \epsilon) \tag{5.14}$$

0 moves, 1 stops This situation is illustrated by Fig 5.5(b). Here tanker 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

$$l_{t+1}^2 = l_t^2 + \epsilon^2 - 2\epsilon l_t \cos\gamma \tag{5.15}$$

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

$$\tan\gamma = \frac{w_0 \sin\alpha}{w_1 + w_0 \cos\alpha} \tag{5.16}$$

Since $\beta \leq \pi - \alpha$, $\cos\beta \geq \cos(\pi - \alpha)$. This inequality can be plugged into stopping condition (5.10) 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 (5.16) and using $w_1 \leq 2w_0$ we obtain

$$\tan\gamma < \frac{w_0 \sin\alpha}{w_1 + \frac{2w_0^2 - w_1^2}{2w_0}} < \tag{5.17}$$

$$\frac{2w_0^2}{2w_0 w_1 + 2w_0^2 - w_1^2} \leq \frac{2w_0^2}{w_1^2 + 2w_0^2 - w_1^2} = 1$$

This implies $\gamma \in [0, \pi/4]$. We use this bound with (5.15) to arrive at the same bound on squared distance as (5.13) describes. Thus, the bound (5.14) applies for this case as well.

Both robots move Fig 5.5(c) shows robot 0 moving to point x and robot 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 law of cosines,

$$l_{t+1}^2 = (a - \epsilon)^2 + (b - \epsilon)^2 - 2(a - \epsilon)(b - \epsilon)\cos\delta = \tag{5.18}$$

$$l_t^2 + 2\epsilon(1 - \cos\delta)(\epsilon - a - b),$$

We start by bounding δ. Note, that $\gamma + \beta/2 + \delta = \pi$. From (5.9) follows, that 0 moves only when $w_1 + 2w_0 \cos\alpha \geq 0$. Given positive weights, if $\cos\alpha \geq 0$ then $w_1 + w_0 \cos\alpha > 0$. If $\cos\alpha < 0$,

then $w_1 + w_0 \cos\alpha = w_1 - w_0|\cos\alpha| > w_1 - 2w_0|\cos\alpha| = w_1 + 2w_0\cos\alpha \geq 0$ (again, given positive weights). Thus, we can use $w_1 + w_0\cos\alpha > 0$ in (5.16) to get $\tan\gamma > 0$. Thus, if r_0 moves, $\gamma \in [0, \pi/2]$. Now, consider two 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

$$1 - \cos\delta > 1 - \frac{\sqrt{2}}{2} \tag{5.19}$$

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,

$$\begin{aligned}
\epsilon - a - b = \epsilon - \frac{l_t(\sin\gamma + \sin(\beta/2))}{\sin\delta} < \\
\epsilon - \frac{s(\sin\gamma + \sin(\beta/2))}{\sin\delta} = \epsilon - s\frac{\sin\gamma + \sin(\beta/2)}{\sin(\gamma + \beta/2)} = \\
\epsilon - s\frac{2\sin(\frac{2\gamma+\beta}{4})\cos(\frac{2\gamma-\beta}{4})}{2\sin(\frac{2\gamma+\beta}{4})\cos(\frac{2\gamma+\beta}{4})} = \epsilon - s\frac{\cos(\frac{2\gamma-\beta}{4})}{\cos(\frac{2\gamma+\beta}{4})} \leq \\
\epsilon - s\cos\Big(\frac{2\gamma-\beta}{4}\Big) = \\
\epsilon - s(\cos(\gamma/2)\cos(\beta/4) + \sin(\gamma/2)\sin(\beta/4)) < \\
\epsilon - s\cos(\gamma/2)\cos(\beta/4) < \epsilon - (\sqrt{2}/2)^2 s = \epsilon - \frac{s}{2}
\end{aligned} \tag{5.20}$$

Combining equations (5.18)-(5.20) we bound the decrease in squared distance:

$$l_t^2 - l_{t+1}^2 > (2 - \sqrt{2})\epsilon\Big(\frac{s}{2} - \epsilon\Big) > \frac{\epsilon}{2}\Big(\frac{s}{2} - \epsilon\Big) \tag{5.21}$$

Comparing (5.14) and (5.21) we see that the latter is 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 $\lceil \frac{4L^2}{\epsilon(s-2\epsilon)} \rceil$. □

Corollary 1. *Consider any initial locations $r_j^0, j = 0, ..., k$, meeting distance s and assume that charging occurs simultaneously. If robots recalculate their movement direction $\vec{D}_j$ every time they travel distance $\epsilon < s/2$ then after at most $\lceil k\frac{4\max_{i,j}\|r_i^0 - r_j^0\|^2}{\epsilon(s-2\epsilon)} \rceil$ iterations all workers will be charged.*

5.6 Experimental evaluation

5.6.1 Comparison with Nelder-Mead method

A set of three experiments was performed to demonstrate the Frugal Feeding Heuristic and to empirically assess the efficiency of the paths it produces and its run-time characteristics.

Experiments are performed in simulation using the Player/Stage robot control and simulation system (Gerkey et al., 2001). The world is an empty circular arena 40 meters in diameter, containing ten mobile robots modelled after the ActivMedia Pioneer 3-DX with SICK LMS-200 laser range finders. Each robot is assigned an individual weight c_i which describes its energy consumption per unit distance travelled. The total energy used as robot i moves from a to b is assumed to be $c_i||a-b||$.

An important and subtle implementation detail concerns the following scenario. Consider a tanker t with cost of movement C_0 and worker w 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 t and w 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, the tanker cost of movement C_0 is changed to $C_0 + \epsilon$ for all experiments. This small bias breaks the symmetry and causes the worker to do all travelling in the above scenario. The value of ϵ is set very small so that it has a trivial effect on the outcome when C_0 and C_i are not equal.

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 d, where here $d = 1$ meter. In a real robot implementation, we assume that another controller could take over to perform a sensor-based docking manoeuvre.

We seek to compare the efficiency of the solutions found by the Frugal Feeding Heuristic and Nelder-Mead methods. 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.

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

$$E = \sum_{i=1}^{n} c_i d_i \tag{5.22}$$

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

Nelder-Mead Controller

The controller uses the Nelder-Mead minimization algorithm (Mathews and Fink, 2004) to find the approximation to the solution of the problem (5.1-5.8). The processing controller then prescribes the approximated solution p_i^* as the location r_i should go to for $1 \leq i \leq n$ where n is the number of

robots. The tanker r_0 is prescribed to visit each of p_i^* for $1 \leq i \leq n$ and wait at each location until the worker arrives.

In order to achieve robustness the solution is recomputed continuously with updated robot positions r_i, and each time the newly generated meeting locations p_i^* are prescribed. This allows the system to recover in case a robot deviates from the prescribed 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, recomputing is performed each time robot locations are updated.

Frugal Feeding Heuristic Controller

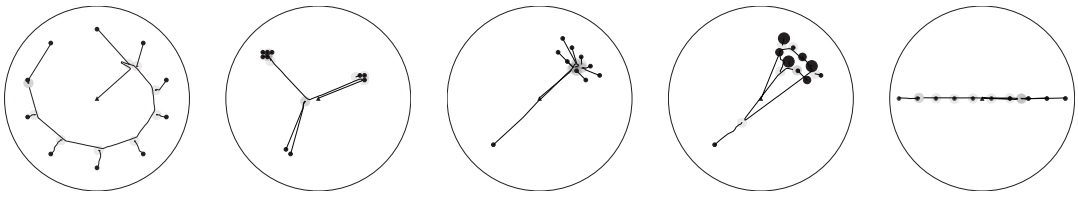
Given the location of each robot r_i and a predetermined visiting order, this controller implements the algorithm described in Section 5.5.2 to compute each robot's direction of travel. This direction is then used to determine a point p_i' which is prescribed to r_i. This results in r_i travelling in the direction originally intended by 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. It should also be noted that this same mechanism is used to implement the stopping conditions.

This computation is repeated continuously with updated robot positions r_i, and each time the newly generated goal locations p_i' are prescribed.

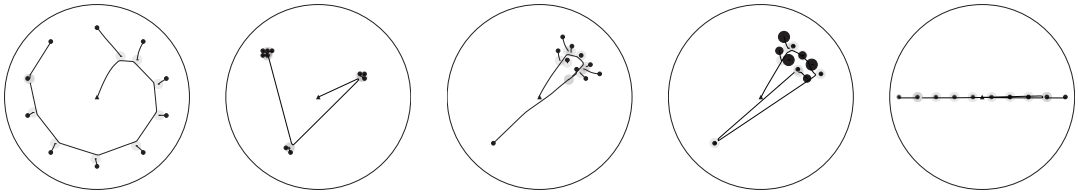
Procedure

Both controllers tested in each of five distinct initial configurations or *maps*. A map consists of a population of ten robots, each with start position, orientation and locomotion cost weight.

26 trials are performed on each configuration/method pair, for a total of 260 trials (approximately 5 hours of total simulation time). The number of worker robots used in simulations is dictated by the rapid growth of time complexity of Nelder-Mead method (see Fig. 5.8). For example, increasing number of workers from 10 to 40 leads to factor of 25 growth of per-step computing time for Nelder-Mead method. This would lead to the corresponding factor of 25 increase in simulation time. In each trial, the total energy used to achieve rendezvous is recorded for statistical performance analysis, along with the trajectory of each robot for visualization.



(a) Map 1 Numerical (b) Map 2 Numerical (c) Map 3 Numerical (d) Map 4 Numerical (e) Map 5 Numerical



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

Figure 5.6: Typical paths taken for maps 1 through 5. Fixed meeting order. Triangle represents tanker, black disks represent workers, with disk radii proportional to worker weight (movement cost). Gray circles represent meeting range. (Figure by P. Zebrowski)

Table 5.1: Total Energy Used For Rendezvous. All Standard Deviations $< 10\%$. (Table by P. Zebrowski)

Method	Map 1	Map 2	Map 3	Map 4	Map 5
Fixed Order Frugal Feeding Heuristic	105.00	61.89	60.89	82.53	52.09
Fixed Order Nelder-Mead	117.08	74.40	65.04	103.26	54.28

Results

Our results demonstrate that both methods achieve the goal of tanker visiting each worker, at a similar energy cost. Figure 5.6 gives an example solution for each method on each map. Visual inspection suggests that both methods find sensible, qualitatively similar solutions. All trials completed with successful complete rendezvous.

Several differences can be observed between Nelder-Mead and Frugal Feeding Heuristic. First, the two methods do not always dictate the same robot path. Figure 5.6(b) shows the path traversed using Nelder-Mead, while figure 5.6(g) shows the same map using Frugal Feeding Heuristic. The difference is reflected in Table 5.1. A second notable difference can be seen in figure 5.6(a) and 5.6(d). Notice the zig-zagging paths taken by the workers. We conjecture this to be an artifact of using a numerical optimization method. 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 travelling at once. The Frugal Feeding Heuristic method produces "stop and go" behaviour. For example, in figure 5.6(a) Nelder-Mead causes all the robots to move toward their meeting location at the beginning of the trial, while in figure 5.6(f), workers do not approach their final meeting location until the tanker is nearby. This emergent behaviour influences the time to completion of the rendezvous, and may also affect the degree of interference between robots. We aim to investigate this in future work.

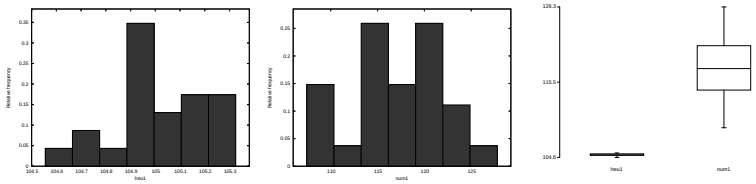
Statistical analysis

The goal of this statistical analysis is to evaluate the statistical significance of the relative performance of Nelder-Mead and Frugal Feeding Heuristic. The hypothesis testing results reported below assume a 5% significance level.

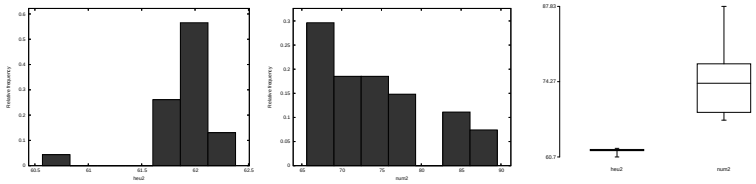
Examining the results within each map, a t-test rejects the hypothesis that Nelder-Mead and Frugal Feeding Heuristic observations come from the same distribution, suggesting that the two methods perform differently in each case. However, the validity of this test is questionable, as the sample size is low (26 trials), and testing the distributions for normality gives contradictory results depending on the test chosen. However, the non-parametric Kruskal-Wallis test (which does not require the normality of samples) produces the same result, and we conclude that, within each of the maps tested, the two methods perform differently. In each case, the mean overall energy used to achieve complete rendezvous was lower for the Frugal Feeding Heuristic than Nelder-Mead.

Next we seek to evaluate the relative performance of the two methods across all maps using the same energy metric. Thus we use all available observations and regress the energy used for rendezvous on the dummy variable $Method2$, representing the Nelder-Mead method.

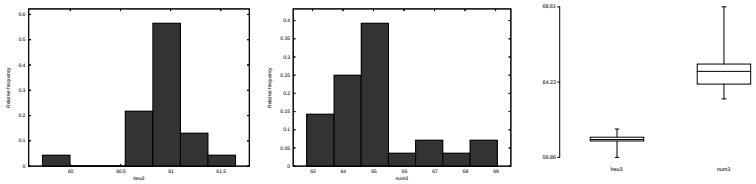
The Wald test and likelihood ratio test for group-wise heteroskedasticity reject the hypothesis that the noise variance is equal between maps. Testing the joint significance of group means shows that the pooled OLS model is inadequate. Thus we choose a fixed effects model corrected for group-wise heteroskedasticity. Summarizing this analysis (see Table 5.2), we observe significant fixed effects of (i) each map, which is reasonable since each map has different theoretically possible minimum energy cost; and (ii) $Method2$, supporting the conclusions 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 mean energy cost and may not be of great practical significance.



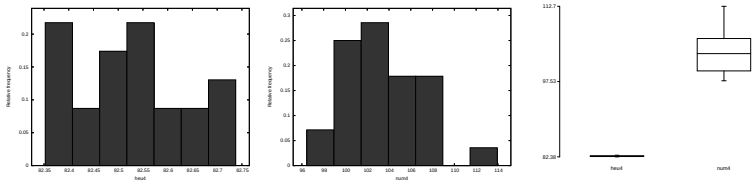
(a) Map 1 Frugal Feeding Heuristic (b) Map 1 Nelder-Mead (c) Map 1 Boxplot



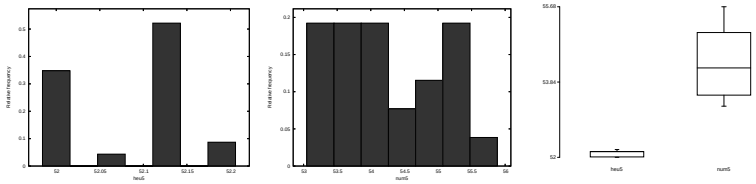
(d) Map 2 Frugal Feeding Heuristic (e) Map 2 Nelder-Mead (f) Map 2 Boxplot



(g) Map 3 Frugal Feeding Heuristic (h) Map 3 Nelder-Mead (i) Map 3 Boxplot



(j) Map 4 Frugal Feeding Heuristic (k) Map 4 Nelder-Mead (l) Map 4 Boxplot



(m) Map 5 Frugal Feeding Heuristic (n) Map 5 Nelder-Mead (o) Map 5 Boxplot

Figure 5.7: Statistical representation of experimental results. Histograms show the distribution of energy losses for repeated experiments on 5 maps using either Nelder-Mead or Frugal Feeding Heuristic methods. Boxplots present results obtained from both methods for every map, left plot representing Frugal Feeding Heuristic, right plot representing Nelder-Mead.

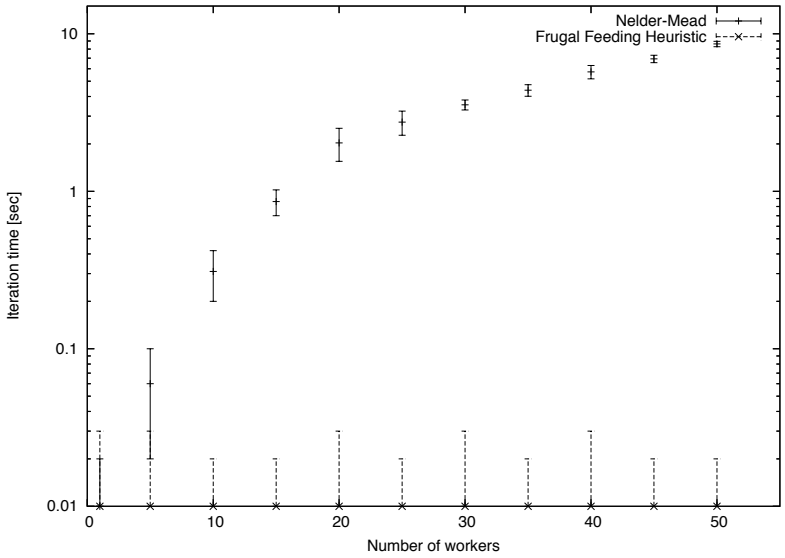


Figure 5.8: Experimental Computation Time (Figure by P. Zebrowski)

We conclude that our empirical evidence shows a small but statistically significant performance advantage of Frugal Feeding Heuristic over Nelder-Mead.

5.6.2 Computation time

A final experiment was performed to compare the computation time used by Nelder-Mead and Frugal Feeding Heuristic with increasing robot populations. We vary the number of workers from 1 to 50 in steps of 5 and measure the computation time required for one calculation cycle. 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 Frugal Feeding Heuristic. The computer used was a 1.5GHz Pentium 4, and each experiment was repeated 20 times. Figure 5.8 shows the results.

The theoretical run-time growth of fixed order Nelder-Mead is undefined. Empirically, we see that computation times increase monotonically, with worse than linear growth. At 50 workers, each computation cycle takes around 10 seconds. While in static environments a few seconds upfront processing is feasible, the poor run-time scaling of the numerical method limits the frequency with which it can be used to respond to new information in dynamic environments.

The theoretical run-time growth of Frugal Feeding Heuristic over all robots is linear in population size. In practice we find the constants are so small that the experimental timing measurements are dominated by noise (probably operating system process scheduling effects) for these population

Table 5.2: 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)

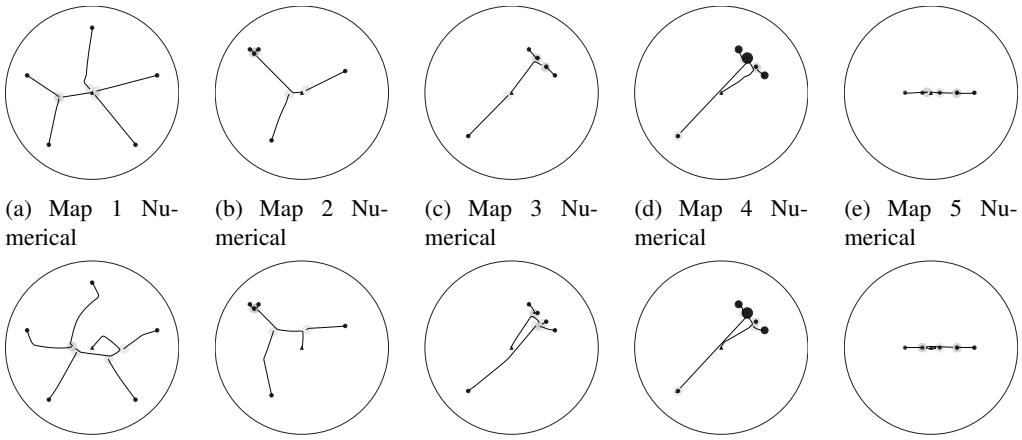
Variable	Coefficient	t-statistic	p-value
du_1	106.811	192.5168	0.0000
du_2	63.9341	88.8531	0.0000
du_3	58.3833	109.3911	0.0000
du_4	89.1727	103.7252	0.0000
du_5	48.6236	75.5821	0.0000
Method2	8.72431	17.2682	0.0000

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



(a) Map 1 Numerical (b) Map 2 Numerical (c) Map 3 Numerical (d) Map 4 Numerical (e) Map 5 Numerical

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

Figure 5.9: Typical paths taken for maps 1 through 5. Variable meeting order. Triangle represents tanker, black disks represent workers, with radius proportional to worker weight (movement cost). Gray circles represent meeting range. (Figure by P. Zebrowski)

sizes, so no slope is seen. The computational cost *per robot* is constant, so when performed in parallel, the method scales perfectly. Thus Frugal Feeding Heuristic is suitable for use in dynamic environments, where a good frequency response can be expected.

5.6.3 Reintegrating ordering

Table 5.3: Experiment 3: Total Energy Used For Rendezvous. All Standard Deviations $< 10\%$. (Table by P. Zebrowski)

Method	Map 1	Map 2	Map 3	Map 4	Map 5
Frugal Feeding Heuristic + nearest neighbor	85.7	45.6	42.7	42.5	19.5
Nelder-Mead + brute force	71.4	39.8	31.5	42.7	16.5

To return to our original problem and demonstrate alternatives for a complete solution, we pair our two methods with simple ordering algorithms.

Method A is Nelder-Mead paired with brute-force ordering, where we compute a solution for each order permutation, and choose the complete solution with lowest cost. This is our baseline for comparison, and represents a "conventional" solution.

In lieu of a well-motivated ordering heuristic, we pair Frugal Feeding Heuristic with a simple

nearest-neighbour ordering, to create Method B. The tanker visits the workers in order of their distance at trial start.

Due to the combinatorial complexity of the brute-force solution, we are limited to experiments with five worker robots. 5.9 shows example solutions for the two complete methods, and Table 5.3 shows the performance results obtained.

In these experiments, Method A leads to slightly better performance, at the cost of vastly more computation. As expected, the quality of the nearest-neighbour ordering depends on the initial configuration, and is sometimes very poor, for example in Figure 5.9(j) where the tanker takes an inefficient route. Finding better ordering heuristics is an area for future research.

5.6.4 Estimating quality of solutions

Having shown that the method succeeds in bounded time, we seek to evaluate the quality of the solutions it generates. This is complicated because (i) the continuous problem has no known closed form solution, and (ii) no quality bounds are available for numerical approximation approaches. However, we can obtain optimal solutions for the discrete version of the problem, and we can provide an upper bound on the cost introduced by the discretization as follows.

Ordered Frugal Feeding cost function is

$$C(p_1, p_2, \ldots, p_k) = \sum_{i=1}^{k} (w_0 ||p_{i-1} - p_i|| + w_i ||r_i - p_i||),$$

where p_0 is the original location of tanker, r_i are the original location of workers, $p_i, i > 1$ is the sequence of meeting points, w_i are the the weights of the robots and k is the number of workers. We know that all optimal meeting points are inside the convex hull, formed by $R = \{p_0, r_1, r_2, \ldots, r_k\}$. Let us set up a regular grid G on the embedding rectangle E of R. We divide the horizontal axis of rectangle into d_x regular interval and vertical axis into d_y intervals. Thus, $\forall p \exists p_g((p \in E) \rightarrow ((p_g \in G) \wedge (||p_g - p|| \leq d)))$, where $d = \sqrt{(d_x^2 + d_y^2)}/2$.

Let the optimal sequence of meeting points be $P^* = p_1^*, p_2^*, \ldots, p_k^*$. By definition of our regular grid there exists a sequence $P = p_1, p_2, \ldots, p_k$ "close" to the P^* such that $p_i \in G$ and $p_i - p_i^* \leq d$. Carefully considering how far these bounded deviations from optimal meeting points can bring general cost we observe

$$C(P) \leq C(P^*) + \sum_{i=1}^{k} w_i d + w_0 d + w_0 \sum_{i=2}^{k} (2d) = C(P^*) + d(2w_0(k-1) + \sum_{i=0}^{k} w_i)$$

Discrete search finds the sequence of points P' which is the best sequence of all formed by taking the meeting points from grid G, including P. Hence, $C(P') \leq C(P) \leq C(P^*) + d(2w_0(k-1) + \sum_{i=0}^{k} w_i)$, or

$$C(P') - C(P^*) \leq d(2w_0(k-1) + \sum_{i=0}^{k} w_i)$$

which is a bound we sought.

So our empirical evaluation procedure is as follows: (i) generate a set of robot weights and start positions within a bounding rectangle; (ii) find the cost of the optimal solution to the corresponding discretized problem, where robot start positions are quantized to a regular grid within the bounding rectangle. A lower bound on the cost of the optimal continuous solution is given by subtracting from this the maximum possible additional cost due to discretization; (iii) find the upper bound of approximation factor by dividing the Frugal Feeding Heuristic cost by the lower bound of optimal cost.

Applying this methodology, we performed 3000 simulations running the Frugal Feeding Heuristic on randomly generated instances of the problem, with initial robot locations drawn from the same uniform distribution, and 3 different uniform distributions of weights, with 1000 experiments for each weight distribution. In each experiment, 10 randomly weighted worker robots and a tanker were placed at random locations in an square arena with 20m sides. A meeting range of 0.1m and a movement step length of 0.01m were set. The path travelled by each robot was recorded, and its length was multiplied by the robot's weight, then summed for the population to give the total energy spent on performing the rendezvous. The arena was discretized using a 40x40 grid, giving a resolution of 0.5 by 0.5m.

Table 5.4 reports the statistics of these approximation factor bounds for each distribution. For example, in the setting where all robots have weight of 1, we observed that the Frugal Feeding Heuristic solution cost is on average no worse than 1.19 times the best possible solution.

Table 5.4: Statistics of approximation factor bounds

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

The best approximation is achieved when all robots have identical weights. The results show

increasing variation of the approximation factor bounds with increasing variation in robot weights. This is due to the method's use of only local information about neighbouring robots in the rendezvous queue. To see this, consider a queue of three workers to be met, with weights $[1, 1, 10]$. The tanker will not consider the third worker until it has met the first, so the tanker may move suboptimally away from the high-weight robot at first. The approximation factor can be made arbitrarily bad by increasing the weight of the tanker and the third worker.

As the support of the weight distribution expands, the probability of this effect increases. Hence the tail of the approximation factor distribution becomes more fat and the kurtosis increases. However, most of the probability remains concentrated around good approximations, hence the skewness of the distribution also grows and the mean does not increase much.

If we know in advance that motion weights are likely to vary widely, this effect can be reduced (but not eliminated) by increasing the number of neighbouring robots considered by the tanker and/or workers, with a corresponding constant factor increase in computation, sensing and communication cost.

Summarizing, the average quality of approximated solutions for uniformly distributed problem appears very good. Therefore, the method could be used where the guaranteed quality of results obtained in a non-scalable centralized manner should be sacrificed in favour of a simple decentralized scalable solution with good average performance.

5.7 Summary

In this chapter we considered the natural optimization problem of finding an energy-efficient simultaneous motion plan 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).

Focusing on the analytic component, we described and compared a discrete algorithm for the case of restricted set of locations and two numerical methods for the continuous case with weighted Euclidean distance energy cost functions. The discrete method produce optimal results in polynomial time; a useful improvement over exponential time naive brute force approach. Our numerical algorithms for the general problem are shown empirically to converge to good solutions.

We have shown that the combinatorial component is NP-hard and proposed a scalable distributed heuristic solution to the analytic component. This heuristic exploits embodiment and the natural parallelism of the robot system to achieve high-quality approximated solutions in arbitrary population

sizes with trivial computation per-robot per-timestep, and using information that should be feasible to obtain in practice. In contrast, the conventional numerical optimization approach does not scale well. We established the correctness of the method by bounding convergence time, and gathered empirical evidence of its average solution quality by experiment.

Chapter 6

Robot sorting with a smooth controller

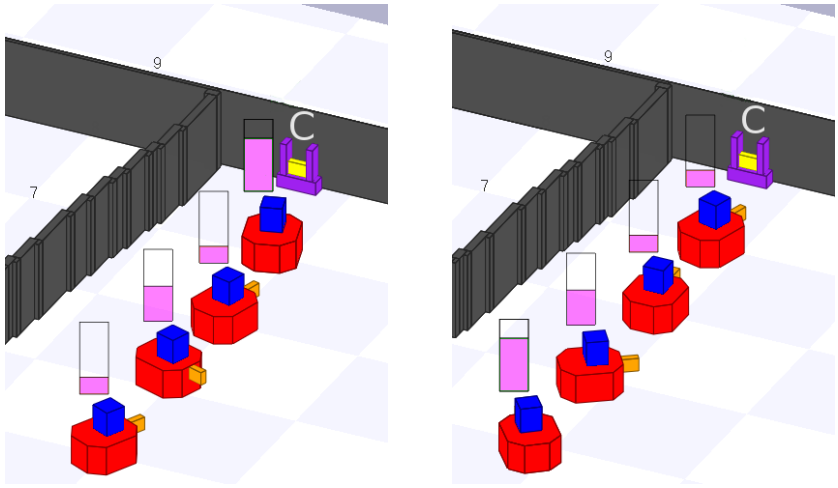
6.1 Introduction

Ordering robots may be a necessary part of many robotic tasks. For example, a team of robots may need to board a transport vehicle in a certain order to use cargo space more efficiently. Likewise, a certain order may be preferred when deploying robots from that vehicle. Maintaining a priority queue of robots may be required when robots line up for service or refuelling (see Fig. 6.1) or when robots perform a convoying task. Finally, ordering robots may improve performance of certain spatial interference resolution algorithms (Zuluaga and Vaughan, 2005).

In previous chapters we showed how the continuous dynamics of a multi-robot system can be used to solve problems with continuous solution space, namely continuous optimization. However, the sorting problem is in a different domain as it is a combinatorial problem with a discrete solution set. In this chapter we explicitly consider solving this discrete problem by a continuous dynamics of a multi-robot system (Litus and Vaughan, 2010). We describe a decentralized multi-robot controller that sorts robots by coupling them according to the Brockett double bracket flow system. This controller is a novel robotic application of the well-known Brockett system.

Let n robots on a plane be initially positioned on a line $y = y_0$ at coordinates $(x_i(0), y_0))$, $i = 1, \ldots, n$ in a Cartesian coordinate system. Assuming, without loss of generality, that the sought ordering coincides with the robot numbering, the goal is to drive the system to a state $(x(t), y(t))$ where $x_1(t) < x_2(t) < \ldots < x_n(t)$ and $y_1 = y_2 = \ldots y_n = y_0$. That is, we want the robots to sort themselves along the horizontal axis and return to the line where they started. The sorting computation should be performed solely by the dynamics of interacting and moving robots.

The next section presents related work, which is followed by a description of the Brockett sorter,



(a) Robots lined up in arbitrary order. The last robot in the queue has a high chance of running out of charge before reaching the station.

(b) Robots are sorted by the remaining charge. The chance of starvation is minimized.

Figure 6.1: Robots queueing for recharging at the station "C" located near the wall. The share of remaining charge is shown for every robot. Sorting robots increases the probability that all robots can charge before running out of energy.

a dynamical system capable of performing sorting by means of smooth dynamics. After an informal theoretical argument about issues that may arise in using the Brockett sorter in a real robotic system we describe a controller based on this sorter. The controller is tested in a short demonstration followed by discussion of the observed behaviour and properties of the system. The chapter concludes by suggesting several directions for future work.

6.2 Related work

Since robotic systems evolve in continuous state space, the relation between discrete computation and continuous systems is very important for treating robot systems as computers. Brockett (1988) shows that double-bracket matrix flow dynamical systems can serve as an analog computer solving a variety of combinatorial problems including sorting. These systems bear similarity to finite aperiodic Toda lattices, a recent thorough review of which is given by Kodama and Shipman (2008). Saxena and Clark (1995) describe an electronic implementation of Brockett double bracket flow built from analogue integrated circuits. Zavlanos and Pappas (2006) describe a dynamical system inspired by the double bracket flows that approximates the solution to the combinatorial weighted graph

matching problem. Bloch and Crouch (1999) argue that from control theoretical point of view some combinatorial problems can be seen as minimization of a function over a set of configurations which can be contrasted with minimization over a class of curves in classical optimal control. The problem of sorting robots could be related to formation control (see Bahceci et al. (2003) for a recent review).

6.3 Brockett smooth sorter

In a seminal paper Brockett (1988) analyzes dynamical systems

$$\dot{H} = [H, [H, N]] \tag{6.1}$$

where H and N are symmetric matrices and $[A, B] = AB - BA$. He shows that these so called "double bracket flow" systems define an isospectral gradient flow, so as H evolves its eigenvalues do not change. Brockett proves that these systems can serve as a versatile analog computer solving linear programming problems, certain combinatorial optimization problems and diagonalizing symmetric matrices. This last ability is of a particular interest for robot sorting since it provides means to sort a list of numbers by a smooth dynamical system.

Brockett proves that if N is a diagonal matrix with distinct elements, the equilibrium matrix $H(\infty)$ will be a diagonal matrix with its elements arranged in the same order as elements of N. In particular, if $N = \text{diag}(1, 2, \ldots, n)$ then for almost all Θ and for

$$H(0) = \Theta^T(\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n))\Theta$$

the equilibrium $\dot{H} = [H, [H, N]]$ will approach

$$H(\infty) = \text{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$$

where permutation π sorts the final list by its size. Our choice of $H(0)$ is dictated by a need to decrease the number of state variables in order to simplify the controller. Diagonal $H(0)$ has the fewest number of variables, but produces no dynamics. However, symmetric tridiagonal matrix $H(0)$ will produce the desired dynamics and result in $H(t)$ being symmetric tridiagonal for any time t. Setting

$$H(0) = \begin{pmatrix} b_1 & a_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & a_{n-1} \\ & & a_{n-1} & b_n \end{pmatrix} \tag{6.2}$$

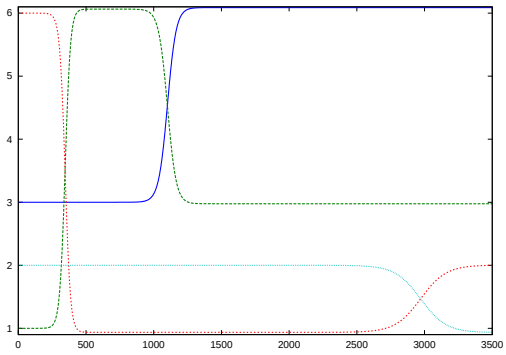


Figure 6.2: Dynamics of sorting system (6.3-6.4) initialized with $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. Four components of vector b (vertical axis) are plotted against time (horizontal axis). In 3 500‘ simulation steps the system converges to $b = (6.088, 2.976, 1.999, 0.937)$.

where $b_i, i = 1, 2, \ldots, n$ are the values to be sorted and $a_i, i = 1, 2, \ldots, n-1$ are small non-zero values will make $H(\infty)$ a diagonal matrix containing entries that approach the sorted list of eigenvalues that are close to b_i. The smaller the values of a_i, the closer matrix $H(0)$ is to being diagonal and having b_i as its eigenvalues, and the closer the diagonal of $H(\infty)$ is to the list of sorted components of b_i. Decreasing a_i, though, comes at the cost of increasing convergence time.

System (6.1) with $N = \text{diag}(1, 2, \ldots, n)$ and $H(0)$ as in (6.2) is equivalent to symmetric tridiagonal Toda equations (Bloch et al., 1990). These equations describe the behaviour of the system of n unit mass particles arranged along a line with adjacent particles interacting with a magnitude that exponentially depends on the distance between them (Toda, 1967). Component-wise in terms of a and b dynamics of this system can be written as

$$\dot{a}_k = a_k(b_{k+1} - b_k), \ k = 1, 2, \ldots, n-1 \tag{6.3}$$

$$\dot{b}_k = 2(a_k^2 - a_{k-1}^2), \ k = 1, 2, \ldots, n \tag{6.4}$$

with initial conditions $a_0 = 0$.

Fig. 6.2 illustrates the evolution of the system (6.3-6.4) with initial values $b(0) = (3, 1, 6, 2)$, $a(0) = (.001, .001, .001, .001)$. The system was simulated in discrete time as $b(t+1) = b(t) + \delta\dot{b}(t)$, $a(t+1) = a(t) + \delta\dot{a}(t)$ with discretization parameter $\delta = 0.005$. Four lines on the figure show the values of four components of vector b plotted against simulation time. After 3500 steps the value of the vector b converges to $b(3500) = (6.088, 2.976, 1.999, 0.937) \approx (6, 3, 2, 1)$: the desired ordering.

6.4 Application to robot sorting

6.4.1 Theoretical considerations

The dynamical system (6.3-6.4) (hereinafter "the sorting system") can serve as a basis for a multi-robot system controller that sorts robots. To do this we need to find a way to embed the sorting system into the state space of a multi-robot system. This way the movement of the robots can be dictated by the evolution of sorting system and result in robots sorting themselves.

Regardless of the means of embedding the sorting system into the state space of a multi-robot system we need to acknowledge the limitations a realistic robot system imposes on the possible trajectories in a state space. For example, robots have limited speed and acceleration and may be non-holonomic. Also, a real system will have noise present in sensor readings and responses to control inputs. Finally, collision avoidance should be employed by a real robot system introducing additional constraints on trajectories. This raises an important question: is the sorting system able to withstand some perturbation of the state variables as the system evolves and still converge to the same equilibrium? A brief analysis shows that this is not the case. Assume that sorting system evolves from tridiagonal matrix $H(0)$ to diagonal matrix $H(\infty) = \text{diag}(\lambda_{\pi(1)}, \lambda_{\pi(2)}, \ldots, \lambda_{\pi(n)})$ and at time t one of the state variables was perturbed resulting in matrix $H'(t) \neq H(t)$. As long as the spectrum of $H'(t)$ differs from the spectrum of $H(t)$ the system will now converge to a different equilibrium $H'(\infty) = \text{diag}(\lambda'_{\pi(1)}, \lambda'_{\pi'()}, \ldots, \lambda'_{\pi(n)})$. There is no feedback in the system to correct this deviation from the original trajectory and restore original spectrum of H. In this sense the sorting system is fragile and using it as a base for a robot sorting controller seems to be difficult.

However, the sorting system should not be discarded because of its fragility. Despite the sensitivity of the equilibrium to the perturbation of state variables, all equilibriums are in fact diagonal matrices with sorted eigenvalues. Thus, if some state variable s_i of the i-th robot in the sought ordering behaves as the i-th diagonal entry of H, the equilibrium values $s_i(\infty)$ will follow the sought ordering irrelevant of the changes in the spectrum of H brought by the deviations from the perfect trajectory. In other words, though the actual values of $s_i(\infty)$ will differ from what they could have been in the absence of perturbations, as long as the system is allowed to converge the values of robots state variables $s_i(\infty)$ will be sorted. In this sense the sorting system is reliable and we can attempt to use it it to control an appropriately constructed robot system.

6.4.2 Implementation

We will embed the sorting system into the robot system as follows. x_i will correspond to the diagonal entries of H while $y'_i = y_i - y_0 + \epsilon$ where ϵ is a small non-zero value will correspond to non-diagonal entries of H. In these variables, sorting system (6.3-6.4) can be rewritten as a first order controller

$$\dot{x}_1 = 2y_1^2, \tag{6.5}$$

$$\dot{x}_i = 2(y_i^{'2} - y_{i-1}^{'2}),\ i = 2, \ldots, n-1 \tag{6.6}$$

$$\dot{y}_i = \dot{y}'_i = -(x_i - x_{i+1})y'_i,\ i = 1, \ldots, n-1 \tag{6.7}$$

$$\dot{x}_n = -2y_{n-1}^{'2}, \tag{6.8}$$

$$\dot{y}_n = y'_n = 0; \tag{6.9}$$

This controller requires every robot i to use the following information:

1. y'_i, the vertical distance from the original line $y = y_0$
2. y'_{i-1}, the vertical distance of the previous robot (if there is one) in ordering from the original line $y = y_0$
3. $(x_i - x_{i+1})$, horizontal distance to the next robot (if there is one) in the ordering

Note, that y'_{i-1} can be computed from y'_i if robot i knows the vertical distance to robot $i-1$. Therefore, the controller requires the robot to be partially localized (know an estimate of its y coordinate) and be able to estimate horizontal distance to one robot and vertical distance to another robot. These requirements can be met by various sensory/communication solutions. To avoid any sort of communication, in the demonstration robots are equipped with fiducial detectors that can determine the relative bearing and distance to the previous and next robot in the ordering. Robots are also localized, though we use only orientation and y coordinate. Horizontal or vertical distance to a team member is calculated from the position of self, and estimates of distance and bearing to the team member. Thus, all information required by the controller is available in this robot system.

There are several issues that emerge when using this controller in a realistic system (or realistic simulation). First, fiducial finders require clear line of sight and hence occlusion can prevent the robot from getting information about distances to other robots. Second, as all sensors, fiducial finders are noisy, therefore information about bearing and distance to the teammates is imprecise. Third, all robots have bounded magnitude of their speed vector, and non-holonomic robots have also

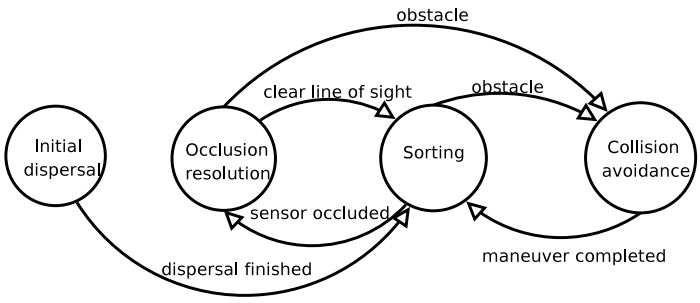


Figure 6.3: State diagram of the robot sorting controller

bounds on the direction of this vector. Finally, robots can not drive through each other so collision avoidance should be used. All these issues will force the robots to deviate from the trajectory prescribed by the sorting system, however as we argued above we still expect the robots to converge to the sorted order. We will address aforementioned issues in the following paragraphs. The state machine of the resulting controller is shown on Fig. 6.3.

Occlusions Robots have no knowledge of the team size or their absolute position in the ordering. They are capable of sensing positions of the previous and the next robot in the ordering if the line of sight is not occluded by other robots. Initially every robot assumes that it is the only robot in the team. Once robot i senses the previous robot $i-1$ in the ordering, without sensing robot $i+1$ before, robot i assumes that i is the last robot. It will behave accordingly and use the relative position of robot $i-1$ in its future calculations of speed vector. If robot i senses the next robot $i+1$ in the ordering without previously sensing $i-1$, robot i assumes that i is the first robot. If robot i senses $i-1$ and $i+1$ simultaneously or in any order, it knows that i is not the first and not the last robot and needs relative positions of both $i+1$ and $i-1$ to calculate its speed vector. Therefore, the robot discovers its position in the ordering (last,first, in between) as the dynamics unfolds.

Regardless of the currently assumed position of robot i if its fiducial sensor is occluded and robot i can not see one or both of the robots it needs to calculate the speed vector, it reduces speed to a certain predefined constant without changing its direction. Once the line of sight is restored, the robot selects the desired speed and direction following Eq. (6.5). Since initially all robots are located along a line and almost all fiducial finders are occluded, the robots disperse themselves by moving with random speeds along the vertical axis for a fixed predefined time. With high probability this allows most of the robots to observe the required teammates and follow Eq. (6.5-6.9).

Noise We will not use any noise reduction techniques and instead treat all noisy sensor readings as true values.

Bounds on a speed vector If the magnitude of the desired speed vector $\vec{v} = (\dot{x}, \dot{y})$ exceeds the maximum speed $V_{\max}$ of the robot, the robot tries to set its speed vector to $\vec{v}_{\text{clamped}} = \frac{\vec{v}}{\|\vec{v}\|} V_{\max}$ thus attempting to go in the desired direction with the maximum speed. In the demonstration below we use non-holonomic steering robots which are driven by a simple negative feedback controller

$$\dot{\theta} = \theta - \angle\vec{v} \tag{6.10}$$

$$s = \|\vec{v}\| \cos(|\dot{\theta}|), \tag{6.11}$$

where θ is the robot bearing, $\angle\vec{v}$ is the direction of the desired speed vector in the same coordinate system, s is the driving speed.

Collision avoidance We employ a simple collision avoidance algorithm that uses laser range-finder sensor readings. If there is an obstacle closer than a certain distance d_{stop}, the robot stops. If there is an obstacle which is at closer than a certain distance $d_{\text{avoid}} > d_{\text{stop}}$ then the direction that gave the smallest distance reading is found. If smallest reading came from the direction to the right of the robot bearing, a collision avoidance manoeuvre with a duration randomly selected in a certain interval is performed. The robot starts to turn left with a fixed turning speed and driving speed. Otherwise, the robot performs a right turn manoeuvre. If smallest reading came from the left, a right turn manoeuvre is performed. Once the collision avoidance manoeuvre is over, the robot continues to set the speed as prescribed by Eq. (6.5-6.9).

6.4.3 Demonstration

For the demonstration we use a team of simulated robots. Robots are placed along a horizontal line and the sorting controller is executed simultaneously on every robot. We perform two kinds of simulations. In the first, "idealistic", simulation the robots are holonomic, with no speed restrictions, no sensor occlusions and no collisions. Therefore, the dynamics of this system are described by Eq. (6.3-6.4). In the second, "lifelike" simulation Pioneer robots are simulated in the well-known Stage robot simulator (Vaughan, 2008). Simulated Pioneer robots can collide, so they need to use collision avoidance, they have non-holonomic driving, top speed restriction and their fiducial sensors can be occluded by other robots. Sensor noise is simulated by adding uniformly distributed random errors to the true distance and bearing reading of fiducial finder before they are used by a controller.

Maximum speed	1.2 m/s
Collision avoidance speed	0.1 m/s
Collision avoidance turning speed	0.5 rad/s
Collision avoidance initiation distance	1 m
Minimum front stopping distance	0.5 m
Collision avoidance duration interval	[1,2]s
Speed during occlusion	0.2 m/s
Initial dispersing duration	1.5s
Fiducial finder bearing accuracy	$\pm$3 degrees
Fiducial finder range accuracy	$\pm$ 15 mm

Table 6.1: Parameters used in Stage simulation

Robots in the Stage simulation use the controller described in Section 6.4.2. Parameters of the Stage simulation are given in Table 6.1. All simulations run until the speeds of all robots converge to a near-zero value.

Figure 6.4 shows the trajectories produced by the robots for three different initial conditions. For every initial condition two trajectories are shown. The first trajectory is produced by an idealistic simulation, the second by a lifelike Stage simulation. Note that Stage simulations are nondeterministic because of the sensor noise and initial random dispersal and infinitely many different trajectories are possible of which we show only one. The idealistic simulation trajectory, on the other hand, is repeatable and fixed up to the rounding and discretization errors during simulation. Each trajectory is described by two graphs. The first graph plots values of x coordinates of robots against simulation time, the second shows the joint (x, y) trajectories of the robots.

In the idealistic simulations most of the robots initially start moving with small speed with vertical component of the speed vector dominating horizontal component. As robots move away from the start horizontal line their speed grows, the horizontal component of the speed vector increases and the vertical component decreases to the point where the vertical component becomes negative and robots begin to return to the start horizontal line but with different horizontal coordinate. Some of the robots move along the start horizontal line for the part of their trajectory while the last, n-th robot never leaves the start line moving only horizontally (see Eq. (6.9)). Depending on the initial conditions robots may depart from and return to the start line several times before the robots converge to the sorted order (see, e.g., robots starting at positions $x = 3$ and $x = 5$ on Fig. 6.4(e),6.4(f)). Also, robots may switch their vertical direction before reaching the start line (see, e.g., robot starting at positions $x = 13$ on Fig. 6.4(e), 6.4(f) and robot starting at position $x = 8$ on Fig. 6.4(i), 6.4(j)). Once the sorted state is reached, robots do not depart from it any more. The final configuration has

robots rearranged in the sorted order along the start horizontal line with the set of final horizontal positions having values very close to the set of original horizontal positions. Therefore, the robots not only end up in the sorted order, but they also jointly occupy same horizontal positions that where occupied by the team initially.

In the Stage simulations robots initially disperse themselves by moving for a fixed time with a random speed in a vertical direction. This eliminates some of the occlusion and allows some robots to move in the direction prescribed by the sorting system. After dispersal the robots start moving with increasing speeds which is limited by the top speed of the robot. Occlusions that were not resolved by the initial dispersal are eventually resolved as occluded robots move, slowly creating new lines of sight (see Section 6.4.2). Some occlusions are resolved by the collision avoidance behaviour. Robots move away from the horizontal line with horizontal component of their speed vector increasing and vertical component decreasing until the vertical component changes the sign and robots return to the start line at a different horizontal coordinate. Part of the robot trajectory may include a horizontal segment when robot moves along the start line without departing from it. As in the idealistic case, a robot may return to and depart from the start line several times (see robot starting at position $x = 5$ on Fig. 6.4(g), 6.4(h)). Robots can also switch their vertical direction before reaching the start line (see the robot starting at position $x = 8$ on Fig. 6.4(k), 6.4(l)). If the robots meet, they initiate collision avoidance which may be repeated several times if their desired trajectories lie close to each other. The occlusion resolution described in Section 6.4.2 ensures that the robots keep moving even if they can not observe one or both of their neighbours, thus eventually restoring the line of sight. Once the robots reach the sorted order, there are no occlusions and robots can finish convergence by bringing their speeds to zero. The final configuration has robots rearranged in the sorted order along or close to horizontal start line. However, they jointly occupy horizontal positions that differ from those occupied by the team initially.

6.5 Discussion

Both idealistic and lifelike simulations result in successful sorting of the robots. However, in a lifelike simulation robots end up converging to a set of positions that differs from the original set. That agrees with theoretical considerations stated in Section 6.4.2 as departures from the perfect trajectory caused by speed limitations, occlusions and collision avoidance change the eigenvalues of matrix $H(t)$ in Eq. (6.1) and, thus, the set of final horizontal positions. Therefore, the limitations of real robots used in simulations break down the position set preservation property of the ideal sorting

system while still reaching the goal of sorting robots.

Another difference between the idealistic and life-like simulation is the convergence time. Life-like simulations take more time to converge due to the speed limitations, time spent on collision avoidance, and time spent moving during the occlusions when trajectory may stray away from the convergence path. For example, it takes approximately 10 seconds for idealistic system to sort robots under conditions shown on Fig. 6.4(a), while life-like Stage simulation takes about 90 seconds. With more robots (see Fig. 6.4(e)) the idealistic system still takes 10s to sort robots while the life-like simulation takes approximately 300 seconds to converge. A similar difference is observed under different initial conditions (see Fig. 6.4(i)) where the idealistic system converges in 7 seconds, while the life-like system sorts robots in 250 seconds. Fiducial sensor range imposes another limit on the practicality of this system as an increase in the team size will lead to an increase in the distance between robots which will eventually exceed the fiducial sensor range.

Formal analysis of the behaviour of this system, including convergence properties is desirable but presents difficulties. While including non-holonomic driving and noisy sensors and controls into the formal model of the system seems tractable, sensor occlusions and collision avoidance present a major obstacle. Sensor occlusions result in a non-smooth changes in the robot controls and require analysis that is significantly more complicated than the analysis of the double flow system itself. Likewise, the simple threshold based obstacle avoidance algorithm we employ introduces non-smooth transitions into the system which are further complicated by a random selection of the collision avoidance duration. Even if more tractable deterministic repulsive potentials are used for collision avoidance, convergence analysis of the flow based system is prohibitively challenging (Zavlanos and Pappas, 2008).

Due to the slow convergence observed in simulations the described system should be viewed as a proof of concept rather than a suggested practical solution. While this system can definitely be used in situations where restrictions on the robot communications and sensing preclude using other sorting methods, it is desirable to find ways to accelerate the convergence and make extensive experimental evaluation of the modified system before it could be recommended as an engineering recipe.

6.6 Summary

We described a multi-robot system sorting controller based on a smooth Brockett double-bracket flow equation. This controller provides a novel demonstration of computational capabilities of multi-robot systems solving a combinatorial problem by means of continuous dynamics. The controller integrates the Brockett system with non-holonomic steering, simple collision avoidance and sensor occlusion resolution. This strength of this approach is in the fact every robot needs to identify only two or one other robots in the team. No conventional pairwise robot-robot comparisons that standard sorting algorithms suggest are necessary. The robots are reactive agents with information between robots exchanged by means of relative position sensing. Robots have no global knowledge of the system state and very limited memory capacity (memory is used only for dispersal and collision avoidance timers and queue position status). These modest information processing requirements for robots come at a cost of slow convergence and a potential need for long distance sensing which is the weakness of this approach.

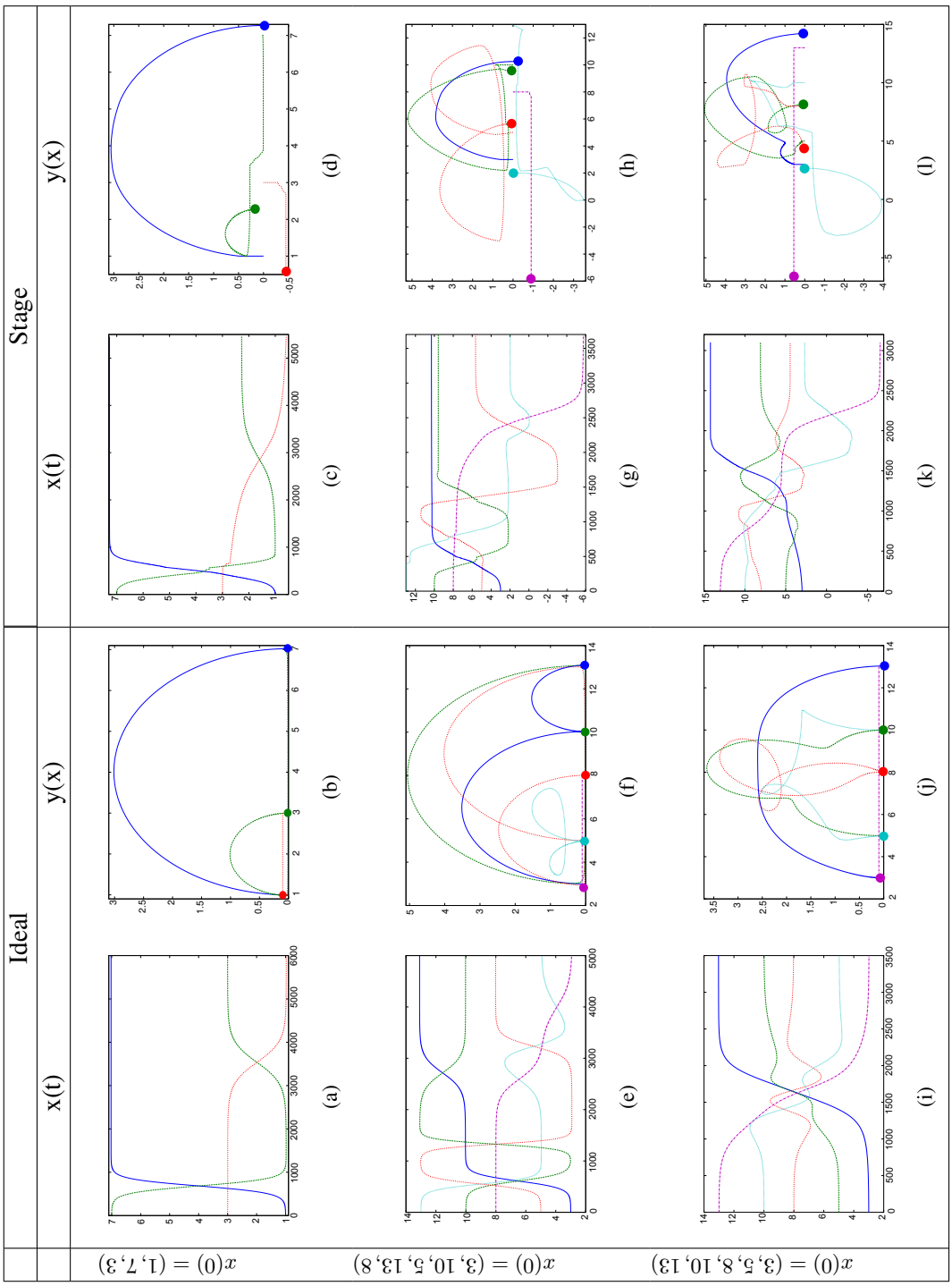


Figure 6.4: Robot trajectories for different initial conditions (rows). $x(t)$ graphs plot x coordinate (vertical axis) against time. $y(x)$ graphs plot y coordinate (vertical axis) against x-coordinate. Disks denote ends of trajectories on $y(x)$ graphs.

Chapter 7

Conclusion

7.1 Summary

In this thesis we argued that physically embedded agents can use embodiment and spatial embeddedness as a surrogate for computational resources, relieving communication and shared memory requirements for parallel computation. Specifically, this applies to distributed gradient optimization algorithms. We have shown that intermediate stages of the gradient descent process can be manifested by the locations of the agents, instead of being represented symbolically.

We applied this vision to a practical task of finding an energy optimal route for a team of robots to rendezvous with a dedicated tanker robot. We have shown, that this difficult task can be solved by a simple computer capable of adding vectors and comparing the norm of the sum with a single number. Embedding this computer as a controller into multiple robots allows the resulting system to approximate a solution to a complex optimization problem of finding an optimal route. Thus, a system as a whole uses its own physical dynamics to assist the parallel computing process that solves the multi-robot single-point and multi-point rendezvous tasks. Meanwhile, a distributed control algorithm used in this system has modest computational and communication requirements. These results provided examples of running computations, described by continuous processes (or discretization of such) on a complex physical system.

Next we have shown that computation required to solve a problem with a discrete solution set can also be non-trivially executed by a multi-robot system. We described the first robotic system that solves a combinatorial computational problem by means of its own continuous dynamics. The goal of the system is to rearrange a set of robots on a line in a certain predefined order, thereby

sorting them. While conventional discrete algorithms suggest pairwise between-robot rank comparisons, our system couples robots according to Brockett double bracket flow system and relies on the dynamics of this smooth system for sorting. A conventional multi-robot simulation with non-holonomic driving, noisy sensor data, collision avoidance and sensor occlusions suggests that this flow system can withstand perturbations introduced into the ideal dynamics by the physical limitations of real robots.

7.2 Future work

Multi-robot rendezvous results can be extended in several directions. The definition of the problem can be amended to include multiple tankers, adding an interesting coordination challenge. Also of interest are more complex cost functions, particularly those involving time constraints, joint time-and-energy constraints, and the dynamics of the robots. A challenging extension is to find good solutions to the general Frugal Feeding Problem, including the order of robot meetings. Finally, it would be useful to evaluate the Frugal Feeding Heuristic in dynamic and uncertain environments.

Robot sorting problem offers another set of research opportunities. The first opportunity is performance improvement of the robot sorting controller. This includes devising faster and more reliable collision avoidance strategies, trying to include some feedback mechanisms to correct changes in eigenvalues and thus preserve the set of original horizontal positions, looking for ways to bring down the system convergence time and eliminate the need for long-distance sensing. The second opportunity is looking for other ways to embed the Brockett sorter into a multi-robot system. While this controller uses the vertical coordinate of the robot as a non-diagonal entry in the H matrix, other modalities may be used. For example, robots can display values of non-diagonal entries by emitting sound, or changing the colour or intensity of a display light. Also, more state variables may be involved by considering non-tridiagonal matrices H. Finally, as double-bracket flow systems' computational capabilities include more than sorting, these capabilities should be evaluated in the context of multi-robot systems.

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

Controlling multi-robot teams is often a challenging task. Part of this challenge comes from the complexity of interactions between team members. Fortunately, these complex interactions themselves sometimes can be the key ingredient in simple and elegant solutions to control problems.

Bibliography

Adamatzky, A., editor (2002). *Collision-based computing*. Springer-Verlag, London, UK.

Adamatzky, A. (2004). Collision-based computing in Belousov-Zhabotinsky medium. *Chaos, Solitons and Fractals*, 21(5):1259 – 1264.

Adamatzky, A., de Lacy Costello, B., Melhuish, C., and Ratcliffe, N. (2004). Experimental implementation of mobile robot taxis with onboard belousov-zhabotinsky chemical medium. *Materials Science and Engineering: C*, 24(4):541 – 548.

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024.

Agre, P. E. and Chapman, D. (1987). Pengi: an implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272.

Ando, H., Oasa, Y., Suzuki, I., and Yamashita, M. (1999). A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robot. Automat.*, 15(5):818–828.

Arkin, R. C. and Mackenzie, D. C. (1994). Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation. In *International Symposium on Robotics and Manufacturing, Maui, HI*, pages 5–12.

Asai, T., de Lacy Costello, B., and Adamatzky, A. (2005). Silicon implementation of a chemical reaction-diffusion processor for computation of voronoi diagram. *I. J. Bifurcation and Chaos*, 15(10):3307–3320.

Bahceci, E., Soysal, O., and Sahin, E. (2003). A review: Pattern formation and adaptation in multi-robot systems. Technical Report CMU-RI-TR-03-43, Carnegie Mellon University.

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

Bellman, R. (2003). *Dynamic programming*. Dover Publications.

Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for realtime multi-agent navigation. In *Proc. of IEEE Conference on Robotics and Automation*.

Berger, R. (1966). The undecidability of the domino problem. *Memoirs American Mathematical Society*, 66:1–72.

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

Bloch, A. M., Brockett, R. W., and Ratiu, T. S. (1990). A new formulation of the generalized Toda lattice equations and their fixed point analysis via the momentum map. *Bulletin of the American Mathematical Society*, 23(2):477–485.

Bloch, A. M. and Crouch, P. E. (1999). Optimal control, optimization, and analytical mechanics. In *Mathematical control theory*, pages 268–321. Springer-Verlag New York, Inc., New York, NY, USA.

Boltyanski, V., Martini, H., and Soltan, V. (1999). *Geometric methods and optimization problems*. Kluwer Academic Publishers.

Bongard, J. C. and Paul, C. (2001). Making evolution an offer it can't refuse: Morphology and the extradimensional bypass. In *ECAL '01: Proceedings of the 6th European Conference on Advances in Artificial Life*, pages 401–412, London, UK. Springer-Verlag.

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

Bourgault, F., Makarenko, A., Williams, S., Grocholsky, B., and Durrant-Whyte, H. (2002). Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94.

Breitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. MIT Press, Cambridge, Massachusetts.

Brockett, R. (1988). Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 799–803 vol.1.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

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

Brooks, R. A. (1991). Intelligence without reason. In Myopoulos, J. and Reiter, R., editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Brooks, R. A. and Lozano-Perez, T. (1983). A subdivision algorithm in configuration space for findpath with rotation. In *IJCAI'83: Proceedings of the Eighth international joint conference on Artificial intelligence*, pages 799–806, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Calamai, P. H. and Conn, A. R. (1980). A stable algorithm for solving the multifacility location problem involving euclidean distances. *SIAM Journal on Scientific and Statistical Computing*, 1(4):512–526.

Chazelle, B. (2009). Natural algorithms. In Mathieu, C., editor, *SODA*, pages 422–431. SIAM.

Chemero, A. (2009). *Radical Embodied Cognitive Science*. The MIT Press.

Choset, H. (1996). Sensor based motion planning: The hierarchical generalized voronoi graph. Technical report, Carnegie Mellon University.

Clark, A. (1999). An embodied cognitive science? *Trends in Cognitive Sciences*, 3(9):345–351.

Conrad, M. (1990). The geometry of evolution. *Biosystems*, 24(1):61–81.

Cortes, J., Martinez, S., and Bullo, F. (2006). Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Trans. Automat. Contr.*, 51(8):1289–1298.

Drezner, Z., editor (1995). *Facility location. A survey of application and methods*. Springer-Verlag.

Dudek, G. and Roy, N. (1997). 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.

Erdmann, M. and Lozano-Perez, T. (1986). On multiple moving objects. *Algorithmica*, 2:1419–1424.

Fikes, R., Hart, P., and Nilsson, N. (1972). Learning and executing generalized robot plans. Technical Report 70, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025. SRI Project 1530.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208.

Fredkin, E. and Toffoli, T. (1982). Conservative logic. *Int. J. Theor. Phys.*, 21:219–253.

Gács, P., Kurdyumov, G. L., and Levin, L. A. (1978). One dimensional uniform arrays that wash out finite islands. *Problemy Peredachi Informatsii*, 12:92–98.

Gerkey, B., Vaughan, R., Sty, K., Howard, A., Sukhatme, G., and Mataric, M. (2001). Most valuable player: A robot device server for distributed control. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Wailea, Hawaii*, pages 1226–1231.

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

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

Hamann, H. and Wörn, H. (2008). Aggregating robots compute: An adaptive heuristic for the euclidean steiner tree problem. In *Proceedings of the Tenth International Conference on Simulation of Adaptive Behavior (SAB'08)*, LNAI. Springer.

Holland, O. (2007). A strongly embodied approach to machine consciousness. *Journal of Consciousness Studies*, 14:97–110(14).

Hwang, F., Richards, D., and Winter, P. (1992). *The Steiner Tree Problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland.

J. Lin, A. M. and Anderson, B. D. O. (2003). The multi-agent rendezvous problem. In *Proceedings of 42th IEEE Conf. Decision and Control*, pages 1508–1513, Maui,Hawaii.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98.

Kodama, Y. and Shipman, B. (2008). The finite non-periodic Toda lattice: A geometric and topological viewpoint. Technical Report arXiv:0805.1389. The contents of this article will be included in, and linked to, MEMPhys (Modern Encyclopedia of Mathematical Physics), to appear on Springerlink and in print in 2010.

Koren, Y., Member, S., and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1404.

Kuffner, J. J. and LaValle, S. M. (2005). An efficient approach to path planning using balanced bidirectional RRT search. Technical Report CMU-RI-TR-05-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

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

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

Lanthier, M. A., Nussbaum, D., and Wang, T.-J. (2005). Calculating the meeting point of scattered robots on weighted terrain surfaces. In *CATS '05: Proceedings of the 2005 Australasian symposium on Theory of computing*, pages 107–118, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Latombe, J. C. (1991). *Robot motion planning*. Kluwer, Boston, MA.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

Lavalle, S. M. and Kuffner, J. J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308.

Li, Y. (1995). A newton acceleration of the Weiszfeld algorithm for minimizing the sum of euclidean distances. Technical report, Cornell University, Ithaca, NY, USA.

Litus, Y. and Vaughan, R. (2008). Distributed gradient optimization with embodied approximation. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 359–365. MIT Press, Cambridge, MA.

Litus, Y. and Vaughan, R. T. (2010). Fall in! sorting a group of robots with a continuous controller. In *Proceedings of the Canadian Conference on Computer and Robot Vision*.

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

Litus, Y., Zebrowski, P., and Vaughan, R. T. (2009). A distributed heuristic for energy-efficient multirobot multiplace rendezvous. *Robotics, IEEE Transactions on*, 25(1):130–135.

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

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Trans. Comput.*, 32(2):108–120.

Lundh, T. (2007). A quantification of the morphological computations in perception systems. In *Proc. Int. Conf. on Morphological Computation*.

Mataric, M. J. (1990). A distributed model for mobile robot environment-learning and navigation. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Mataric, M. J. (1994). Interaction and intelligent behavior. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Mataric, M. J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:323–336.

Mathews, J. H. and Fink, K. K. (2004). *Numerical Methods Using Matlab*. Prentice-Hall Inc., 4 edition.

Mazer, E., Ahuactzin, J. M., Talbi, E.-G., and Bessiere, P. (1993). The Ariadne's clew algorithm. In *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior*, pages 182–188, Cambridge, MA, USA. MIT Press.

Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: symbols and search. *Commun. ACM*, 19(3):113–126.

Nilsson, N. (1984). Shakey the robot. Technical Report 325, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025.

O'Donnell, P. A. and Lozano-Pérez, T. (1989). Deadlock-free and collision-free coordina- tion of two robot manipulators. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 484–489.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420.

Paul, C. (2004). Morphology and computation. In *Proc. Int. Conf. on Simulation of Adaptive Behavior*, pages 33–38. MIT Press.

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

Pfeifer, R. (1996). Building "Fungus eaters": Design principles of autonomous agents. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior SAB96 (From Animals to Animats*, pages 3–12. MIT Press.

Pfeifer, R., Iida, F., and Bongard, J. (2005). New robotics: Design principles for intelligent systems. *Artif. Life*, 11(1-2):99–120.

Pfeifer, R., Iida, F., and Gómez, G. (2006). Morphological computation for adaptive behavior and cognition. *International Congress Series*, 1291:22–29.

Polani, D., Sporns, O., and Lungarella, M. (2006). How information and embodiment shape intelligent information processing. In Lungarella, M., Iida, F., Bongard, J. C., and Pfeifer, R., editors, *50 Years of Artificial Intelligence*, volume 4850 of *Lecture Notes in Computer Science*, pages 99–111. Springer.

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

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

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA. ACM.

Rimon, E. and Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8:501–518.

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

Russell, S. J. and Norvig (2003). *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall.

Saxena, N. and Clark, J. (1995). Analogue system for eigenvalue computation and sorting based on an isospectral matrix flow. *Electronics Letters*, 31(1):24–26.

Schlude, K. (2003). From robotics to facility location: Contraction functions, Weber point, convex core. Technical Report 403, Computer Science, ETHZ.

Schoppers, M. J. (1987). Universal plans for reactive robots in unpredictable environments. In *IJCAI'87: Proceedings of the 10th international joint conference on Artificial intelligence*, pages 1039–1046, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Simeon, T., Laumond, J.-P., and Nissoux, C. (20 December 2000). Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14:477–493(17).

Simon, H. A. (1969). *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, first edition.

Smith, S., Broucke, M., and Francis, B. (2007). Curve shortening and the rendezvous problem for mobile autonomous robots. *IEEE Trans. Automat. Contr.*, 52(6):1154–1159.

Strandberg, M. (2004). Augmenting RRT-planners with local trees. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3258–3262.

Snchez, G. and Latombe, J.-C. (2001). A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *In Int. Symp. Robotics Research*, pages 403–417.

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

Toda, M. (1967). Vibration of a chain with nonlinear interaction. *Journal of the Physical Society of Japan*, 22(2):431–436.

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

Ulrich, I. and Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1572–1577.

van den Berg, J., Guy, S. J., Lin, M. C., and Manocha, D. (2009). Reciprocal n-body collision avoidance. In *Proc. of International Symposium on Robotics Research*.

van den Berg, J. and Overmars, M. (2005). Prioritized motion planning for multiple robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2217–2222.

Varela, F., Maturana, H., and Uribe, R. (1974). Autopoiesis: The organization of living systems, its characterization and a model. *Biosystems*, 5(4):187 – 196.

Vaughan, R., Stoy, K., Sukhatme, G., and Mataric, M. (2000). Go ahead, make my day: Robot conflict resolution by aggressive competition. In *Proceedings of the 6th International Conference on the Simulation of Adaptive Behaviour*, pages 491–500.

Vaughan, R. T. (2008). Massively multi-robot simulations in Stage. *Swarm Intelligence*, 2(2-4):189–208.

Wang, C. (1975). On the convergence and rate of convergence of an iterative algorithm for the plant location problem. *Qufu Shiyun Xuebao*, 2.

Wang, H. (1961). Proving theorems by pattern recognition ii. *Bell Systems Tech. J*, 40:1–41.

Wang, H. (1963). Dominoes and the AEA case of the decision problem. In Fox, J., editor, *Mathematical Theory of Automata*, pages 23–55. Polytechnic Press, Brooklyn, New York.

Weiszfeld., E. (1937). Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Math. J.*, 43:355–386.

Winfree, E. (1998). Simulations of computing by self-assembly. Technical Report TR 1998.22, California Institute of Technology, Pasadena, CA.

Winfree, E. (2003). DNA computing by self-assembly. *The Bridge*, 33(4):31–38.

Winfree, E., Liu, F., Wenzler, L. A., and Seeman, N. C. (1998). Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544.

Xue, G.-L. (1987). A fast convergent algorithm for $\min \sum_{t=1}^{m} c_t||x - a_t||$ on a closed convex set. *Journal of Qufu Normal University*, 13(3):15–20.

Zavlanos, M. and Pappas, G. (2006). A dynamical systems approach to weighted graph matching. In *Decision and Control, 2006 45th IEEE Conference on*, pages 3492–3497.

Zavlanos, M. and Pappas, G. (2008). Dynamic assignment in distributed motion planning with local coordination. *Robotics, IEEE Transactions on*, 24(1):232–242.

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

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

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

Ziemke, T. (2007). The embodied self : Theories, hunches and robot models. *Journal of Consciousness Studies*, 14:7, s. 169-179.

Zuluaga, M. and Vaughan, R. (2005). Reducing spatial interference in robot teams by local-investment aggression. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta.