TASK-SWITCHING FOR SELF-SUFFICIENT ROBOTS

by

Jens Wawerla

Dipl. Inf. (FH) University of Applied Sciences Konstanz, Germany, 2002

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
Summer 2010

APPROVAL

Name: Jens Wawerla

Degree: Doctor of Philosophy

Title of Thesis: Task-Switching for Self-Sufficient Robots

Examining Committee: Dr. Arthur Kirkpatrick
Professor of Computing Science, SFU
Chair

Dr. Richard T. Vaughan, Senior Supervisor
Professor of Computing Science, SFU

Dr. Robert F. Hadley, Supervisor
Professor of Computing Science, SFU

Dr. Ronald C. Ydenberg, Supervisor
Professor of Biological Sciences, SFU

Dr. Alexandra Fedorova, SFU Examiner
Professor of Computing Science, SFU

Dr. Gaurav S. Sukhatme, External Examiner
Professor of Computer Science,
University of Southern California

Date Approved: 1 June 2010

Declaration of
Partial Copyright Licence

Abstract

Task-switching enables a system to make its own decisions about which task to perform. It is therefore a key ability for any truly autonomous system. Common task-switching methods range from computationally expensive planning methods to often suboptimal, minimalistic, heuristics. This thesis takes a bio-inspired approach, motivated by the fact that animals successfully make task-switching decisions on a daily basis. The field of behavioural ecology provides a vast literature on animal task-switching. Generally these works are descriptive models of animal behaviour, either modelling to fit the data from observed animal behaviour, or theoretically optimal models of how animals ought to behave. But what is needed in robotics are methods that generate behaviour based on the information available (due to sensing) to the robot. Furthermore these methods have to take the physical limitations (velocity, acceleration, storage capacity etc.) of the robot into account. This thesis takes inspiration from descriptive behavioural ecology models and proposes a situated and embodied task-switching method suitable for mobile robots. To evaluate the quality of the decisions an objective function is needed. Reproductive success is commonly used in Biology, here economical success is used. We illustrate the applicability of the proposed methods on Toda's *Fungus Eater* robot. The decisions this robot faces are (1) when to work and when to refuel and (2) where to work or refuel respectively. Both decision types are essential to any autonomous, mobile robot. The proposed task-switching methods are based on Optimal Foraging Theory, in particular on rate-maximization and the Marginal-Value Theorem.

To my parents

"I may not have gone where I intended to go,
but I think I have ended up where I needed to be."

— Douglas Adams

Acknowledgments

Writing a dissertation is a matter of a few month but the work that leads up to it takes years. Years of hard work, which I would not have been able to do without the loving support from Sarah Brown. She gave me the strength and the confidence to overcome the great many challenges of a Ph.D. program and finish this work.

I am very thankful to Richard Vaughan, my senior supervisor, for his support and all the discussions during meetings, over coffee and at the pub, which helped to form me as a scientist. I am very fortunate to have had an advisor like Richard to guide me through the academic and personal challenges of a Ph.D program. I am also very grateful to Yaroslav Litus for sharing the Ph.D student life with me in the Autonomy Lab. Being a Ph.D student can in times be stressful in many respects, being able to share the experience is invaluable. I am equally thankful to Greg Mori for his friendship and his encouragement.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Whenever we look around us, we observe animals and plants concerned with resource management. The set of resources an animal or plant is interested in varies and ranges from food, water, space, shelter to reproduction opportunities. The laws of thermodynamics make it essential that any biological system obtains energy. For now, it is irrelevant whether this energy is gained from other biological systems, chemical processes, light, or any other way. Without energy, no system, artificial or natural, can perform work, where here we mean work in the context of physics. When a system has no usable energy, this leads to death in biological systems and must be avoided by the system. Let us look at a brief example to clarify the difference between usable and unusable energy: a tree that gains energy from sunlight via photosynthesis will eventually run out of energy and die if we replant it in a dark environment. The wood, however, still contains quite a bit of energy that we can use to heat our homes, but the energy contained in the wood cannot be used by the tree.

Access to energy can be a challenge for any agent, biological or artificial. Usually food (energy) does not just come to the agent, but the agent has to actively search, collect and process items to obtain energy. For some systems, e.g. plants, the process of obtaining energy does not appear to require complex decision making. Yet other agents developed highly sophisticated behaviours to ensure access to energy. For example, Cristol and Switzer (1999); Switzer and Cristol (1999) report American crows *Corvus brachyrhynchos* not only drop walnuts onto hard surfaces, a behaviour not unique to just crows, but they adjust the height of the drop according to the hardness of the shell as well as the hardness of the ground surface.

As Pfeifer and Scheier (2001) put it: "The ability to survive in complex environments is given for all biological systems. Achieving this ability in artificial agents turns out to be an extremely hard problem."

This provides us with the motivation to investigate resource management in agents in general and energy management in robots in particular. The latter is of special interest due to the inevitable consequences any agent faces if lacking energy - no mistakes allowed. This presents an interesting action-selection problem for agents who must make decisions with regard to energy management. In order for an agent to face an actual action-selection problem, the agent must be able to choose between at least two actions. The challenge in action-selection problems lies in the fact that trial and error learning methods, such as genetic algorithms or reinforcement learning methods, are not suitable during the lifetime of a single agent because of the inherent threat of not obtaining enough energy in the error case.

It is worthwhile to define what is meant by the term "resources", especially since the word is used in different domains. Every agent can be described by a set of hidden or observable state variables that defines its situation in the world (McFarland and Houston; 1981). Such state variables can be blood oxygen, body fat, money etc.. Within this thesis, we define resources as items provided by the environment or other agents that satisfy a need for at least one of the state variables. There may exist resources that will increase the level of several state variables, e.g. eating an apple provides water and fructose to an animal eating it. In turn, different resources might influence the same state variable, e.g. AA batteries and AAA batteries provide electrical energy to a robot, but since the amount of energy provided is different, the action-selection mechanism should treat them separately.

Further, we can identify two different types of resources, *essential resources* and *dispensable resources*. *Essential resources* are necessary for continued agent function. If the level of an essential resource, e.g. oxygen, water, or food, drops below a certain threshold at any given time, the result is death of the agent. *Dispensable resources* are desirable to the agent but a lack of them does not cause any threat to the agent. Examples of dispensable resources include mates, or nesting materials.

With these definitions, we can clarify two phrases often used interchangeably in robotics literature, *self-sufficient* and *autonomous*. Some authors use the phrase "energy autonomy" to differentiate between two very different things - autonomy and self-sufficiency. By the

term *self-sufficient agent*, we mean an agent that satisfies its demand for all essential resources at every moment in time, over an extended period of time, in a given environment. Therefore, a solar powered robot during the arctic summer is self-sufficient in the most trivial way. A human being without a spacesuit on Mars is not self-sufficient at all. *Autonomy* however refers to controllability of the agent. As McFarland (1994a) put it, "... an autonomous agent must have some degree of motivation and cognition, organized in such a way that an outside agent cannot obtain sufficient knowledge to control the autonomous agent"

From these definitions, it becomes clear that a self-sufficient robot does not necessarily have to have a high degree of autonomy, e.g. the Mars Exploration Rovers provide themselves with solar energy but only make a move if told so by mission control in Pasadena, CA. On the other hand, to achieve a high degree of autonomy a robot must be self-sufficient. If it were not self-sufficient but would depend on a third party, e.g. a human operator, to refuel it, the robot would be subject to external control and hence would be less autonomous. Therefore we focus on self-sufficiency now in order to achieve autonomy in the future. We are particularly interested in

- how to build a robotic system that achieves self-sufficiency
- the control strategies and action-selection mechanisms for such a system, especially in the face of uncertainty and partially unknown environments
- the optimality of the resulting behaviour
- the underlying principles of resource management

In what is to follow, we have a close look at the relevant literature in the fields of robotics, ecology, and engineering. We start by introducing *The Fungus Eater* as a framework for this investigation. Next we discuss related literature on action-selection in ecology and techniques employed in the domain of robot energy management. Then we introduce a descriptive framework for the task-switching problem. The framework describes the work-fuel cycle each robot is subject to. In contrast to McFarland and Spier (1997) basic utilities cycles, we explicitly describe the entire problem space, including reward functions and switching cost. The framework allows us to phrase task-switching problems with different levels of difficulty.

We then outline robot controllers for three of those problems. The first controller handles situations in which the robot has to execute a given sequence of tasks and decide when to refuel. The other control policies deal with discounted labour situations and with task-switching under diminishing returns. In an additional chapter we switch to a multi-robot problem and demonstrate team task-allocation. We conclude with a look at future work.

Chapter 2

Encounter on Taros

In the introduction, we motivated investigating robots concerned with resource management. We further argued that resource management requires an agent to deal with an elementary action-selection problem. This idea is not new, yet the action-selection problem is unsolved. In this chapter we introduce some of the previous work in resource related action-selection in robots.

2.1 The Design of a Fungus Eater

In 1962 Masanao Toda, a Japanese psychologist, proposed to study the *Fungus Eater* (Toda; 1962, 1982). His proposal was intended to give psychology research a new direction by suggesting investigation of complete agents in real world scenarios rather than isolated faculties under laboratory conditions. Toda's *Fungus Eater* is a humanoid robot sent to the distant, fictional planet Taros. Its task, and therefore its purpose in life, is to collect uranium ore. In order to achieve this task, it is equipped with a Geiger counter to detect the ore and with arms to collect ore. Further, the robot possessed the means for decision making, namely a computer. Since all actions, including "thinking" require energy, the *Fungus Eater* has to provide itself with energy which comes in the form of fungi that grow on Taros and which the robot collects, stores, and consumes. For that purpose, it is also equipped with a sensor to detect fungi. It will die of starvation should the robot ever run out of fungi.

What is interesting in Toda's proposal, is that this task is very simplistic: eat some fungi and use the resulting energy to perform labour, namely collect ore. The conflict

arising from the need to execute two mutually exclusive tasks, collecting ore and collecting fungus, provides a non-trivial action-selection problem with no known, generally applicable, optimal solution. The analogy to biology is very apparent. Animals face the same problem on a daily basis. While animals generally do not collect ore, they have to spend time keeping clean, building shelters, displaying, feeding their offsprings etc., tasks we summarize as work.

In his book, Toda suggests using the *Fungus Eater* as a way to do experimental studies with human subjects by giving them the task of teleoperating a *Fungus Eater* and observing them while they do so. He also attempts to analyze the Fungus Eater Problem and sketches control programs for the *Fungus Eater*. He starts by using a value iteration approach, by which the robot selects the course of action that leads to the highest expected reward (see Stuart and Peter (2003) for an introduction to value iteration). It is fairly straightforward to calculate the expected reward with respect to a known, fully observed environment. Although this might be challenging due to combinatorial problems, the main problem with this approach is the fact that it requires knowledge of a function that maps current fungus storage to the expected reward gained. Obtaining such a function in a dynamic environment is not trivial and Toda does not give a solution to this problem. Reinforcement learning method may be used to estimate this value function (see Sutton and Barto (1998) for details on reinforcement learning). But this introduces two new problems, (1) how to trade-off exploration and exploitation and (2) how to sufficiently explore the state space, including the low energy states, without actually running out of energy.

Later Toda attempts to employ utility theory to the decision making problem. He argues that ore has an absolute utility proportional to the amount of ore, where fungus has no absolute utility. Instead, fungus has an induced utility, meaning possessing fungus enables the robot to potentially gain ore, which has an absolute utility. Again, no implementable control is derived. Toda also argues strongly in favour of a learning mechanism since this would make the *Fungus Eater* adaptive to changes in the environment and it would avoid the need for the expected ore function, which as we have seen before, is difficult to acquire.

2.1.1 Critique

Toda (1982) has to be seen as a first attempt for a new direction of thought and thus may contain flaws. Toda also has formal training in psychology and is, therefore, writing for his field. His text and thoughts may not absolutely satisfy the needs of other disciplines like engineering or computer science. Therefore, since we are concerned with building a *Fungus*

Eater, it is important to point out the limits of Toda's proposed robot.

An important issue with the *Fungus Eater* is that the robot's fungus storage has no upper limit. This is not merely a minor lack of realism. Since the laws of physics require some limit on capacity, it has far reaching consequences for action-selection mechanisms. Without any upper limit on the fungus storage, a feasible policy is to first collect all fungi and then focus on ore. The benefit of this policy is that the robot has to only make one decision in its lifetime: at what point has it collected all fungus?

Toda (1982) is somewhat unclear about the value of ore collection to the robot. In his science-fiction style instructions to Toda's test subjects (human teleoperators), the amount of ore collected is directly proportional to a reward given to the operator. In the *Solitary Fungus Eater*, some utility is assigned to collected ore. We want to stress the point that it is very important to design the value system correctly (Pfeifer; 1996). E.g. a human operator might stop controlling the robot after she earned x dollars, because her internal utility function may assign the same value to x dollars as it does to $x + 1$ dollars. The action-selection mechanism needs some form of value system by which to evaluate policies, which might include a timely execution of the task. The *Solitary Fungus Eater* assigned the same utility to "collecting 1 kg of ore immediately" as it does to "sleep now and collect 1 kg of ore tomorrow". Wawerla and Vaughan (2008) analyze an abstract *Fungus Eater* model and show optimal recharging strategies in time-discounted labour situations (see also Ch 6). In a more pragmatic way, Pfeifer (1994) solves this problem by adding a simple constraint: the *Fungus Eater* faces unemployment if the rate of ore collected drops below a threshold set by the robot's owner. Here the underlying assumption is that unemployment has a negative impact on the robot's value system and thus must be avoided by the robot.

This brings us to a very important point. Robots are built with a purpose in mind. Their job might be to mine, entertain, defuse, measure, clean-up, explore, etc. but they have to perform the task they are built for. In order to do so, they might have to collect energy. Out of this arises the action-selection problem.

2.1.2 Fungus Eater Offspring

Nakahara and Toda (1964) take the *Fungus Eater* and, by constraining the problem domain, define the *Fungus-Eater Game*. This game has a known, fixed, finite number of rounds L. A certain amount of fungus is given to the *Fungus Eater* at the beginning of the game and more can be obtained if the *Fungus Eater* encounters fungus during the game. In each round the

Fungus Eater has to spend one unit of fungus and must choose amongst two actions, "focus on fungus" or "focus on uranium". After making a choice, it is stochastically determined, with known probability, whether the *Fungus Eater* encounters fungus, uranium, both, or nothing in this round. What resource the *Fungus Eater* obtains during a particular round depends on the selected action and the encountered resource. Game states are detailed in table 2.1. The game ends after L rounds or once the fungus storage is empty. The objective of the game is to collect as much uranium as possible.

Action	Encountered Resource			
	Fungus & Uranium	Fungus	Uranium	Nothing
Focus on Fungus	a fungus	a fungus	1 uranium	$\emptyset$
Focus on Uranium	1 uranium	a fungus	1 uranium	$\emptyset$

Table 2.1: Payoff matrix for the Fungus-Eater Game

Nakahara and Toda (1964) analyze the game and outline a numerical method to find the optimal strategy. This method is based on maximizing the expected future uranium return based on how much fungus is in storage and how much fungus is needed to survive to the end of the game.

This formal game is abstracted from the original robot sent on its mission to a distant planet. Sensing and locomotion have been abstracted away and the behavioural switching cost has been removed. But there is yet a larger problem. By rewarding the robot even if the chosen resource is not encountered, the decision problem is greatly reduced or even removed completely, depending on the configuration. An example from Nakahara's paper, $L = 31$ rounds are to be played, the probability of encountering fungus $P(F) = 0.4$, the probability of encountering uranium $P(U) = 0.5$, a fungus is worth $a = 3$ futs, and the robot starts the game with $x = 9$ futs (a fut is the amount of fungus required for one round). This means the robot has to collect $y = (L - x)/a = 7$ fungi. The probability of encountering a fungus and at the same time not encountering a uranium is $P(F \wedge \neg U) = 0.2$. Hence, if the *Fungus Eater* plays a trivial strategy, e.g. always chooses the action *focus on uranium*, it will, on average, encounter 6 fungi, just one short of the required 7 to survive the game but with the advantage of not having missed a single uranium. This is not to say that this strategy is optimal. Rather, the argument is that the *Fungus-Eater Game* deviates greatly from the original *Fungus Eater* proposal. Hence solutions to the game may not be suitable for a robot on Taros.

Whether or not Toda's *Fungus Eater* contributed to progress in psychology, it inspired robotics researchers. Rolf Pfeifer at the University of Zürich, conducted experiments with a simulated *Fungus Eater* to investigate the emergence of emotions (Pfeifer; 1994). Two years later this work would result in a set of "design principles for autonomous agents". By investigating *Fungus Eater*, Pfeifer (1996) identified 10 principles for designing and building robots. This extended and helped to define the "New AI approach" more precisely. The new approach to AI was introduced by Brooks in mid 1980 (Brooks; 1986, 1990, 1991) to the field of robotics. In a nutshell, Pfeifer's principles are

- The complete agent principle
  the agent must be *autonomous* - function without human intervention, *self-sufficient* - sustaining all essential resources over an extended period of time, *embodied* - the agent has to be realized in the real world as a physical system, and *situated* - the agent itself has to control the whole interaction with the world

- The ecological niche principle
  an agent is always designed for a given environment and evaluating the agent's performance is only meaningful within this environment.

- The principle of parallel, loosely coupled processes
  cognition emerges from the interaction of a multitude of loosely coupled processes. This is along the lines of Brooks "subsumption architecture" (Brooks; 1986).

- The value principle
  an agent must have a value system by which the agent can evaluate the benefit of its actions and which provides some sort of 'motivation' for the agent to perform the desired task.

- The principle of sensory-motor coordination
  sensing, control and actuation have to be viewed as a whole rather than individual modules, this automatically leads to agents being grounded in their environment and their sensor-motor system.

- The principle of ecological balance
  sensors, actuators, and the controller must be balanced in their complexity.

- The principle of cheap design
  designs that capitalize on the system-environment interaction lead to simpler and, thus, cheaper designs

- The frame-of-reference principle
  from which point of view we are talking about the world - the agent's or ours? It is also concerned with the complexity of behaviour and the internal mechanisms that lead to a certain behaviour.

- Compliance with principles
  Unfortunately Pfeifer does not further describe this principle.

Over the years Pfeifer refined and rephrased the design principles. Noteworthy, Pfeifer and Gómez (2005) introduces the principle of redundancy, asking for a partial overlap of the agent functionality. This overlap should be based on different physical processes in order to increase the robustness of the agent with respect to failures within the agent and changes in the environment. Pfeifer and Scheier (2001) gives an elaborated, detailed account of the design principles and the ideas for embodied AI.

In addition to an early account of the design principles, Pfeifer (1994), presents the "Learning Fungus Eater" and the "Self-sufficient Fungus Eater". The first is a Khepera^{TM}robot controlled by an Artificial Neural Network (ANN) employing Hebbian Learning. Details are given in Pfeifer and Verschure (1992). Although learning to avoid obstacles and reaching a goal, this robot is not actually a *Fungus Eater*. As the author notes, the robot cannot sustain itself and "there is no need for it to eat fungus. In fact there is no need for it to do anything". This insight led to the "Self-sufficient Fungus Eater", a simulated robot that employs a simple policy: if the agent's energy level is higher than the amount of ore collected per unit time, it should focus on ore while at the same time ignore fungus and vica versa. The paper does not give any analysis of the quality of this policy but focuses on the emotional attributes observers assign to the agent.

At about the same time Wehrle (1994) investigated a group of *Fungus Eater* in a competitive setting. The outlined robot controller is called a "cybernetics system", basically a mixture of control theory and subsumption architecture. Little detail is given about the controller and qualitative results, let alone optimality analysis.

2.2 The Fungus Eater Problem - a Definition

In the previous section, we have seen numerous *Fungus Eater* robots from different research disciplines devised with different objectives in mind. We also argued that Toda's original description of the *Fungus Eater*, its task, and the reward structure are imprecise and thus do not provide a basis for an actual implementation. In this section we define the Fungus Eater Problem (FEP). This is a novel definition which not only includes the robot, but also the environment, the task, and the entrepreneur. The objective is to overcome the limitations of Toda's definition and to suggest the FEP as a generic robot resource management problem.

The *Fungus Eater* is a robot employed by an entrepreneur for the purpose of performing at least one primary task in which the entrepreneur is interested. While details of this task are secondary, it might be delivering mail, collecting items, exploring distant planets, guiding people through exhibitions, etc.. The primary task is the sole reason for the entrepreneur's investment in the robot (McFarland and Bösser; 1993). Therefore, the task, or more precisely the execution of the task, is the only reason for the *Fungus Eater*'s existence.

By the laws of thermodynamics, executing any task requires energy and the robot can carry only a finite amount of energy. To enable the robot to execute the primary task for extended periods of time, the robot must posses the means to extract energy from the environment and store it for later use. Note this does not necessarily imply that the robot opts for replenishing its energy storage. The robot might very well choose to not refuel and instead spend the energy otherwise needed to obtain more energy on the task at hand (Wawerla and Vaughan; 2008). Which option the *Fungus Eater* chooses solely depends on the robot's value system, but the option must be given to the *Fungus Eater*, in order for it to exhibit the full action-selection problem by having to choose between the task and refuelling. This choice between at least two mutually exclusive actions is at the heart of a FEP.

That said, we disagree with Spier and McFarland (1997), who argue that single-resource problems are unsatisfactory in terms of investigating decision-making and suggest to investigate two-resource problems instead. Our point of view is that it is not the number of resources that renders a decision problem easy or difficult. It is the number of mutually exclusive actions the decision maker has to choose from that determines the difficulty of the decision process. Certainly this number has to be at least two. A simple example will illustrate this argument. Consider a *Fungus Eater* that feeds on fungi and at the same time

is employed to collect fungi for the entrepreneur. Clearly this is a single resource problem, in which the robot's objective is to maximize the number of fungi collected. Even without having to transport the fungi to a central delivery point, the robot faces an interesting decision problem in cases where there is more than one depleting fungus patch in the environment. The challenge lies in the question of when to switch patches.

Spier and McFarland (1997) further argue that more insight into decision making cannot be gained by considering more than two resources. We are inclined to disagree. A robot collecting ore and feeding on fungus does not face the same combinatorial complexity issues of a robot which additionally requires water or may in addition be rewarded for collecting gold.

Following McFarland's ideas (McFarland and Houston; 1981) (q.v. 3.5.2), the *Fungus Eater* is defined by an internal state-space - a minimal set of state variables. These variables sit within an Euclidean vector space as its orthogonal axes. In contrast to McFarland, we impose a lethal lower and/or upper boundary only on a subset of state variables and allow others to grow unbounded: collected ore, merit points, or money, for example, should in most cases be allowed to grow unbounded. Other state variables might have a hard limit, e.g. battery capacity, but an attempt to exceed this limit does not harm the agent (other than possibly in form of lost opportunity cost).

The *Fungus Eater* has a set of actions at its disposal. As we mentioned earlier, the number of actions in this set has to be at least two, where those two actions cannot be executed at the same time in any meaningful way. Otherwise no decision making is required. At any moment in time, the agent has to perform at least one action and since we are interested in embodied agents, executing any action requires energy. Therefore each executed action will alter at least one state variable, namely the variable related to the robot's energy supply. This in turn means we do not allow any null action, that is an action that does not change any state variable at all. This does not rule out a sleep or wait action in which the agent does nothing else but spend a minimum of energy to stay "alive". Such an action can be useful in scenarios where the agent has to wait for a resource to be freed, e.g. the only existing charging station is currently occupied by another agent, or a solar power robot during the night.

We briefly mentioned a value-system. As Pfeifer (1996) puts it, "If the agent is to be autonomous and situated it has to have a means to judge what is good for it and what isn't". In addition the value system tells the robot what the entrepreneur expects it to

do. As we argued earlier robots are built and "employed" with a purpose in mind. In decision theory the value-system is referred to as utility and a rational agent behaves in a way that maximizes utility (Stuart and Peter; 2003). While it is widely agreed that utility functions are useful in guiding decisions of rational agents, the problem is that no generally applicable way of designing utility functions is known. In any case we have seen above that neither Pfeifer's (Pfeifer; 1994) nor Toda's *Fungus Eater* have a real incentive to execute the primary task in a timely manner. One solution is to discount the reward obtained by executing the primary task. This serves several purposes: (1) it enforces timely execution of the primary task, (2) it embeds the robot into the economic reality of its owner (who's finances are subject to discounting), and (3) it reflects the fact that real systems are always subject to a finite lifetime. By introducing discounting to the *Fungus Eater* we solve the fundamental limitation of the original proposed definition.

In accordance with Pfeifer's design principles, the *Fungus Eater* is designed for a certain environment and may not be operational in a different environment. Hence it is important to define the environment the *Fungus Eater* operates in. Certainly the number of possible environments is very very large and influenced by a multitude of parameters. Where clearly things like the layout of obstacles, width of doorways, steepness of the ground etc. are important, resources are the absolute focus of the FEP.

Resources are entities that, when consumed or collected by the *Fungus Eater*, change at least one state variable. Note this change may not only be positive. Resources might also decrease the value of a state variable, e.g. eating salty food makes you thirsty. In addition, the rate of change might also depend on the action selected to obtain the resource. For example, eating a soup with a spoon is usually slower than drinking the soup from a cup. To avoid confusion, we refer to resources when we mean the actual entity and we use resource type when we mean an entity class that is identifiable by the rate it changes a state variable. For example, AA and AAA batteries are considered different resource types since they provide electrical energy at different rates. This means the complexity of the FEP is governed by the number of resource types as well as the number of resources.

In addition, the distribution of resources is essential to the FEP. It is certainly possible to consider the location of each individual resource item, but it greatly reduces the complexity of the decision problem if it is possible to cluster resources spatially such that it is sufficient to just consider each cluster. Following the behavioural ecology convention

(Danchin et al.; 2008), we suggest clustering resources of each type so that the spatial density is approximately constant within the cluster. We refer to such a cluster as a patch. This includes single point patches e.g. a charging station. Note the underlying assumption is that all resources within a patch are equally beneficial to the *Fungus Eater* and their relative location to the robot has no influence on the robot's decision.

Often resources are influenced by the action of the *Fungus Eater*, that is, once a resource is consumed it is no longer available in the world. This leads to diminishing returns, which must be considered by the action-selection mechanism. Charnov (1976b) addresses diminishing returns in the marginal value theorem (q.v. 7.2). Other reasons for resource dynamics are "world player" processes - processes not under the control of *Fungus Eater*. This may be other non-cooperative agents, natural phenomena such as circadian or annual rhythms, growing/regrowing resources, or just "bad luck" externalities such as a power outage.

Knowledge about the distribution and dynamics of resources is a clear advantage for the *Fungus Eater*. Given enough time it is in principle possible to plan the optimal course of action when all necessary information is available. In practise, a robot often has only limited time and perfect information is not available. We discuss the common robotics literature on this topic in section 3.1. An interesting observation is that partial information not only makes the decision harder, it also introduces an additional action to choose from. With partial information, a rational action might be to explore the world and build a model of the resource distribution and dynamics. Clearly such a model is only useful if it leads to higher gain rates in the future that compensate for the cost of generating such a model. This introduces another series of interesting questions, how much should a forager spend on exploring the world? When is a model good enough to be used? How does one keep the model updated?

The last point worth discussing is centralized vs. decentralized foraging. Centralized foraging refers to situations in which the foraged goods have to be delivered to one central location. Decentralized foraging on the other hand refers to situations in which the foraged item can either be dropped off or consumed immediately after it has been discovered or the item is virtual and is not required to be delivered, e.g. merit points. Differentiating between the two types of foraging is useful because of the resulting implications. In centralized foraging, the forager has to consider the on-board storage capacity for the foraged item, an unnecessary consideration in the decentralized case. This requires the centralized forager to return to a certain location in the world periodically. This may seem to add complexity to

the decision making process, but, in fact, it introduces a central decision point from which every decision option can be compared, essentially eliminating the combinatorial planning problems a decentralized forager faces.

We defined the FEP as a generic robot resource management problem and discussed the most important aspects that influence the complexity of this problem. Next we take a look at related action-selection literature.

Chapter 3

Action-Selection

Every system, natural or artificial, that possesses at least one self-controllable subsystem, has to decide which subsystem to deploy when and how. As Maes (1989) puts it

> How can an agent select "the most appropriate" or "the most relevant" next action to take at a particular moment, when facing a particular situation?

This is arguably the most elementary question in a large number of research fields, from ethology, to artificial intelligence, to robotics. Hence the literature is vast and a detailed review beyond the scope of this document. Instead we highlight works directly related to self-sufficiency decisions and task-switching.

3.1 Planning

The traditional or "good old fashioned" approach to action-selection in Artificial Intelligence (AI) and robotics is planning. In planning, the agent uses all data necessary and selects its course of action according to some optimality criterion, which usually includes detailed models of the world and the interaction of objects within the world. LaValle (2006) is good introduction to planning algorithms.

A famous robotics example is Shakey (Nilsson; 1984). Its planning software, STRIPS, uses a complete world model to plan, e.g. Shakey's traverse to a goal location. Planning in Shakey is slow and every unaccounted for or new situation requires re-planning.

Another robot that heavily relies on planning is Hyperion (Tompkions et al.; 2002; Wettergreen et al.; 2002, 2005). This robot is solar powered with its solar cell array mounted at fixed angles on the robot. Therefore, it is essential for the robot to move in such away as to maximally expose its solar cells to the sun as the star, relatively speaking, moves across the sky. To do so, Hyperion has a detailed elevation map of the area and an accurate motion model of the sun. Using ray-tracing, the illumination of the solar cells can be predicted for any location at any point it time. Using a planner based on the D* planning algorithm, Hyperion plans an energy-efficient path. It demonstrated sun-synchronous exploration for 9.1 km over the course of 24h during the arctic Canadian summer.

The Mars Exploration Rovers (Wikipedia; 2008) demonstrate long-term self-sufficiency and "autonomous" exploration of Mars. Since the rovers collect energy using solar cells, they are self-sufficient by design, but their action-selection mechanism is not autonomous at all. Spirit and Opportunity are merely remote-controlled by a group of NASA engineers on Earth. To support the engineers in activity planning for the robots, MAPGEN (Bresina et al.; 2005), a planning tool, is used.

Common problems with planning systems are the computational complexity usually due to combinatorial issues and the reliance on detailed world models, which may not be accurate or may be out of date (Brooks; 1990, 1991).

3.2 Reactive Systems

An opposite approach is to do no planning whatsoever, to "just" react to the current sensor stimulus. This has three main advantages (1) decreased computational complexity since the sensed data is usually of lower dimensionality than data from elaborate world models. (2) since the action-selection is based on current sensor reading and a decision is made at every time step, no prolonged re-planning is required in case a change in the environment is detected, and (3) reactive agents do not require world models, which are difficult to obtain and maintain (Brooks; 1990, 1991). As an interesting side note, one of the first electric, mobile robots (Walter; 1963) also used a reactive control system, although somewhat more in the spirit of Braitenberg (1986).

A famous representative of this school of thought is the subsumption architecture (Brooks; 1986). Ghengis, a six legged robot built by Brooks (1989), demonstrated that this reactive approach is capable of controlling very complex machines in challenging environments. This

approach is not without its limitations. Purely reactive systems do not utilize past experiences nor do they consider future consequences of their actions. For example the fact that a charging station is available at the end of the corridor is not considered by the reactive system since the agent cannot observe this station while being inside one of the rooms along the corridor. Hybrid systems were introduced to solve the persistence problem. Connell (1992), for example suggests a three layer system, where the bottom layer is a servo layer, basically on a control theory level. Next is a subsumption level, monitoring the sensors and making in-time decisions of how to react. On top is a symbolic layer, processing events from the lower levels and setting the general direction by parameterizing the lower levels.

Many of the early hybrid architectures suffered from sensor noise and thus noisy world models. Probabilistic approaches aim at solving the uncertainty problems by maintaining beliefs of the state of the world instead of axiomatic fact. Early, successful obstacle avoidance algorithms like VFH (Ulrich and Borenstein; 1998) use histogram grids to obtain more reliable obstacle information from noisy sensors. A great introduction to probabilistic robotics is provided by Thrun et al. (2005).

For a more detailed review of action-selection mechanisms (ASMs) for robotics, refer to Pirjanian (1999) review paper. Summarizing, we can say, the holy grail of action-selection mechanisms has not yet been found. Next we have a closer look at ASMs directly aiming at self-sufficiency.

Examples of reactive self-sufficient robots include Silverman et al. (2002, 2003), where a fixed energy threshold is used. Whenever the battery voltage drops below a fixed threshold the robot selects control structures that make it approach and dock on a charging station. Others like Austin et al. (2001) used a time based decision criterion, the robot would dock and recharge after a certain amount of operational time has passed. Slightly more sophisticated are systems that integrate the current flowing in and out of the battery and thus obtaining an estimate of the remaining battery charge and returning to the charging station once the battery charge drops below a threshold. Hada and Yuta (1999) is one such example.

3.3 Artificial Evolution

Action-Selection is a very difficult problem to which no general solution is known. A very tempting way to solve any difficult problem is to employ a black box method and indirectly

solve the problem. One such method is to evolve a system that performs in the desired way. In this section we are going to discuss work involving genetic algorithms (GAs) in the context of robot self-sufficiency. For a general discussion, the reader is referred elsewhere. A good introduction is (Holland; 1992).

Parker and Zbeda (2007) used a genetic algorithm (GA) approach in order for robots to learn to use photo-taxis to navigate to a charging station. Since GAs require a large number of generations to discover a suitable solution, the typical approach, as suggested by Harvey et al. (1996), is to evolve the robot controller in simulation and then upload the best controller to the actual robot. This was also the approach used by Parker and Zbeda. Matarić and Cliff (1996) discuss a number of problems related to using GAs for real systems. The prominent ones are, (1) discrepancy between simulation and the real world, which might lead the GA to learn the wrong behaviour and, (2) designing a fitness function is a difficult problem in and of itself. We want to add one more point that makes GAs unattractive for learning self-sufficiency. The most important usefulness of a learning robot is its ability to adapt to the environment. This benefit is lost in the way GAs are generally used because they are evolved in simulation, uploaded to the robot and afterwards not altered any more. Even in a more natural setting in which real robots actually "reproduce" and hence evolve, this learning mechanism is prohibitively slow. Also note that living systems evolve over generations and not during individual lifetimes.

3.4 Foraging in Behavioural Ecology

A very different field of research to look for possible action-selection mechanisms for the Fungus-Eater Problem is Behavioural Ecology. As we argued earlier, animals are concerned with resource management on a daily basis, requiring them to decide between mutual exclusive resource options. Giving an overview, let alone a review of, the wide field of ecology is outside the scope of this document. Instead, we select a subset of topics relevant to the *Fungus Eater*'s action-selection, give a brief overview, and point the reader to relevant references. Most of the material presented in this section comes from (Stephens and Krebs; 1986), (Stephens et al.; 2007) and (Houston and McNamara; 1999).

3.4.1 Optimal Foraging Theory

The underlying assumption in Optimal Foraging Theory (OFT) is simply what the phrase implies, that foraging animals do so optimally. The argument is that if animals from a certain species (with a given set of genes) would not behave optimally, the genes, and hence the species, would have been replaced by an evolutionary process with a species that performs better. In other words, only optimal behaviour is evolutionary stable and as a consequence, animals must behave optimally.

Average-Rate Maximizer

An important model in OFT is that of the Average-Rate Maximizer (ARM). Under this model, an animal is assumed to behave as if to maximize the average intake rate of food. Following Stephens and Krebs (1986), Holling's disc equation (Holling; 1959) gives the gain rate as the net amount of energy gained divided by the sum of search and handling time. A more general, stochastic version is

$$R = \frac{\lambda \bar{e} - s}{1 + \lambda \bar{h}} \tag{3.1}$$

where λ is the encounter rate, $\bar{e}$ is the average energy gained per encounter, s the energy cost of search per unit time and $\bar{h}$ is average handling time for a food item. The objective of an ARM is to maximize this rate R.

How to calculate the average gain rate is still somewhat a controversial in behavioural ecology literature, more precisely, in which form animals seem to use it. The question is whether to calculate the ratio of expectations

$$RoE = \frac{\sum_{i=1}^{n} G_i}{\sum_{i=1}^{n} T_i} \tag{3.2}$$

or the expectation of ratios

$$EoR = \frac{\sum_{i=1}^{n} \frac{G_i}{T_i}}{n} \tag{3.3}$$

The original formulation by Charnov and Orians (1973) is using the ratio of expectations. Templeton and Lawlor on the other hand argue that Charnov is flawed and actually uses the expectation of ratios. Gilliam et al. (1982) strongly dismiss this argument. Bateson and Kacelnik (1996) take up the discussion and takes both averages on European starlings *Starnus vulgaris*. From their experiments they conclude that the behaviour of starlings is

better matched by the expectation of ratios. Bateson discusses a few possible explanations. The most interesting in the context of the *Fungus Eater* is the following hybrid model

$$R = \frac{\sum_{j=1}^{n/f} \left(\frac{\sum_{i=1}^{f} G_i}{\sum_{i=1}^{f} T_i} \right)}{\frac{n}{f}} \tag{3.4}$$

where f is the frame size over which the average is computed. A frame size of $f = n$ corresponds to the rate of expectations, while $f = 1$ resembles the expectation of ratios. Limiting the frame size would allow the forager to adopt faster to changes in the environment. Although we do not limit the frame size in the robot controller presented in Sec. 7.4 because we are interested in the principled idea of the control strategy, it would be an interesting extension.

Zero-One Rule

According to Stephens and Krebs (1986) several researchers independently discovered that, if the forager faces a decision between two or more different food types, ARM leads to the zero-one rule (q.v. Schoener (1971)). This rule states that an ARM will behave optimally, that is, it will maximize the average intake rate by either always consuming a resource of a certain type upon encounter or to always reject it. Policies under which the forager consumes resources of a given type every once in a while are ruled out. Details of why this is the case are given by Stephens and Krebs (1986), but we will give a brief intuition. Let us assume there are only apples and strawberries in the world. For a given set of encounter rates, handling times and caloric content, we gain more energy per unit time by just consuming the apples, because while eating strawberries we lose the opportunity to eat another apple.

To determine which resource types should be included in the diet, one sorts the resource types by their energy rate (e/h) and adds resource types to the diet as long as the average intake rate of the overall diet is larger then the energy rate of the next resource type to be added. In other words if the energy rate of a food item is below the average intake rate of my diet without that food time I should not add to my diet.

Marginal-Value Theorem

With the zero-one rule, we have seen how a forager should make decision about which resource types to include in its diet. Another important type of decision to be made is when to leave a patch. The Marginal-Value Theorem (MVT) is of use here (Charnov; 1976a,b; Charnov et al.; 1976). We give more details on the MVT in Sec. 7.2.

Information Use

We can identify several subproblems of information use, including discrimination, tracking, and sampling. Discrimination is concerned with clustering objects in two or more groups where the assumption is that each group is valued differently by the forager. Stephens et al. (2007) suggest using a receiver operating characteristic (ROC) curve. This curve plots the probability of false accept vs the probability of correct accept, over the whole range of a discrimination threshold. Given the ROC curve, it is possible to determine the optimal discrimination threshold. For details see Stephens et al. (2007).

Tracking a dynamic environment is a challenging task. McNamara (1982) use a simple low pass filter to track and "learn" a certain value, e.g. rate of encounter or travel time, of the environment. This certainly assumes that the frequency of the noise is higher than the frequency of the change we intend to track. This very basic system also fails to utilize information about cycles in the environment, e.g. circadian or annual rhythms.

An important, but in Stephens and Krebs (1986); Stephens et al. (2007) only briefly addressed, question is how to sample the environment. Lima (1984) analyzed a model of a bird having to decide how long to sample a patch for, given that the patch is either completely empty or filled to a certain degree. This model describes the optimal behaviour given the ratios of empty vs. full patches and filling degree of the full patches. From a generative model point of view, this knowledge is not available to the forager *a priori*, in fact it is part of the knowledge the forager has to obtain by sampling. We will show in chapter 7 how sampling can be included in the task-switching algorithm.

We conclude this section about information use with a key remark by Stephens et al. (2007), "Information is valuable when it can tell you something that changes your behaviour". Therefore, information has no inherent value. The value comes by utilizing information in some way.

Short Fall Risk

So far we have primarily reviewed deterministic models. Unfortunately, the world is often appears non-deterministic from the agent point of view and the variance introduced by this fact can mean risk for a forager. Risk-sensitive foraging models handle different forms of risk. In the interest of the *Fungus Eater* we will ignore predation risk and solely concentrate on short fall risk.

An early short fall model is the z-score model introduced by Stephens (1981). As before, we will follow the outline by Stephens and Krebs (1986) and focus on the results and refer the reader to the papers for the derivation. The z-score model calculates the probability that a forager has accumulated enough food storage at a given point in time, e.g. dusk, to survive a certain time interval, e.g. a day, where the assumption is that the amount of food collected is normally distributed and the variance is equated with risk. Stephens derives the extreme-variance rule from the z-score. This rule says that a forager should choose the smallest variance if the mean of the collected food exceeds the requirements, and should choose the largest variance if the mean is below the required amount of food. This intuitively makes sense, as survival is guaranteed, there is no need to take a risk. On the other hand, if the forager is starving, taking a gamble is the only option.

The z-score model is quite limited. It only considers the energy reserves at one point in time, e.g. dusk. Cases in which the forager ran a negative energy budget on it's path to dusk are treated as viable options, which they are clearly not. The Lazy-L model (Stephens; 1982) is an attempt to overcome this limitation by introducing absorbing boundaries to the state space that represent lethal conditions.

More interesting are models that include the forager's preference for earlier rewards. Not only are those closer to observed animal behaviour, (Green et al.; 1981) but they also better reflect economic reality.

3.4.2 Discounting

Discounting is the process of calculating the present value of an item, e.g. a reward, received in the future. In economics, money is usually discounted over time to reflect the fact that money available now has a higher value than the same amount a year from now. One reason is that money available now can be spent on an investment and thus will yield a gain.

It is argued that similar principles apply in the animal kingdom. Food obtained now

is worth more to an individual then food obtained tomorrow, especially if the animal cannot survive until tomorrow without that food. Stephens (2002) identifies four reasons for discounting in animals:

- Collection-risk discounting:
  - Termination risk: reflects the risk of having to terminate a sequential task because of some unknown cause. If V is the value gained per step in the sequence and w is the constant probability to continue for one step in the sequence to the next, the discounted value is given by $V + wV + w^2V + w^3V + ...$
  - Interruption risk: covers the risk of being interrupted while performing a sequential task, but in contrast to the termination risk, the task can be continued after the interrupt. If $1 - w$ is the probability that an encountered gain cannot be obtained, the discounted value becomes $wV + wV + wV + wV + ...$
- Opportunity-cost discounting:
  - Rate cost of delay: having to wait for an item delays future rewards
  - Lost opportunity to use: in waiting for an item, the forager forgoes the opportunity to utilize a given alternate benefit.

Stephens (2002) then argues further that none of the economic discounting reasons can sufficiently explain the observed preferences for immediacy in animals, because this would require implausibly large discounting rates. Instead, Stephens suggests that an additional reason for immediacy can be found in the discrimination problems an animal faces when deciding between two reward options. In operand conditioning, t_i is the duration of a reward and τ the delay before a reward is given. It is argued that animals decide based on the ratio $R = (t_1 + \tau)/(t_2 + \tau)$, where option 1 is selected if $R > 1$ and option 2 otherwise. As τ increases the difference in the reward duration is 'damped out'.

In a later paper, Stephens et al. (2004) show in a series of simulation experiments that impulsiveness (favouring immediate rewards over delayed possibly larger rewards) can achieve higher long-term intake rates than behavioural rules that actually compare long-term rates. The explanation is that impulsive rules have a discrimination advantage over long-term rules because they ignore the common elements and this lets them evaluate a stronger signal of the relevant differences between a pair of alternatives, as argued earlier.

For the *Fungus Eater*, we are free to choose the decision mechanism that is most suitable for our purposes. Since there does seem to be a reason to base a decision on the ratios of two options as in Stephens case, we do not need discounting to help with the discrimination problem. Yet there are other good reasons for us to consider discounting. By discounting the reward a *Fungus Eater* obtains, we embed the robot into the economical reality of its owner. Simply put, we build the owner's need to recoup her investment into the robot controller. Another reason for discounting in artificial intelligence (AI) is that we often deal with infinite reward sums. Deciding which of two infinite sums is larger is not possible. By summing discounted rewards, these sums become finite and thus we can compare them in a meaningful way (Stuart and Peter; 2003). A third reason for discounting is that we deal with a physical system and as such the *Fungus Eater* is bound to fail at some point in time. Discounting will prevent the controller from deciding to try to obtain most of its reward after the expected life time of the robot has passed.

Forms of discounting

If we base our action-selection decisions on discounted rewards, the form of discounting is important, since it will determine which action we prefer under which conditions. In AI exponential discounting is typically used (Stuart and Peter; 2003; Sutton and Barto; 1998). Economics also tends to employ exponential discounting since it directly corresponds to interest rates. In exponential discounting, the reward value V is given by:

$$V = Re^{-kD} = R \cdot \beta^D \tag{3.5}$$

where R is the reward magnitude, D the delay, $k > 0$ the decay rate and $0 < \beta < 1$ the discount factor ($\beta = e^{-k}$).

In experimental settings humans and animals on the other hand consistently show hyperbolic discounting (Schweighofer et al.; 2006). This type of discounting takes the form

$$V = \frac{R}{1 + hD} \tag{3.6}$$

where $h > 0$ is a parameter governing the intensity of the discount. One important difference between the two discount functions is that two agents using either function would show different behaviours. Exponential discounting leads to constant preferences, that is if I prefer option A obtainable after 1 day, over option B achievable after 30 days, I also prefer option A after 101 days from now compared to option B after 130 days. This is not necessarily

true in the hyperbolic case. Here an individual may prefer one apple after 1 day over two apples after 30 days and at the same time prefer two apples 130 days over one apple after 101 days. This is commonly known as preference reversal (Kacelnik; 1997). It is therefore often argued that exponential discounting leads to rational behaviour, because the agent always makes the same choice under the same conditions, one reason it is preferred in AI (Schweighofer et al.; 2006).

How can we explain hyperbolic discounting and preference reversal in humans and animals? Kacelnik (1997) points out that hyperbolic discounting is equivalent to rate maximization (see Sec. 3.4.1). And as Stephens and Krebs (1986) argue, rate maximization leads to maximizing food intake. Green and Myerson (1996), referring to data from Raineri and Rachlin (1993) who show that the discounting rate is dependent on the amount of the reward, investigate if the amount dependent discount rate could explain preference reversal. Green concludes that varying discount rates can led to preference reversal. Meaning animals could actually perform exponential discounting with varying discount factors. But as Green points out, the variance in the data is more consistently explained by hyperbolic discounting.

Yet another explanation could be that animals are 'optimized' to make repeated choice decisions. In natural settings, animals rarely have to make a single choice decision as they are modelled by Eq. 3.5. For example decisions like whether to consume the nearby but small apple or the distant and large one followed by no further decision. Instead they typically face decisions like small-and-close-by or large-and-far-apple and what to consume after that? Again a small immediate option or the delayed but larger one? A exponentially discounted reward under repeated choice is given by

$$V = \sum_{i=1}^{n} \beta^{iD} R \tag{3.7}$$

where, as before, D is the delay, R the non-discounted per choice reward, β the discount rate and n is the number of repeated choices (Stephens (2002) investigates a similar model). In Fig. 3.1 we plotted the accumulated reward from two reward options under repeated choice. By choosing the first option (orange) the agents obtains a reward of 5 units every 4 units of time. The second option (green) yields a reward of 10 units every $4 + \Delta T$ units of time. By altering ΔT, we increase the delay between rewards for the second option, and by changing the number of repeats n, we can clearly see that the preference changes. This is not based on any animal data. We simply wanted to point out that preference reversal can

be observed under exponential discounting given a repeated choice scenario. So what may appear to be hyperbolic discounting, could in fact be exponential discounting while using the current reward to make predictions about future rewards. Alternatively we could adopt Eq. 3.7 to take the experienced global average to predict future rewards. This explanation is interesting because it closely related to the Marginal-Value Theorem (Sec. 7.2) where foragers make patch leaving decisions based on estimating future rewards.

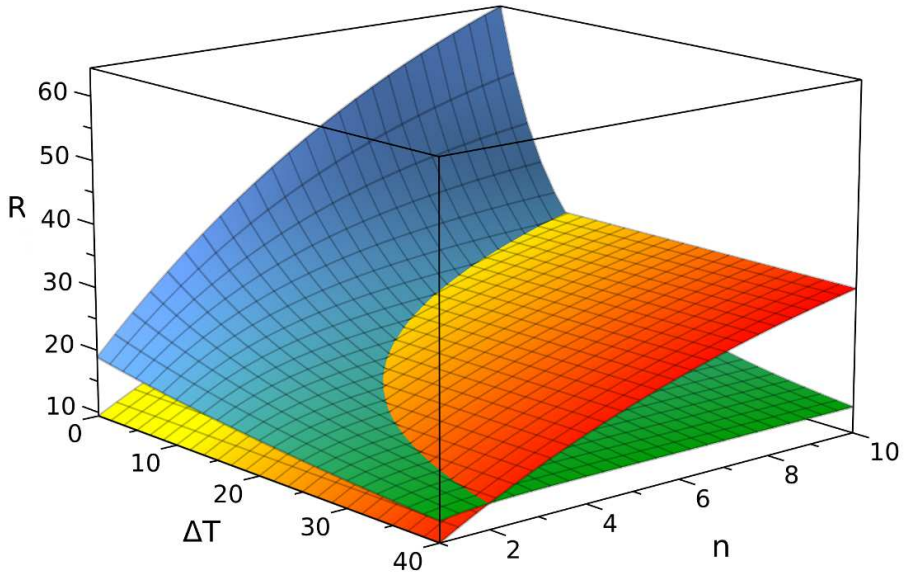


Figure 3.1: Preference reversal in exponential discounting for repeated choice. ΔT denotes the difference in delay of the two reward options, n represents the number of repeated choices and R is the accumulated discounted reward.

3.4.3 Matching Law

The matching law of operand conditioning states that the relative response rate of an animal matches the relative rate of obtained reinforcement. It was first introduced by Herrnstein (1961) (q.v. Herrnstein (1997)). Herrnstein's strict matching law was refined by Baum (1974) to the generalized matching law,

$$\log\left(\frac{R_1}{R_2}\right) = b + s\log\left(\frac{f_1}{f_2}\right) \tag{3.8}$$

Where R_i is the response rate, f_i is the rate of obtained reinforcement, b is the bias and s is the sensitivity.

We mention the matching law here because although it is descriptive, it is one of the few models that can easily be turned into a behaviour generating model. Seth (2007) points out a few interesting points. (1) There is an essential difference between the matching law and probability matching. In the matching law, the foragers response matches the ratio of *obtained* rewards, where in probability matching, the response matches the ratio of *available* rewards. Note the latter is much more difficult to implement since gathering knowledge about available rewards is more difficult than measuring obtained rewards. (2) Although in a trivial way, the zero-one rule (Sec. 3.4.1) is covered by the matching law, but not so by probability matching. (3) Neither the matching law nor probability matching guarantees optimality.

From Seth (2002) we can see that matching can yield higher overall rewards than exclusive choice depending on the reinforcement schedule. Under variable interval schedules, a certain delay has to elapse after a reward on a given option until that option can be rewarded again. In this case exclusive choice is no longer optimal and matching can be observed in animals and humans.

3.4.4 Ideal Free Distribution

An important optimality criterion in multi-agent foraging in patchy environments is the ideal free distribution (IFD), described by Fretwell (1972). The IFD is an equilibrium distribution of foragers such that no forager can relocate in order to increase its profit. It is a non-trivial problem to design a robot controller that leads to an IFD. Since this document is mainly concerned with the solitary Fungus Eater Problem, the IFD is of lesser interest. We mention it here since it provides a nice optimality criterion for the case where all foragers have to be employed. Situations in which foragers are allowed to remain idle are not covered by the IFD. In Sec. 3.5.4 we review the literature on control strategies that aim at achieving an IFD of foraging agents.

3.4.5 Dynamic Modelling

A lot of work in ecology relies on dynamic models of animal behaviour. An important and widely used technique in dynamic modelling of behaviour is dynamic programming (DP)

(Mangel and Clark; 1988). Although a powerful optimization technique, its application to action-selection for robotics is limited. DP analyzes the problem from the final time step backwards to the current moment in time. This requires the agent to have perfect knowledge of its own and the world's state during that time period. That also necessitates defining a planning horizon, something that is often difficult since, for example, the life time of the agent is unknown *a priori*. One option is to choose a smaller planning horizon than the expected life time of the agent, but then DP requires some target state the agent should be in at the end of the planning period. In general this is difficult. It is certainly possible to select trivial conditions which usually do not lead to optimal behaviour, e.g. full energy reserves at the end of each day. A large planning horizon requires even better models. Thus from an engineering point of view, it seems very unlikely that any biological system actually employs DP on a large scale to make decision about what to do next.

3.5 From Ethology to Robotics

In the previous section we reviewed popular, theoretical models of animal behaviour. Now we look at research concerned with taking these models, ideas of these models, or just plain observations of animal behaviour and turning them into robot controllers.

3.5.1 Robot Foraging

A canonical task in robotics is transporting pucks, especially in a foraging setting (Cao et al.; 1997). To a large extent this might be due to the fact that this transportation task is at the limit of the capability of today's research robots in terms of sensing and object manipulation. In any case, it makes for a challenging robot problem, especially in the multi-robot domain. The parallels to foraging in ecology are striking and a number of roboticists have been inspired by animal research. For example, inspired by social insects, Holland and Melhuish (1999) use very simple rules to achieve puck clustering with multiple robots. The rules are of the kind, "if I carry a puck and I encounter another puck, I drop my puck next to it". Especially elegant are those solutions that merely are based on mechanical properties, e.g. consider robots that are assigned the task to construct clusters of boxes or pucks. If these robots are designed to only be able to push one box at a time, clusters will emerge over time by the robots simply pushing the pucks together.

It has long been known that multi-robot system performance initially increases with

increasing number of robots, but due to interference, the system's performance is subject to diminishing returns (Vaughan et al.; 2000a). Therefore, the performance increase slows down and once a critical mass of robots is reached a drop of performance can be observed. Once the environment is saturated, the system might not perform at all anymore. This effect has been formally modelled by Lerman and Galstyan (2002). Since it is undesirable the focus of a lot of work has been on ways to reduce the interference amongst robots.

Vaughan et al. (2000b) and Sadat and Vaughan (2010) separate robots by having approaching and departing robots use different routes. By placing virtual bread crumbs on a shared map the robots automatically find routes of low interference. Another way to reduce interference is "bucket-brigading". Østergaard et al. (2001) introduced this method and inspired a series of follow-up papers. The basic idea is to spatially separate robots by having them travel only short ways and pass pucks to each other rather then attempting to drive to the sink location at the same time. The challenge lies in how to get the robots to cooperate by puck passing without using a centralized controller. As usual the argument against a central controller is that it would not scale. Østergaard et al. (2001) use a simple set of rules similar to Holland and Melhuish (1999) and achieve an increase in performance.

Shell and Matarić (2006) extend the bucket-brigading idea by assigning each robot a fixed-size work area. Within this area a robot moves pucks that it encounters in the direction of the sink. By using the natural uncertainty in the position estimate, the work areas slowly drift over time and thus full coverage is achieved. A natural extension of this work area idea is to have the size be adaptive. Lein and Vaughan (2008) use the exact same system as Shell, with one addition. The size of the work area slowly grows over time unless robots "bump" into each other, which causes the size of the work area to decrease. Lein reported a substantial performance increase compared to the fixed size work area approach.

To better handle situations in which the pucks are clustered, Lein and Vaughan (2009) extend the system so it does not rely only on noise in the position estimates for relocating the work area, but actively seeks for better locations. This is done by moving the center of the circular work area closer to where the robot encountered the last puck.

Brown et al. (2005); Vaughan et al. (2000a); Zuluaga and Vaughan (2005) investigate robots using aggressive displaying behaviour to lower to negative effect interference has on performance. In situations in which two robots are in conflict about a spacial resource, e.g. a door way that can only be passed by one robot at a time, the robots use aggressive displaying behaviour to determine which robot have the right-of-way. By choosing the level

of aggressiveness proportional to the need to go first, the robots can achieve the correct ordering for accessing the spacial resource.

Reducing interference is one issue in multi-agent foraging, the other is minimizing energy usage. This falls in the domain of worker allocation. Here the question is how many workers are required to get the job done without recruiting idle workers who would just waste energy. Liu et al. (2007) present an adaptive controller to dynamically decide between assigning robots a foraging task or to have them rest, with the goal to maximize net energy income. In chapter 8 we show how we can use interference to address this problem.

3.5.2 D.R.K / Cue × Deficit Model

Closer to the *Fungus Eater* than multi-robot foraging are single robot systems concerned with resource gathering. One model, the d.r.k model introduced by Spier and McFarland (1996) and based on earlier work by Sibly and McFarland (1976), is one of the few ethology inspired action-selection mechanisms. The basis of this work is to model the animal as a set of internal state variables which represent its physiological state (McFarland and Houston; 1981). Such state variables could be hunger, thirst, temperature, blood oxygen level, etc. These variables form an Euclidean vector space, where each variable is represented by one orthogonal axis. The agent moves within this space, depending on the actions the agent performs. Since the physiological variables are constrained, e.g. hormonal levels cannot be negative, running out of energy causes death, or there is an upper and lower limit on body temperature in order for the bio-chemical processes working properly, this space is surrounded by a lethal hull. The objective of the ASM is it to keep all state variables with in the lethal hull at all times. We argued earlier (2.2) that this definition needs to be extended to be useful for the *Fungus Eater*.

Sibly and McFarland (1976), introduce a utility function to evaluate the goodness of a certain action with respect to the current state. They argue for a convex function to assign higher costs to variables whose state is more critical at a given time. Purely for mathematical reasons a quadratic cost function is chosen, which, for a two state variable agent, results in a simple decision rule:

$$\begin{aligned} if \quad & d_a r_a k_a > d_b r_b k_b \quad : \quad consumeA \\ if \quad & d_a r_a k_a < d_b r_b k_b \quad : \quad consumeB \end{aligned} \tag{3.9}$$

Where d_i denotes the deficit of state variable i. r_i is the availability of a resource that lowers the deficit of state variable i and k_i is the accessibility, in other words, a limit on the rate of a consuming a resource.

Spier and McFarland (1996) extend this model by saying not only has each resource an accessibility, but this accessibility also depends on the tool the agent uses. For example a bowl of soup has a higher accessibility if the agent consumes it with a spoon instead of chopsticks. Later this is further relaxed by saying each action has a certain accessibility with respect to a resource. Where Sibly and McFarland (1976) is a theoretical analysis, Spier and McFarland (1996) put it to the test on a simulated agent employing the decision rule given in Eq. 3.9. They show that the *drk* model outperforms two alternative strategies, one in which no tools are used and one in which a tool is always selected.

Going back to the state space, McFarland and Spier (1997) introduced the notion of basic cycles and trajectories an agent performs in the state space. E.g. a robot's basic cycle may consist of working, thus earning money and spending energy, searching for energy, thus spending energy but keeping its bank account steady, and recharging, by which it gains energy in exchange for money. McFarland identifies cycles of different shape depending on whether the robot has to pay for its energy or not. These cycles do not have to be static. They might move within the state space, where cycles that move towards the lethal hull are not sustainable. These cycles do not have to be closed, that is, after one cycle the agent might not be at the same location in the state space as where it started the cycle. E.g. if the agent makes a profit during a cycle, it starts the next cycle at a higher monetary state.

Fusing availability and accessibility to what is called a "cue", McFarland and Spier (1997) program real robots with a Cue×Deficit decision rule. Very similar to Eq. 3.9, the robot chooses to work if the work cue times the work deficit is higher than the energy cue times the energy deficit and visa versa, where the strength of a cue is calculated from the current sensor reading. This causes a major problem if the robot does not sense any charging station. Then the energy cue is zero and therefore, refuelling is never chosen as an appropriated action, because no matter how high the energy deficit, multiplied with a zero cue, results in zero activation. Spier overcomes this problem by introducing what is called an ambient cue which reflects the robot's trust in the world that a resource is available somewhere. The ambient cue is basically a means to ensure each cue is always larger than zero. Details about the robot used can be found in (Steels; 1994).

Spier and McFarland (1997) extend their previous analysis of the Cue×Deficit model

by comparing it in simulation to a greater number of alternative decision strategies. These alternatives are

S1 consume what ever resource is closest to the robot

S2 consume the resources whose deficit is the highest, if this resources in not in the field of view, search for it

S3 consume the resource, that has the lowest cost to consume. Here the cost is calculated as $cost_A = (deficit_A - payoff_A + lossPerCycle_A \times distance_A)^2 + (deficit_B + lossFromConsumingA + lossPerCycle_B \times distance_A)^2$

S4 follow the previous strategy, but if the required resource is not within the field of view, consume the closest resource.

All strategies were tested on a large number of environmental settings, varying the sensor range and the resource distribution. Spier concludes that "The Cue×Deficit strategy performs very well in all the environments, only being bettered by" S4 "in the 20 resource unlumped environment with a low sensory range." Looking at the provided graphs, we conclude, that S2 and S3 perform generally very purely. S4 performs similarly well as the Cue×Deficit strategy and usually outperform S1, which performs remarkably well for a very simplistic control strategy.

In yet another extension, Spier (2001); Spier and McFarland (1998), added learning to the *d.r.k. model*. The agent learns both the availability r and the accessibility k. In the tradition of operand conditioning, the simulated agent was placed in a Skinner box in which it could operate two different manipulandu in order to obtain one of two resources, each of which would satisfy the deficit in the agent's state variables to a certain degree. Delta learning is used to learn r as well as k. Eligibility traces are used to help with the credit assignment problem caused by the delay between operating a manipulandum and receiving a reward/resource. Spier argues that this model can account for phenomena like outcome devaluation (e.g. food aversion). Where he clearly shows that the agent learns accessibility, learning of availability is not shown. Also reducing the r matrix element of the chosen reinforcer to zero instead of actually "poisoning" the agent is somewhat unsatisfactory from an embodied point of view.

3.5.3 Robot Ecosystem

The robot ecosystem was designed by Steels (1994) in collaboration with McFarland (1994b) at Brussels University. The ecosystem is a rather ordinary, rectangular robot arena in which one or more robots live. To supply the robots with energy, charging stations marked with a blue light are placed in the arena. A third component are boxes with lamps mounted on top. A key idea in the ecosystem is that the energy flow into the system is constant, hence the lamp boxes act as competitors to the robots for energy. Obviously there is also competition between the robots themselves. In order for the boxes to not only lower the globally available amount of energy, the robots can actively decrease the energy demand of the boxes for a certain amount of time by bumping into them. Therefore by working (bumping into the boxes), the robots can actively increase the amount of energy available to them.

Steels (1994) robots master 6 different behaviours: *forward movement*, *touch-based obstacle avoidance* which causes the robot to retract and turn away from obstacles, *smooth obstacle avoidance* path deviation in proximity to obstacles, *blue photo-taxis* attracts the robot to the charging station, *halt while recharging*, and *yellow photo-taxis* attracts the robot to competitive boxes. All 6 behaviours are "programmed" by sets of simple rules by basically adding weighted sensor values to the left and right wheel speed commands. For example, the rule responsible for *forward movement* modifies the left wheel command like this $v_l = \frac{(150-v_l)}{10}$ and the rule for *halt while recharging* at the same time modifies the left wheel speed to $v_l = v_l - 150E_{available}$. Steels is somewhat imprecise what $E_{available}$ actually is, but the idea can be clearly understood: decrease the wheel speed if charging. Therefore this example illustrates the idea of a robot controller based on parallel running processes setting commands independent of each other and achieving complex behaviour, namely driving forward, stopping while recharging, and leaving the charging station either once the batteries are sufficiently charged or once the level of provided energy drops due to competitors (the lamp boxes). This is a classic example of a fusion-based ASM in contrast to priority-based ASMs, such as subsumption architecture or winner take all systems.

In an extension, Steels (1996, 1997) attempts behaviour acquisition in a simulated version of the ecosystem. In contrast to the previous version, the robot is initially given a "set of primitive innate basic behaviours" but has no knowledge as to which behaviour to execute in which situation. Instead the robot is required to learn behaviour arbitration. The core

of Steels approach is two couplings, each linking two quantities. The *similarity-coupling* modifies the target quantity to match the source quantity with a given rate. And the *reverse-coupling* modifies the target quantity to match the inverse value of the source quantity, again with a given rate. By making couplings depended on two conditions, namely an increase or a decrease in the source quantity, Steels derives four benes (to suggest an analogy with genes)

- A follows increase in B
- A follows decrease in B
- A reversely follows increase in B
- A reversely follows decrease in B

In order to learn the correct behaviour for a given situation, the robot randomly selects a sensor as well as a motivational quantity and then evaluates each of the four benes for the selected sensor and motivational quantity. Steels (1997) implements two sensors, one for the battery level and a binary one for contact with the charging station and three motivational quantities each expressing the activation of a primitive behaviour, *move-forward*, *align with charging station*, *align with competitor*. Instead of giving the robot a complex cost function which encodes the expected behaviour, Steels simply has the robot evaluate the viability of couplings. This is done by adding a bene to be tested to the set of established benes (this set is initially empty), then executing the benes for a short period of time, during which the minimum and maximum energy level are observed. If the centre between minimum and maximum is higher then previously observed, the bene is said to be beneficial, since it increased the energy level and thus it is added to the set of established benes. Otherwise, it is not beneficial and thus dropped. To augment learning, the robot is initially placed in a benign environment, that is a small environment with a high likelihood of connecting with the charging station by chance and a small number of slowly regrowing competitors. Over time the environment becomes more challenging. Steels (1997) argues this can be observed in "newborn and young animals who are often first exposed to less challenging environments to give them a chance to progressively acquire new behaviours needed later in life".

A critique on the approach comes from Birk (1996, 1998). He argues that the min/max-level is a long-term evaluation method suited to improve overall efficiency of behaviours and thus is too slow to solve dangerous situations such as being close to the lethal hull of

the agent's state space. In addition, he argues that what is learned depends largely on the duration of the trial period. Therefore Birk extends Steels (1997) work by adding a second, faster feedback mechanism. This one is based on the change of energy consumption. An increase of energy consumption is interpreted as a bad sign, where a decrease is viewed as beneficial. For example bumping into an obstacle increases the energy consumption since the motors push against the obstacle, where establishing contact with the charging station lowers the current drawn from the battery because energy is also drawn from the second energy source. Unfortunately, no performance results are given by Birk (1998) and it remains unknown if the proposed ASM learning system was actually tested in simulation or on real robots.

3.5.4 ϵ-sampling and ω-sampling

An important question uninformed foragers have to answer is, "what is the quality of my current resource source and can I do better by foraging at a different patch"?

Thuijsman et al. (1995) suggest ϵ-sampling. Given two patches with unknown resource gain rates, an ϵ-Sampler initially selects a patch at random. With probability $1 - \epsilon$ the forager stays in this patch and with probability ϵ the forager switches to the other patch. If the instantaneous resource gain rate of the new patch is higher then a critical level, the forager remains in the new patch. Otherwise, the forager returns to the previous patch. As for the critical level, Thuijsman suggest using the low pass filtered instantaneous gain rate. In ecology, this low pass filtering is known as the linear operator function and was originally introduced by Bush and Mosteller (1955).

ϵ-sampling was introduced to explain the seemingly irrational behaviour of matching (q.v. 3.4.3). Thuijsman's argument is that in a multi-animal environment, ϵ-sampling explains matching and that it is optimal because it leads to an IFD (Sec. 3.4.4).

Seth (2000, 2002) picks up on the idea of ϵ-sampling and argues that the claims of Thuijsman et al. (1995) are not substantial, explicitly that ϵ-sampling neither explains matching nor does it lead to an IFD. Instead Seth argues that ω-sampling does both.

ω-sampling is very similar to ϵ-sampling, where the primary difference is that the forager maintains not only one global estimate of the gain rate but one estimate for each patch. Again these estimates are obtained by low pass filtering the instantaneous gain rates. As in ϵ-sampling, a forager employing ω-sampling, initially selects a patch at random, stays in this patch with probability $1 - \epsilon$ and leaves it with probability ϵ. In addition an ω-sampler

continuously compares the estimated gain rates for all patches and switches to the most profitable one.

We do not doubt that ω-sampling performs better then ϵ-sampling, nevertheless the results (Seth; 2002) have to be viewed with a careful eye. A major limitation is that there is no patch switching cost, hence exploration comes only with the potential cost of lost opportunity for foraging in a suboptimal patch but no cost for travel, etc. has to be paid. Clearly a fixed exploration cost will have some influence on the payoff of exploration.

Interesting, and at the same time raising questions, is Seth's distinction between two types of resource allocation. In continuous allocation (CA) the forager obtains a certain amount of resources at every time-step. This is in contrast to probabilistic allocation (PA), where at every time-step the forager obtains a certain amount of resources with probability q and with probability $1 - q$ nothing is obtained. In robotics a charging station or a solar panel would be examples for CA, where searching for and picking up battery packs would constitute PA. The problem is that in PA the instantaneous gain rate is 0 with probability $1 - q$, but at least in ϵ-sampling the forager switches patches if the instantaneous rate falls below the critical level. Instead of using PA directly, Seth (2002) seems to use the expected gain rate, which seems to be equivalent to CA. We address the problem of measuring the instantaneous rate of atomic units, probabilistic allocation, in chapter 7.

Both foraging methods require two parameters, (1) the probability ϵ of exploring the other patch and (2) a filter constant for the low pass filter. Both are difficult to determine. Seth resolves this elegantly by using a GA to set the parameters, but this also shows the limitations. A set of parameters is only optimal for a certain environment, for example, changing the gain rates will require different parameters.

Chapter 4

Basic Behavioural Cycles

In the previous chapter we defined self-sufficiency of the *Fungus Eater* in terms of a state-space. All variables that govern the functionality of the robot sit on orthogonal axis of this space and the objective of the controller is to keep all values within a lethal lower and upper bounds. This state-space idea goes back to McFarland and Houston (1981). It defines the objective of the controller well and is an intuitive way of characterizing the robot. The main question we have to answer is how do we control the actions of the *Fungus Eater*, using this state-space approach? McFarland and Houston (1981) suggested using control theory. While it is common in robotics to employ, for example, PID controllers as the lowest level motor controllers or even for controlling the attitude of unmanned aerial vehicels (UAVs), applying control theory on a behavioural level is not common. Work like that by Ijspeert (2008) is pushing the boundary, by using central pattern generators to produce walk and swimming behaviours. However the work is still on a motor level.

Often task-switching is based on selecting the task that leaves the robot in a better state. This might seem to make the state-space approach attractive. We might simply be able to hill-climb within this space. Unfortunately the state-space only characterizes the vital space of the *Fungus Eater*, it does not provide a cost function. For example, if I am half a litre short on water and at the same time at my optimal body temperature, am I in a better state if I obtain half litre of water and by doing so raise my temperature by one degree ?

McFarland and Spier (1997) use the state-space idea and propose a control heuristic. We outlined this heuristic in detail in Sec. 3.5.2. Interesting now is the notion of "basic utility cycles". Similar to our definition of the FEP, McFarland's robot uses energy in order

to maximize the owner's utility. In the simplest possible case the state-space of McFarland's robot consists of two variables, energy and money, where money is earned for the robot's owner. Within this space McFarland identifies a basic cycle of work - find fuel - refuel. For the robot to be self-sufficient this cycle has to stay within certain limits of the state-space. We remark that often only the energy variable has to stay within tight bounds, money may grow unbounded. The shape of this cycle is determined by factors such as if the robot has to pay (with money previously earned) for energy or if money is constantly drained from the robot etc.. As with the state-space, the basic utility cycles are a description and a rationalization of behaviour but it is not immediately clear how they help to generate behaviour. It also appears the cycle is imbalanced. While there is a clear transitioning phase from work to refuelling, McFarland and Spier (1997) do not mention a transitioning step from refuelling to work. Perhaps this step is included in the refuelling or working part respectively, but that seems inaccurate since in general during transition neither fuel nor money can be obtained.

4.1 Work-Fuel Model

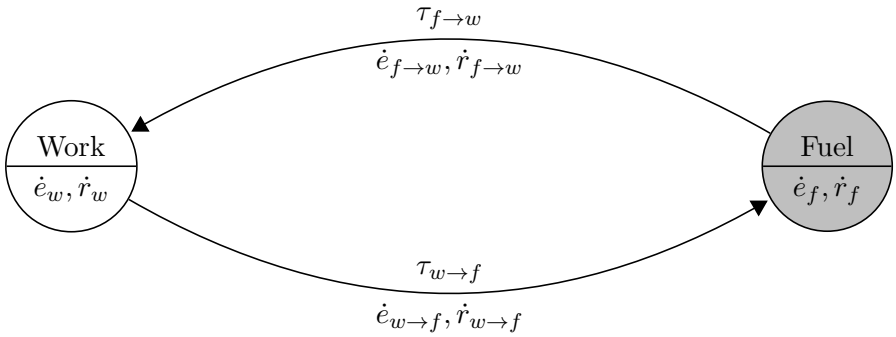


Figure 4.1: WF-Model: The model consists for 4 states, work, fuelling, transitioning from working to fuelling and vice versa.

In this thesis we introduce the concept of basic behavioural cycles. Although not generating behaviour directly and similar to McFarland and Spier (1997) we use the behavioural cycles as a framework for our work. It allows us to define the task-switching problem clearly and characterizes the difficulty of a particular problem. As we will see this concept is also

helpful because it allows us to easily distinguish between two very different types of task-switching a *Fungus Eater* has to perform: (1) switching between tasks of different types, e.g. between working and fuelling, and (2) switching within the task domain, e.g. changing charging stations should they deplete.

A key aspect of basic behavioural cycles is that the behaviour of a self-sufficient robot is usually cyclic. We say usually, because this might not be the case for trivially self-sufficient agents, e.g. machines plugged into the power grid. All other agents have to periodically regain energy. If the environment is constant or we base the agents decisions on averages, the agent always returns to highly similar places in the state-space with every recharging cycle. Since rational agents are required to make the same decisions under the same conditions (Stuart and Peter; 2003), we only have to determine the behaviour for one cycle, the behaviour for future cycles must be the same. Alternatively we can define cycles over work, e.g. transporting from a source to a sink, see Sec. 4.2.8 for an example.

The simplest incarnation of a behavioural cycle is the Work-Fuel model (WF-model). This model is shown in Fig. 4.1. Within the WF-model the *Fungus Eater* can be in one of four mutually exclusive states: working, fuelling, or transitioning between the former two states. Each of the four states is defined by the rates the robot obtains or expends energy $\dot{e}_x$ and increases or decreases its reward $\dot{r}_x$. While the energy rates are straightforward, the reward rates need more explaining. Earlier we stated that the *Fungus Eater* can only obtain rewards while working, therefore the reward rate while transitioning or fuelling should be zero. But this would not capture the case where the robot has to pay for the energy it uses. We might also need a non-zero reward rate while fuelling or transitioning in cases where we want to model interest on the obtained reward.

Another feature of the WF-model requiring more explanation is that it allows the transition from working to fuelling to be different from the transition in the opposite direction. This can be useful to model situations in which the working behaviour is performed over a spatially extended area, e.g. in a transportation task where the robot always transitions from the transportation sink to fuelling and from there moves directly to the transportation source. The time required to perform each transition is given by τ_x. Note that τ_x defines the task-switching time. Given the energy expenditure and reward rates we therefore know the switching cost both in terms of true cost and lost opportunity.

4.2 Example WF-Models

We introduced the general idea behind the WF-model, next we show a few examples and point to solutions if they are known.

4.2.1 Co-located WF-Model

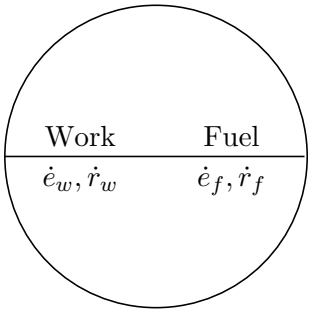


Figure 4.2: WF-Model without switching cost, work and fuel are co-located

Arguably the most trivial WF-model is shown in Fig: 4.2. Work and energy are co-located, therefore the robot does not have to pay a switching cost. This models for example stationary assembly robots that are permanently plugged into the power grid. The robot might still have to pay for the energy it uses and thus has to work to be self-sufficient. Other types of robots covered by this model are solar powered robots in permanent light conditions or mobile robots on a power floor. These floors were first described by Shannon (1993) and were for example used by Watson et al. (2002) for embodied evolution research. For this thesis this model is of little interest since there is no task-switching.

4.2.2 Chain of Point Style Work Sites

A common case in robotics is modelled by a chain of point style work sites and a central fuelling station. This model is shown in Fig. 4.3. In this model the robot has to visit n work sites, where n is potentially very large. Up on arriving at a work site the robot receives one unit of reward and is then free to either move on to the next work site or to approach to the fuelling site. This model assumes that the actual work period is small compared to the travel time and thus can be ignored. We further assume that the order in which the sites have to be visited is given. This models for example office mail delivery or pick up, guarding property, or monitoring the environment.

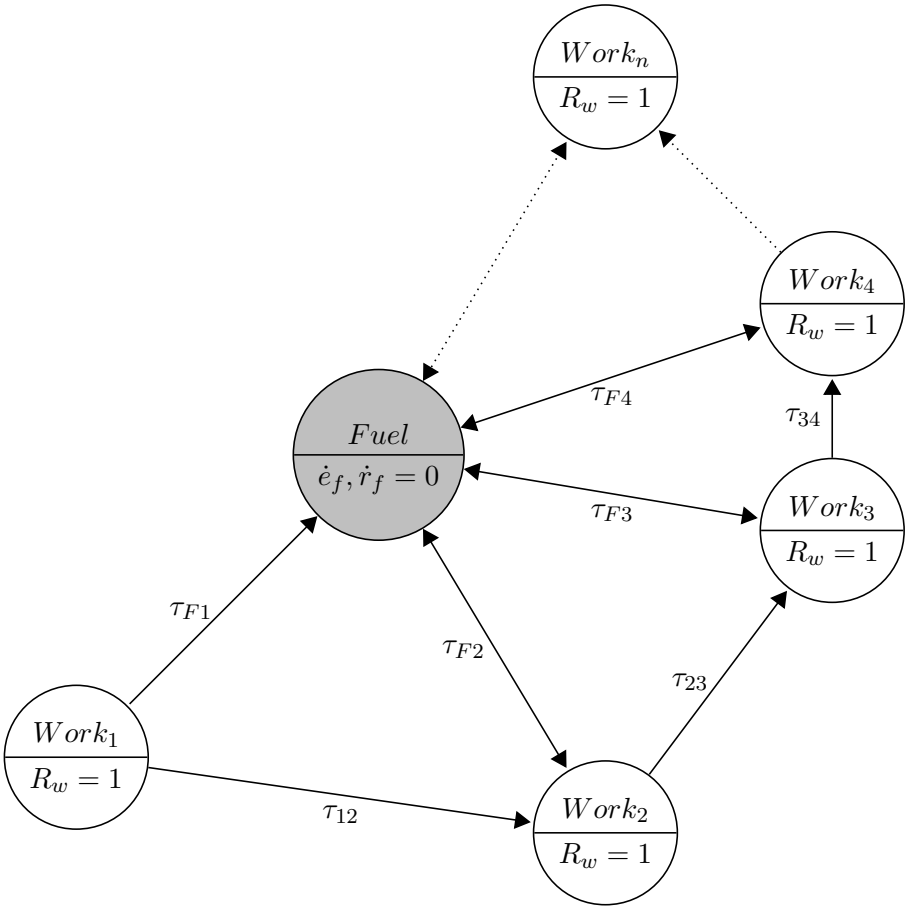


Figure 4.3: WF-Model: chain of point style work sites with a central fuelling station

.

For a solution to exist to this problem it is necessary that the onboard energy storage is large enough for the robot to travel at least from the fuelling site to a work site back to the fuelling site. We can relax this constraint by providing the robot with a second means to obtain energy, that can be used at any point, e.g. solar cells. This guarantees that a solution exists, no matter how far the work site are from the fuelling station. But it also increases the complexity of the decision process. Three heuristic solutions to this problem are compared by Wawerla and Vaughan (2007) and in Chapter 5. Note that this solution also solves the initial case, provided a solution exists.

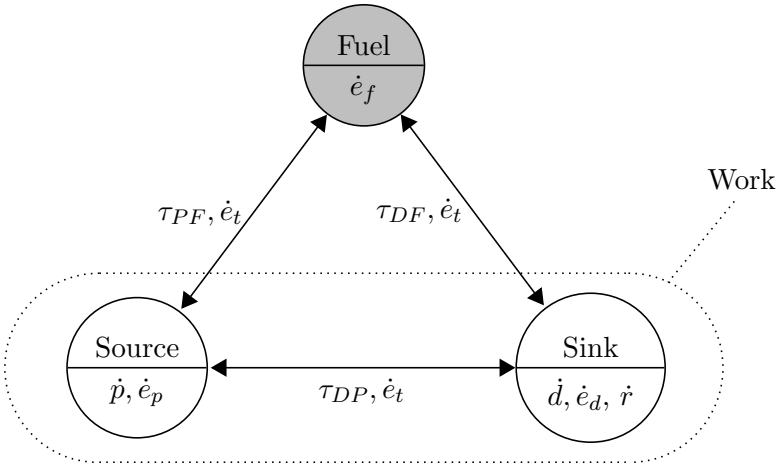


Figure 4.4: Model of a simple transportation task, the robot has to transport goods from a source to a sink, while maintaining self-sufficiency

4.2.3 Discounted Labour

Scenarios in which reward is discounted and the robot has one work and one fuelling site can be modelled using the model shown in Fig. 4.1 and letting $\dot{r}_w = \beta^t$, where $0 < \beta < 1$ is the discount factor. We presented this model as well as a control policy in Wawerla and Vaughan (2008). A detailed outline is given in Chapter 6.

4.2.4 Tranportation Task

A common motivation for robotics research is to transport goods from a source to a sink. In behavioural ecology this is known as central place foraging (Stephens et al.; 2007) (often this means transporting from multiple sources to one sink, e.g. a nest). We can frame this problem as shown in Fig. 4.4. The robot's objective is to transport goods from the source to the sink. At the source the robot loads with a rate $\dot{p}$. Unloading at the sink happens with a rate $\dot{d}$, and during unloading the robot is reward at a rate of $\dot{r}$. Travelling between the various locations takes τ_x time, during which the robot expends energy at the rate of $\dot{e}_x$. At the fuelling site the robot obtains energy at a net rate of $\dot{e}_f$.

Considering the geometry of this problem we can see that the robot should take n trips from source to sink and then has two choices after delivering the n-th load: (1) go back to the source, load one more time, then refuel and deliver afterwards, or (2) refuel first and then pick up another load and deliver it. The total length of both option is the same, but

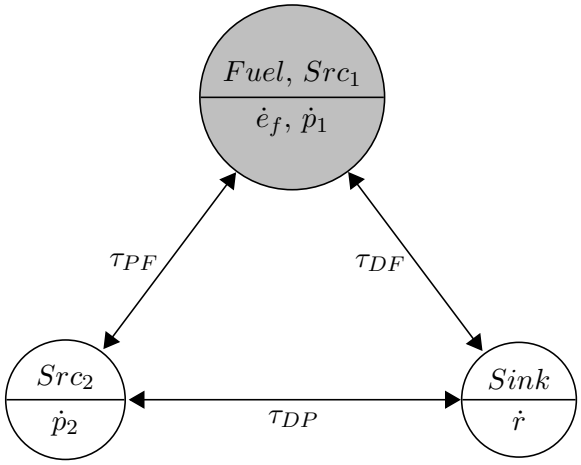


Figure 4.5: Provisioning-Model

the remaining energy might make early refuelling necessary. If not, the robot is better off fuelling later, and thereby increasing the amount of fuel it can load while the cost stays constant.

Applying the arguments from Chapter 6, the robot should arrive at the fuelling site with as little energy remaining as possible. For this non-discounted case, the robot should then fully refuel. Using the principles from Chapter 6 it is possible to work out the fuelling limits for a discounted reward case. For the case considered here we can now calculate the reward obtained per work-fuel cycle.

Let E be the capacity of the onboard energy storage and L be the loading capacity of the robot, then number of source-sink trips per work-fuel cycle is given by

$$n = \left\lfloor \frac{E - (\tau_{DF} + \tau_{DP} + \tau_{PF})\dot{e}_t - \frac{L}{\dot{p}}\dot{e}_p - \frac{L}{\dot{d}}\dot{e}_d}{2\tau_{DP} \cdot \dot{e}_t + \frac{L}{\dot{p}}\dot{e}_p + \frac{L}{\dot{d}}\dot{e}_d} \right\rfloor \tag{4.1}$$

The reward per cycle is then

$$R = (n+1)\dot{r}\frac{L}{\dot{d}} \tag{4.2}$$

4.2.5 Provisioning

Ydenberg and Davies (2010) investigate a problem similar to our transportation model (Sec. 4.2.4). They determine the theoretically best provisioning routines for seabirds. In their analysis the birds have to feed themselves and provide food for their offspring. The latter corresponds to work and self-feeding trivially maps to fuelling in the *Fungus Eater*.

In Ydenberg and Davies' paper, the birds are given two choices, feed and collect food for delivery at the same location or feed at location A and then move to a different location B to collect food to be delivered to their offspring. Certainly the total travel distance of the second option is larger than that of the first option. The reason a bird might choose the second option is that food ideal for self-feeding may differ from food best suitable for delivery. By assuming that self-feeding only has to recover the energy spent for provisioning, travelling and self-feeding, (energy requirements for other activities are ignored) one can calculate the rate of food delivery for each option. Rational birds should then choose the option that yields a higher delivery rate.

Fig. 4.5 shows the provisioning model expressed in our nomenclature. Fuel and source src_1 for delivery are co-located and a second source src_2 for delivery exists with a different loading rate. For the second option to be interesting at all, it needs to provide a higher loading rate ($\dot{p}_2 > \dot{p}_1$). Ydenberg and Davies (2010) conclude that the two patch option yields higher delivery rates if

$$\frac{L}{\dot{p}_1} - \frac{L}{\dot{p}_2} > \tau_{PF} + \tau_{DP} - \tau_{DF} \tag{4.3}$$

where L denotes the carrying capacity for the bird.

In the second part of the analysis, Ydenberg and Davies also consider cases in which the birds self-feed on every n-th provisioning trip. This case is interesting from a *Fungus Eater* point of view, because is resembles situations in which the robot makes n transportation trips per fuel cycle (see Sec. 4.2.4). A solution similar to the previous one is derived.

Ydenberg and Davies (2010) describe how an animal should behave under different circumstances. For us to actually implement this as a robot controller we would also have to consider the onboard energy storage.

4.2.6 Diminishing Returns

An interesting scenario is when the gain rate from a task is not constant, for example if a task becomes more and more difficult to perform as it reaches completion. Vacuuming a floor is one such example, as the floor becomes cleaner the amount of dirt collected per unit time decreases. This is typically referred to as diminishing returns (see Sec. 7.1). A robot in this situation has to decide when it is more economical to stop performing the current task, possibly pay a switching cost, and start at a new task. Two controllers for this case are presented in Chapter 7. This is certainly not limited to work as the same principles

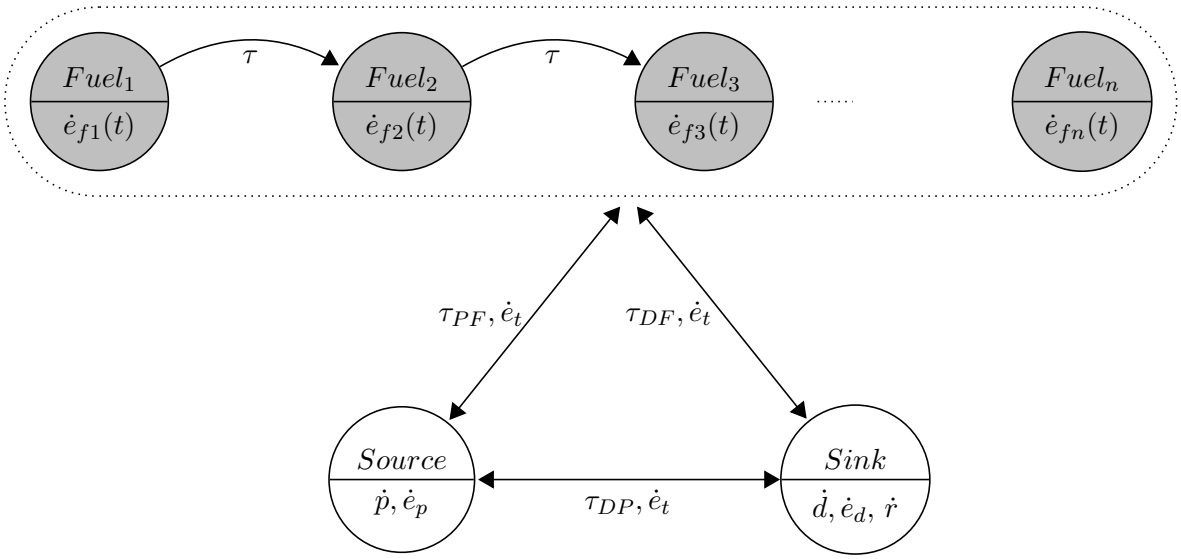


Figure 4.6: Model of a simple transportation task, the robot has to transport goods from a source to a sink, while maintaining self-sufficiency from patchy energy sources subject to diminishing returns

apply in fuelling situations. Batteries for example exhibit diminishing returns, so a robot that can exchange its batteries would have to make the same sort of decisions.

A noteworthy difference between this model and the previous ones is that here we are not switching between obtaining two different resource types. Instead we are aiming to maximize the gain rate from one type.

4.2.7 Diminishing Return and Capacity Limits

A challenging scenario is presented in Fig. 4.6. It is a combination of the transportation task (Sec. 4.2.4) and the diminishing returns scenario (Sec. 4.2.6). Goods have to be transported from source to sink and energy to perform this task has to be gathered from patchy energy sources which are subject to diminishing returns. To keep things tractable, we assume that the distance from the source to any food patch is constant. The same goes for the distance between the sink and any food patch. The per patch gain function $g_f(t)$ is negatively accelerated with patch residence time. The problem lies in the fact the we now have to consider how much onboard energy storage is remaining while we decide whether to switch patches. Previously this was not the case and we could apply a control strategy based on the Marginal-Value Theorem, which no longer applies. For example, while gathering energy, the

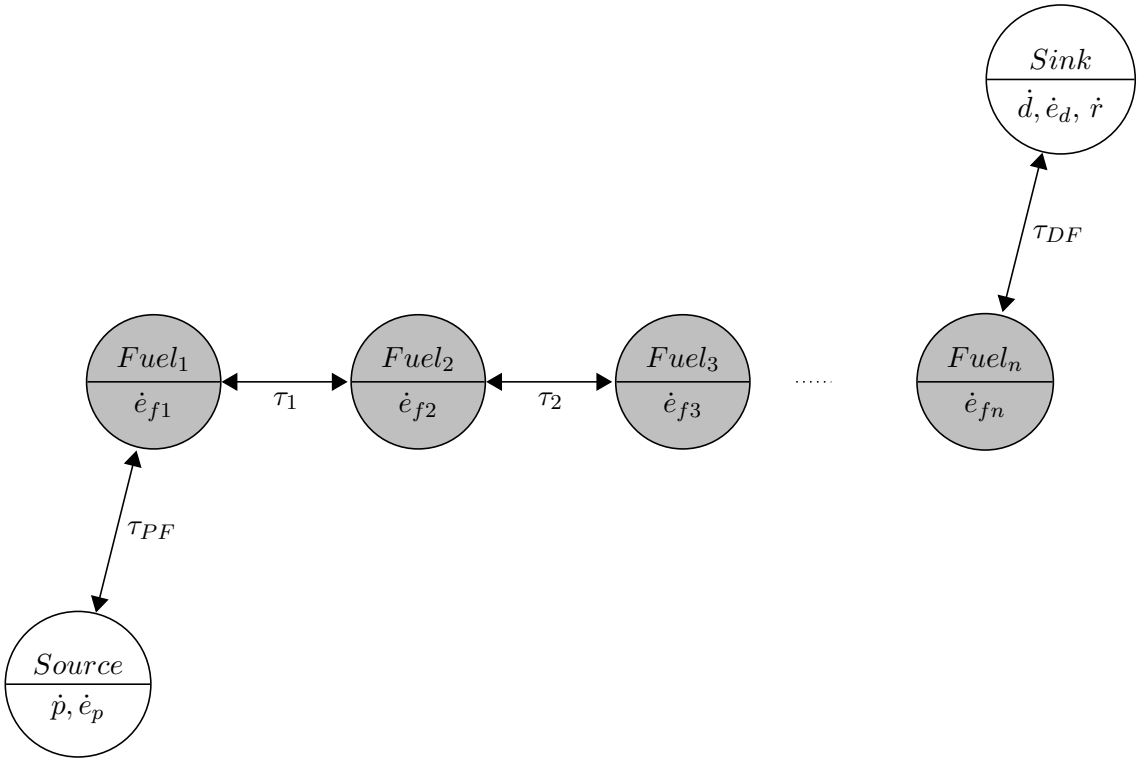


Figure 4.7: Model of a simple transportation task, the robot has to transport goods from a source to a sink. Source and sink are too far appart to be reached with a single fuel load, n pit stops have to be taken to make the trip.

instantaneous gain rate dropped to the global average gain rate, the MVT would indicate to switch patches (see Sec. 7.2 for details). If the energy storage is already 95% full, we can only obtain 5% (plus the energy expended during switching) from the new patch. The switching cost, both in terms of energy expended as well as time lost not working, has to be amortised over the increase in gain rate experienced while obtaining that 5%. In this case it might be better to stay in the current patch longer and fully refuel there. It could also be beneficial to leave the fuelling site, not completely refuelled.

An equivalent problem is given by working, when labour is subject to diminishing returns and the carrying capacity is limited (which it necessarily is for physical objects). For example a robot picking apples in a orchard and having to deposit those apples at the entrance.

4.2.8 Resource Chains

In all transportation tasks considered so far, the robot had enough energy onboard to make the delivery trip without having to refuel several times. In Fig. 4.7 we show a scenario in which the onboard storage capacity for energy is too small to make a trip from sink to source without having to refuel several times. This models, for example, transporting freight across Canada. Note that in this model the cycle cannot be defined by fuelling cycles, instead we use returning to either the source or the sink as the end point of a cycle.

In a simple version of this problem where the energy rates are the same at all fuelling sites and docking is free, one can simply fill up at every fuelling station encountered. If there is a non-zero cost for docking, we can use the adaptive charging method described in Sec. 5.3.3. Because of the docking cost, we need to minimize the number of fuelling stops, which can be accomplished by using an adaptive threshold.

The scenario is more difficult if some fuelling site offer higher rates than others. If the sequence in which the sites are encountered is given, e.g. because the robot has to follow a given trajectory, we can employ an algorithm based on rate-maximization. Let D be the distance the robot can drive with a full energy supply and d be the distance it can move with the remaining energy. At every fuelling site the robot has to decide how much to fuel to take on. The decision is

- fully refuel if the energy rate at the current site is the highest of all sites within D
- refuel so that x units of energy are onboard, if the energy rate at the current site is the highest in d but not in D. x is the amount of energy needed to travel to the fuelling site with the highest rate in D.
- otherwise do not refuel

This method is similar to the one presented in Chapter 5, but it is simpler because only one type of energy source has to be considered and the switching cost is zero.

The scenario becomes a hard planning problem if the trajectory is not given and the robot has to determine it based on the energy rates. More interesting is the case where the energy rates and distance between the fuelling sites are unknown. This situation is related to the zero-one rule (Sec. 3.4.1) but a probabilistic version is needed to handle the short-fall risk (Sec. 3.4.1). We leave solving this problem for the future.

Chapter 5

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

5.1 Introduction

In this chapter we propose a near-optimal solution to the work site chain model introduced in Sec. 4.2.2. We examine this problem using the example of a mobile autonomous robot performing a long-duration survey task. The robot has limited battery capacity so it must recharge from time to time. Time and energy spent in recharging are not available for doing work, so, to maximize the robot's rate of doing work, and thus its utility, we should recharge as efficiently as possible. Computing the perfectly optimal robot behaviour is computationally intractable or impossible in typical circumstances, so we seek useful heuristic methods.

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

Figure 5.1: Application: the USC/UCLA NAMOS robot boat: a mobile aquatic observatory with anchored buoy base-station. A docking/recharging system is under development.

5.1.1 Problem Domain and Motivation

A single, self-sufficient and self-controlling mobile robot must visit a (possibly very long) series of waypoints. The number of waypoints, the energy cost of travelling between them and the robot's limited onboard energy storage capacity mean that the robot must recharge its onboard energy store (e.g. a battery), possibly many times. The series of waypoints could be given *a priori* or could be dynamically generated based on previous data. An example of such a scenario is the Networked Aquatic Microbial System (NAMOS) robotic boat devised by researchers at USC and UCLA (Sukhatme et al.; 2006) that measures the spatio-temporal distribution of plankton and related phenomena in bodies of water. In this application, a series of waypoints is generated by an adaptive sampling algorithm that attempts to maximize the value of the next sample (Zhang and Sukhatme; 2007).

A similar example are planetary exploration missions, of NASA's Spirit and Opportunity rovers currently on Mars (Wikipedia; 2008). Here the waypoints are points of geological interest defined by human experts. In principle these waypoints could be predefined but in practise are likely to be generated as the mission runs.

5.1.2 Related work

Research has been done on many aspects of self-sufficiency, ranging from docking mechanisms and control, action-selection, energy-efficient path-planing, to unconventional robot fuels (Ieropoulos et al.; 2003). The work most relevant here is that on energy-aware path-planning and when-to-recharge decision making.

Aside from the huge literature on general cost-minimizing optimization and path planning, we see some work dedicated to energy gathering and recharging. Wettergreen et al. (2005) present a solar powered robot that plans motion synchronous to the sun in order to maximize energy gathered by solar cells. The problem of path-planning while minimizing energy expense is also discussed by Sun and Reif (2003). Both these authors use conventional planning methods.

Litus et al. (2007) describe heuristic methods to achieve energy-efficient rendezvous between robots, with suggested application to robot-to-robot recharging.

As for deciding when to recharge, the standard method is a fixed threshold policy, where the robot switches to recharging activity once its energy level drops below some predefined threshold. Jung's docking work employs this policy (Silverman et al.; 2002), for example. Austin et al. (2001) have the robot recharge after a constant time has elapsed, approximating the energy threshold while being easier to measure. The threshold approach is attractively simple, but its efficiency depends critically on selecting a suitable threshold. In the experiments below, we use a fixed threshold strategy as a baseline performance target.

An interesting conceptual extension is to a multi-robot threshold. In a multi-robot scenario with one shared charging station, Michaud and Robichaud (2002) use a threshold on the 'remaining operational time' value. This value takes the energetic state of all robots into account and is calculated for each robot. This allows the robots to cooperate and share a charging station.

Conversely Sempe et al. (2002) employ a no-threshold strategy, in which a group of robots have to share a charging station, but no communication is allowed. These robots employ an opportunistic strategy, recharging whenever they sense an available charging station, even if they are not low on energy. This combined with a high work/charge ratio of 8 to 1 decreases the access conflict enough for all robots to be self-sufficient. However, visiting a charging station with a nearly full battery can greatly reduce overall efficiency.

A more carefully motivated approach to decision making under self-sufficiency is discussed by Spier and McFarland (1997). Here a Cue$\times$Deficit model is used to determine the action to be taken (see Sec. 3.5.2). This work is closest to our work, since it investigates choosing between multiple resources. The major difference is that Spier considers multiple resources, for each of which the robot has a certain deficit, whereas we consider one resource: energy, which can be obtained from multiple sources.

5.2 Problem Statement

A mobile robot must visit an ordered series of n work sites $W = w_1, w_2, ...w_n$ in the shortest possible time (see Fig. 4.3). We are especially interested in very large series, so that brute-force planning is infeasible. For simplicity of presentation we assume all points are co-planar, but the solution methods do not require this. We will compare several decision methods, the traditional fixed threshold method and the brute-force method require full knowledge of the work site locations. In contrast the adaptive method only requires knowledge of the next to work site locations and the rate-maximization method only requires of knowledge of the subset of location that are reachable with one battery charge.

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

The robot's recharging model has this simple form:

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

This is a sensible model because there is always a locomotion cost involved in going to the charging station. In order to maximize the return on this cost, the robot should gain as much energy as possible from a charger visit. There is no locomotion cost for using solar cells, but the time spent recharging is lost for working: an opportunity cost. Since in most

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

scenarios the solar current is smaller than the charger current, it makes sense for the robot to only solar charge as long as necessary to reach the next goal location. Thus, when solar charging, sample points and chargers are reached as early as possible and with an empty battery, which maximizes the benefit of a charging station (see chapter 6 for an extensive discussion).

5.3 Solutions

Here we describe four methods for solving the problem described above. These are a brute-force search that gives optimal performance, two naive threshold methods for comparison, and our novel heuristic method.

5.3.1 Brute-force Optimal

Optimal solutions can be calculated with the following simple brute force method. Build a binary tree with the first site w_1 as the root node. The first (left-hand) child node corresponds to site w_2. The cost of traversing from the root to this child is the time required to travel from w_1 directly to w_2 including any required solar cell recharging time (should the distance be greater than that allowed by a single battery charge). The second (right-hand) child node corresponds to the charger location C. The cost of traversing from the root to this child is the time required to travel from w_1 to the charger C then fully recharge the battery and afterwords travel to w_2, again including possible solar charging time. This step is repeated recursively for each child until the last way point is considered. To find the most efficient solution one finds the leaf node with the shortest travel time. Traversing back from this node gives the optimal work-and-charging strategy.

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

5.3.2 Fixed Threshold

Here we use remaining battery charge B as our decision variable. For lack of any general principled approach, we use the following conservative strategy: find the largest distance between any work site and the charger $l = \max\{d(w_i, c)|1 \leq i \leq n\}$ where $d(a, b)$ is the

distance between sites a and b. The charging threshold is set so that the robot has enough energy left to reach the charger even in the worst case, that is the site that is furthest away from the charger. Therefore the threshold is $B_t = \frac{l}{v} I_d$. This means that the robot chooses to travel to the next work site via the charging station if $B < B_t$. Note, this does not guarantee that the robot will always be able to reach the charger, in cases where I_d is high or B_{max} is low, the threshold might actually be higher the B_{max}. In cases like this the robot can rely on power from the sun.

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

5.3.3 Adaptive Threshold

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

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

5.3.4 Rate Maximization

This method is an application of foraging theory (see (Stephens and Krebs; 1986) for an excellent exposition). The *rate-maximizing forager* model attempts to explain the choices animals make when presented with alternative food sources, and maps neatly onto our problem.

Specifically, the rate-maximizing forager model predicts that an animal faced with several

exclusive food options will maximize the net energy rate

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

where e_j is the net energy gain from food item j, h_j is the handling time, λ_j is the encounter rate and p_j is the attack probability for this item. In this model, the attack probability is the control variable by which the animal decides how likely it is to attack a food item. Without reproducing the details of (Stephens and Krebs; 1986) a rate maximizing forager applies the zero-one rule and attacks always only prey that yield a higher net energy rate than the combined rates of all food items that considered individually have a lower rate. Note that this model describes a solitary forager: Seth (2007) provides a probabilistic model that may better describe the behaviour of multiple foragers in a competitive situation.

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

$$R_{c_i} = \frac{I_s T_{s_i} + I_c T_{c_i} - I_d T_{d_i}}{T_{s_i} + T_{c_i} + T_{d_i}} \tag{5.1}$$

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

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

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

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

Now the robot can select the option which provides the highest energy gain rate, but it has more options to choose from if it's battery is full enough. What about the next path segment or the one after that? At any given work site, the robot only has to make a decision about going direct or via the charger to the next site. So if the robot chooses to go via the charger we can stop our rate analysis for this segment, since future segments do not

influence the robot's decision anymore. But if we do not recharge, it is worth evaluating the energy gain rate of the next segment. Again going from w_{i+1} directly to w_{i+2} is trivial $R_{d_{i+1}} = I_s$. In the other case where the robot travels via the charger to w_{i+2} we use Eq. 5.1. But this time for the segment $S_{i+1} = \overline{w_{i+1}w_{i+2}}$ and we also have to calculate B_{i+1} by subtracting what we spend on the first segment so $B_{i+1} = B_i - \frac{|\overline{w_i w_{i+1}}|}{v} I_d$. This way we can iteratively calculate the energy rate for each segment. We can stop this iteration once the projected battery level B_k reaches zero. Because the robot has to recharge at this point, future segments do not influence the current decision.

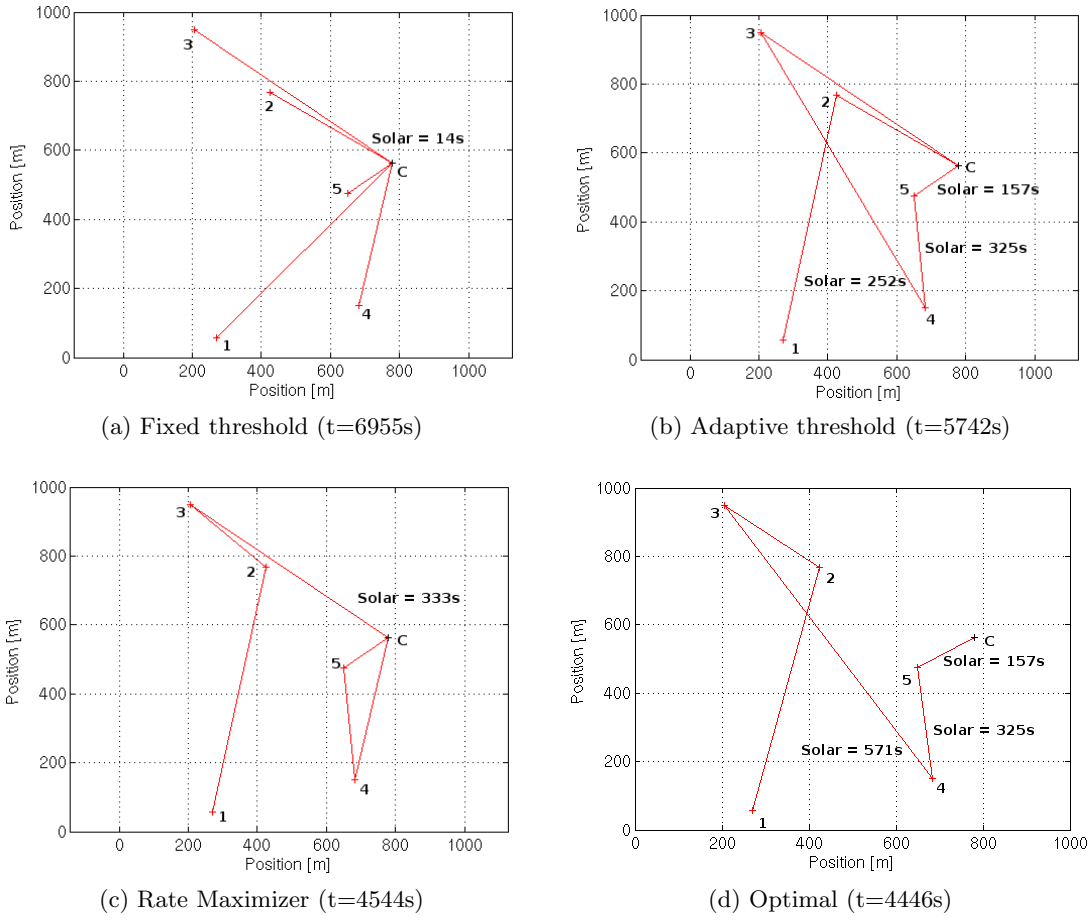
The rate-based decision is simply, if the energy gain rate from the charger option of the current segment $R_{c_i} = \max\{I_s, R_{c_i}, R_{c_{i+1}} ... R_{c_k}\}$ choose to recharge from the charger now, otherwise proceed directly to the next waypoint and re-iterate.

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

Rate maximizing foraging animals are clearly not limited to two energy resources, as Stephens and Krebs (1986) show. We restricted ourselves because of scalability issues of the optimal method. But extending the rate maximizer to deal with m energy sources is straight forward. On each segment s_i we calculate the energy gain rate R_{j_i} for each energy source j. Combinatorial problems do not occur since we terminate the current analysis on each branch on which the robot recharges. As in the two resource case we interactively calculate rates till the battery is depleted. The charging option with the highest rate is the one the robot selects. In this case the complexity is $O(knm)$ with k being the number of work sites the robot can visit with one battery charge and n is the total number of work sites of the trajectory.

5.3.5 Example solutions

Before discussing the statistical results, we give an example of the type of plans produced. The example problem shown in Fig. 5.2, with five work sites, one charging station, is interesting in that each of the four methods produced a different solution, each very typical for individual weaknesses. The robot starts with a full battery at site 1. To ensure all robots finish in an identical state (which masks artifacts caused by experiments with few work sites), the charger C is the final location and the trial ends when the robot is totally charged.



(a) Fixed threshold (t=6955s)

(b) Adaptive threshold (t=5742s)

(c) Rate Maximizer (t=4544s)

(d) Optimal (t=4446s)

Figure 5.2: An example problem, where each of the four methods found a different solution.

The fixed threshold method (Fig. 5.2a) demonstrates the conservatism of its high threshold setting, with the robot visiting the charger C 5 times: once between every waypoint. Nevertheless, due to the remoteness of work site 3, the robot has to solar charge for 14 seconds on the return trip from 3 to C. The complete run takes 6955 seconds. The adaptive threshold method (Fig. 5.2b) visits the charger between sites 2 and 3. It spends a long time solar charging towards the end of the run, which takes 5742 seconds to complete. The rate maximizer method (Fig. 5.2c) takes only 79% of the adaptive threshold time. It recharges slightly later: in between site 3 and site 4, taking a detour that requires a solar charge, but having visited the charger empty, the robot completes the run without charging again. The optimal method (Fig. 5.2c) is very slightly faster than the rate maximizer, taking 99% of

the time. The brute-force search found a solution that recharges only at the final return to the charger, and uses the solar cells to arrive there empty.

5.4 Experiments

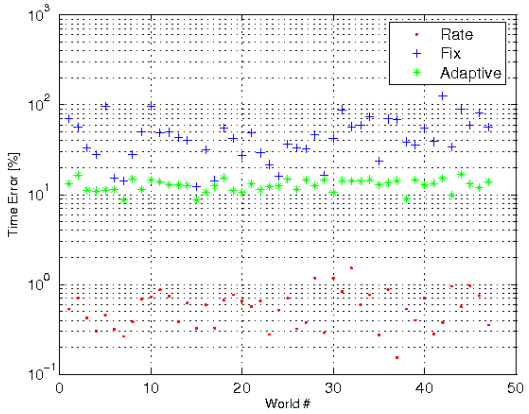
5.4.1 Experiment 1: series of 20 randomly-placed points

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

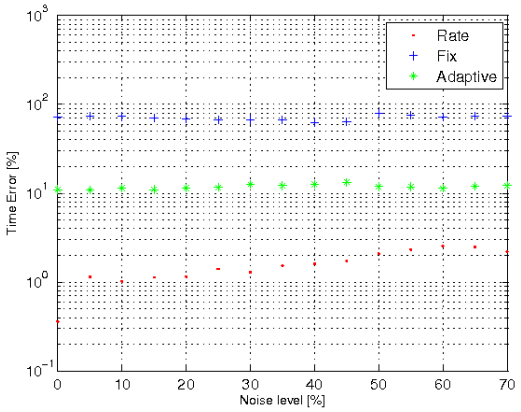
We explored the performance of all four methods under a wide variety of parameters. The number of sites and the charging station current were fixed. The rest of the setup was as follows: (i) generate a world with 20 work sites and a single charging station placed at random (drawn from a uniform random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

Presenting these results is challenging, since so many parameters are varied. The absolute values of performance are not very meaningful since it depends on the length of the trajectory which is randomly generated. Therefore, for each trial we subtract the time taken by the optimal method, to give an error signal which we can use to assess the heuristic methods.

Figure 5.3a shows the mean error of each method compared to the optimal solution averaged over all parameter sets in each of the 50 worlds. It shows that the adaptive threshold performs better (has smaller error) then the fixed threshold method and that the rate maximizer performs better then the other two methods. Table 5.1 (Experiment 1) gives the mean and standard deviation of the time error for all experiments. The distribution of these errors is depicted in Fig. 5.4, which shows a histogram of the error. Note the axes are scaled differently on all the graphs.



(a) Exp. 1: The mean error, compared to optimal performance, of each of the three heuristic methods, over a range of parameter settings. Note the Y axis is a log scale



(b) Exp. 3: Performance of the Rate Maximizer under cost estimate error.

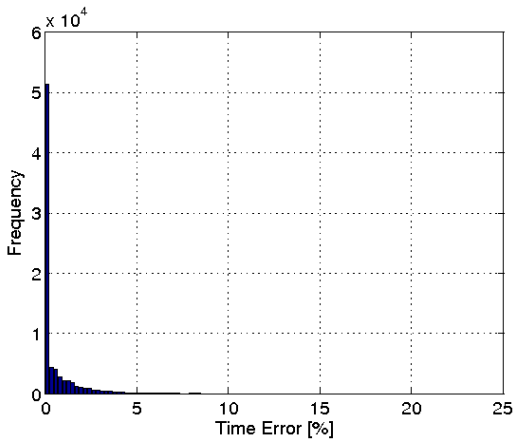
Figure 5.3: Comparing results from heuristic methods to bruteforce

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

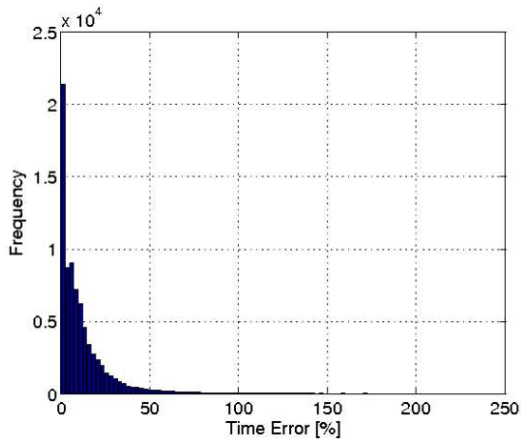
Table 5.1: Summary of statistical results for all three experiments

5.4.2 Experiment 2: series of 1000 randomly-placed points

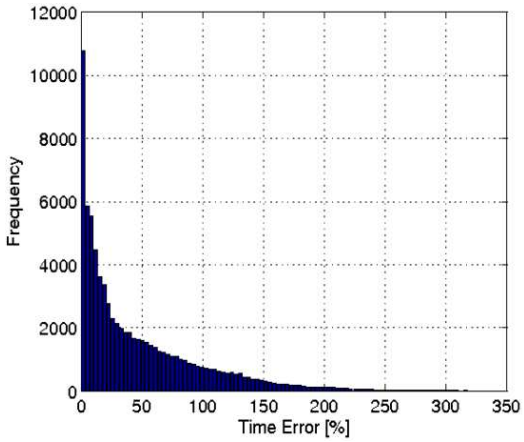
To evaluate the methods' performance in long-duration tasks, we generated 100 trajectories with 1000 work sites each and ran the three heuristic methods over all parameter combinations for each trajectory. Here it is impossible to compare the performance of each method to an optimal solution, since it could not be calculated in a reasonable amount of time for these large problems. Instead we compared the threshold methods against the rate method. Table 5.1 (Experiment 2) shows how the threshold methods performed compared to the rate maximization. On average the rate maximizer performs 14% better then the adaptive threshold and 55% better then the fixed threshold method.



(a) Rate Maximization



(b) Adaptive Threshold



(c) Fixed Threshold

Figure 5.4: Histograms of the relative error. The differences are so large that the axes are scaled differently to preserve visible detail.

Pattern & charger pos.	μ_{rate}	σ_{rate}	μ_{ada}	σ_{ada}	μ_{fix}	σ_{fix}
Meander beginning	0.23	0.97	12.38	15.54	144.21	103.26
Meander middle	0.62	0.97	10.59	18.75	40.50	45.75
Meander end	1.12	1.89	13.33	17.27	129.51	89.25
Circle center	1.06	4.18	36.27	69.27	169.55	244.08

Table 5.2: Average time error for regular trajectory pattern

5.4.3 Experiment 3: robustness to cost estimate error

We repeat experiment 1, this time randomly varying the actual cost of driving the robot between work sites, to simulate unbiased error in the robot's *a priori* cost estimates. The error was varied from 0% to 70% in 5% intervals. 20 simulations per noise level and parameter set were performed.

As Table 5.1 (Experiment 3) shows, the performance of the rate maximizer decreases with increased cost estimate error. However, the rate maximizer still performs significantly better than the other methods. And as Fig. 5.3b shows, the error increases with increasing noise but remains relatively small, suggesting that the rate maximizer is fairly noise tolerant.

5.4.4 Experiment 4: series of 20 points in regular pattern

We repeated Experiment 1 with various regular patterns of work sites, to see if the geometry of the world would influence the results. The setup is as before with the exception that we use four different patterns, the first three are rectangular meander patterns where we place the charger either at the beginning, middle or end. The fourth is a circle with the charger in the center. Results are given in table 5.2 and show that regularity in the trajectory has no influence on the performance.

5.5 Summary

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

The idea behind the solution is to maximize the local *rate* of energy intake. Given finite energy storage capacity this has the natural result of maximizing the rate at which energy

can be spent on work. Thus by adopting this strategy we observe that the rate of doing work of the simulated systems we observed is very close to optimal. Yet the method is scalable, i.e. requires computation and memory independent of the length of the problem. It is simple to implement and appears to be reasonably resistant to error in cost estimates. The method dramatically outperforms naive threshold-based heuristics.

Chapter 6

Optimal Recharging Strategies For Time Discounted Labour

In this chapter we present a robot control strategy to handle a WF-model introduced earlier. In particular we investigate a model in which the reward obtained by working is discounted for time. To keep the analysis tractable we let the transition costs between working and fuelling behaviour be symmetrical.

The laws of physics dictate that energy cannot be transferred instantaneously, or in other words, refuelling takes time and this time cannot be spent working. If a robot spends an hour refuelling, it starts to work one hour later and since the reward is discounted over time, it receives a smaller payment then if it would have started working immediately. But the initial charging period is strictly required, as no work can be done without previously obtaining energy. This conflict between the mutually exclusive tasks of refuelling and working raises two interesting questions:

Q1 How much energy should be accumulated before starting work?

Q2 At what remaining energy level should the agent switch back to obtaining energy?

Most real-world robot systems avoid these questions by maintaining a permanent connection to an energy source, e.g. industrial robotic manipulators wired into the mains power grid, or solar powered robots which are capable of gathering energy while performing some

task at the same time. This chapter addresses the more interesting class of machine, including animals and mobile robots, that must obtain and store energy prior to working. The Q1, Q2 action selection problem must be solved by every animal and long-lived robot in some way or another. Further we are considering only rational agents. Any introductory textbook on decision making (e.g. Stuart and Peter (2003)) defines an agent to be rational if it always selects that action, i.e. an answer to Q1 and Q2, that returns the highest expected utility. Here we assume that utility is proportional to the reward obtained by working, which is discounted over time. By discounting future rewards we calculate the present day value of these future rewards. This allows us to analyze the values of rewards obtained at different points in time and make a decision based on a time-corrected, and thus fair, comparison of reward values. We are therefore able to make decision like whether we should prefer larger but later rewards to smaller but sooner rewards. Note that we use exponential discounting, since this is the standard in economics (Varian; 1992). For more details on discounting see Sec. 3.4.2.

6.1 The Model

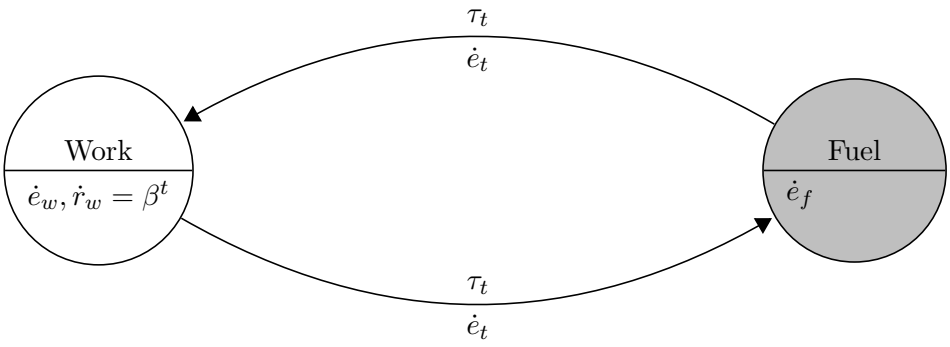


Figure 6.1: Refuel and time discounted labour model, see text for details.

In this section we describe the behavioural model used in this work. In order to keep the analysis tractable we choose an abstract, slightly simplified model. The world is modelled as two distinct spatially separated sites: a work site and a fuelling site. Moving between sites has a non-zero cost (Fig. 6.1). The robot has a energy storage of $E(t)$, where $0 \leq E(t) \leq E_{max}$. If the energy supply drops to zero anywhere but at the refuelling site, the robot loses its ability to move or work and can gain no more reward. The robot can be in

one of four states:

- **refuelling** with a fuelling rate of $\dot{e}_f$, to do so the robot has to be at the fuelling site.
- **transitioning from the fuelling site to the work site**, the duration of this transition is τ_t and the robot has to spend energy at a rate of $\dot{e}_t$, so the transitions cost in terms of energy is $\tau_t \cdot \dot{e}_t$
- **working**, which gets the robot a reward of

$$R = \int_{t_0}^{t_0+\tau_w} \beta^t dt \tag{6.1}$$

  where t_0 is it time when the robot starts to work and τ_w is the duration the robot works for. Therefore the reward the robot earns by working is discounted with a discount factor $0 < \beta < 1$. While working the robot spends energy with a rate of $\dot{e}_w$. In other words, the robot turns energy into work and therefore reward. In the case where the robot performs several work sessions, the reward is accumulated and only the overall reward is of interest to the owner of the robot. As with refuelling, work can only be performed at the work site.
- **transitioning from the work site to the fuelling site**, the duration of this transition is τ_t and the robot has to spend energy at a rate of $\dot{e}_t$

The robot's goal is to achieve as much reward as possible. To do so, it has to make two decisions,

- when to stop refuelling and resume work
- when to stop working and refuel.

We mostly refer to the action of accumulating energy as *refuelling* and not as *recharging* because we want to emphasis the general nature of our model.

It is worth pointing out that in a real world scenario all important variables, namely the energy rates, could be known in advance or are easily measured by the robot. Here we assume these variables to be constant, though in an actual implementation we would use averages as approximations. It would also be feasible to do some form of piece-wise linear approximation of the energy rates. The discount factor can also be assumed to be known, since this factor is task dependent and, hence, is set by the owner or designer of

the robot or by some external market. As we show below, even if all else is fixed, the robot owner can use the discount factor as a control variable that can be tweaked to fine tune the robot's behaviour. Everything else is predefined by the tasks, the robot's construction, or the environment.

In order to improve readability, we need to introduce some additional notation. $k_1 = \frac{\dot{e}_f}{\dot{e}_w}$ is the ratio of the energy rate while refuelling to the rate while working. Similarly, $k_2 = \frac{\dot{e}_t}{\dot{e}_w}$ is the energy rate while transitioning over the energy rate while working. $k_3 = \frac{\dot{e}_t}{\dot{e}_f} = \frac{k_2}{k_1}$ is the ratio of the energy rate while in transition and the energy refuelling rate. τ_f is the time spent refuelling during one fuel-work cycle. The amount of work the robot can perform is limited by the energy supply the robot has, so we express the potential work duration as a function of refuelling and transitioning time where $\tau_w = \tau_f k_1 - 2\tau_t k_2$. This is essentially the amount of time the robot can work for given the amount of energy the robot got from refuelling minus the energy the robot has to spend to travel to the work site and back to the charging station. We also introduce the period of time $T = \tau_f + 2\tau_t + \tau_w = \tau_f(1 + k_1) + 2\tau_t(1 - k_2)$ as the length of one refuel-work cycle.

6.2 When to stop working

Let $e(t)$ be the energy in the robot's storage at time t. At what energy level $e(t) < \epsilon_{w\to f}$ should the robot stop working and transition to the fuelling site? Since the value of work is time discounted, work that the robot performs now is always more valuable then the same amount of work performed later. This creates an inherent opportunity cost in transitioning from the work site to the refuelling site because it takes time and costs energy $\tau_t \cdot \dot{e}_t$ that cannot be spent working. This implies that the robot needs to work as long as possible now and not later. Hence the only two economically rational transitioning thresholds are:

- $\epsilon_{w\to f} = \tau_t \cdot \dot{e}_t$

  The robot stops working when it has just enough energy left to make it to the fuelling station. The robot will spend the maximum amount of energy, and therefore time, working, ensuring the highest reward before refuelling. Comparatively, should a higher transitional threshold be used, the robot would stop working earlier and refuel earlier, but discounting results in a smaller reward. Should the transitioning threshold be smaller, the robot would have insufficient energy to reach the refuelling station. In this case, the robot cannot gain any further reward because it runs out of fuel between

the work and refuel sites.

- $\epsilon_{w \to f} = 0$

  The robot spends all of its energy working and terminates its functionality while doing so. At first glance this option seems counter intuitive, but one can imagine highly discounted labour situations, such as rescue missions, where the energy that would otherwise be spent on approaching a refuelling site is better spent on the task at hand. This might also be a rational option if the transition cost is very high, e.g. NASA's Viking Mars lander (Wikipedia; 2010) took a lot of energy to Mars in the form of a small nuclear reactor, rather than returning to Earth periodically for fresh batteries (the recent Mars rovers (Wikipedia; 2008) employ solar cells to recharge their batteries originally charged on Earth).

6.2.1 Suicide or live forever?

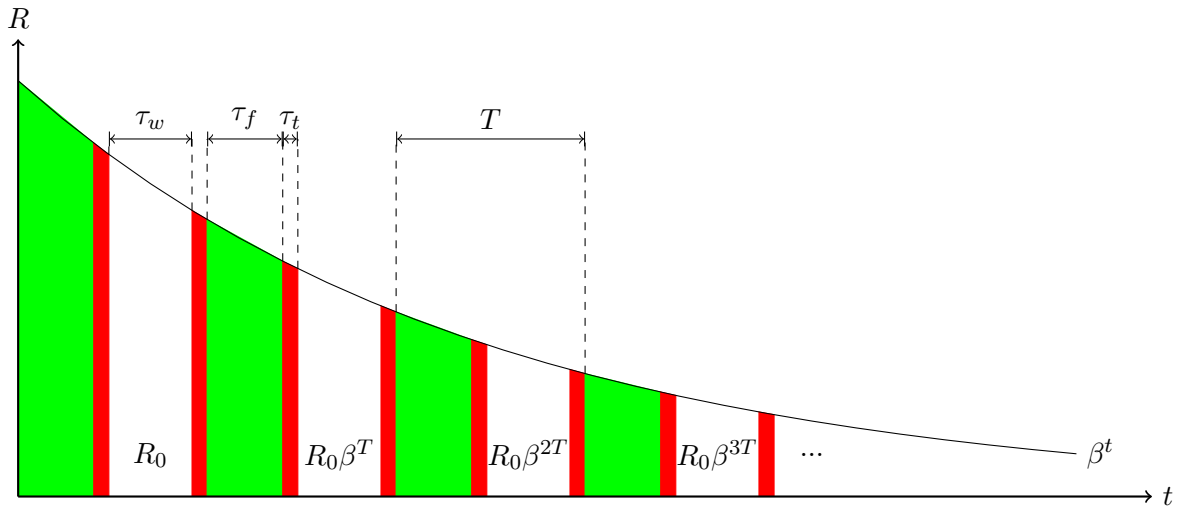


Figure 6.2: General discounting in work - fuel cycles. The green areas are periods in which the robot refuels for τ_f units of time. Areas marked in red areas are times of travel between the work and the fuelling site, with a duration of τ_t. The white areas are times the robot is at the work site and thus earns a discounted reward for a period of τ_w units of time. The overall cycle time is T.

The question we address in this section is whether the robot should spent all its energy on the task at hand and therefore terminate its functionality once the energy is spent or if

it should save a part of the energy in order to obtain additional energy. The second option then allows the robot to perform more work in the future.

Animal life-histories are subject to a similar problem. Here the question is whether to follow a semelparous strategy (reproduce once and then die) or to employ a iterparous strategy (reproduce multiple times). An example for a semelparous species is the Pacific salmon. While most mammals including humans are iterparous organisms. Cole (1954) analyzed which strategy yields the larger population growth. He concluded, "For an annual species, the absolute gain in intrinsic population growth which could be achieved by changing to the perennial reproductive habit would be exactly equivalent to adding one individual to the average litter size". This is known as *Cole's Paradox*. About twenty years later Charnov and Schaffer (1973) resolved this paradox by pointing out that Cole assumed adult survival rate P to be equal to offspring survival rate C. But since adult survival rate is usually much larger than the offspring survival rate $P >> C$, iterparity should be favoured. In fact the average litter size in the case of a semelparous strategy should be $\frac{P}{C}$ larger than in the iterparous case.

An investigation of life-histories in terms of reproductive success is not suitable for robots, which usually do not reproduce. In the following we use our discount labour model to determine whether a robot in a given scenario should terminate while working or continue indefinitely with the work-refuel cycle. Let

$$R_0 = \int_{\tau_f+\tau_t}^{\tau_f+\tau_t+\tau_w} \beta^t dt = \beta^{\tau_f+\tau_t} \frac{\beta^{\tau_w} - 1}{\ln(\beta)} \tag{6.2}$$

be the reward obtained from spending $E_{max} - \epsilon_{w\to f}$ energy or τ_w time during the first working period (see figure 6.2). In this figure the white areas correspond to time in which the robot performs work and thus obtains a reward proportional to the size of the white area. Later work periods are discounted more strongly and hence provide a smaller reward. Shaded areas correspond to times in which no reward is earned because the robot either travels between the work and refuelling site, marked as red or it refuels coloured in green. The size of the sum of these areas is proportional to the opportunity cost, that is, reward that, in principal, could have been obtained if the time had been spent working.

Let T be the duration of one full work-fuel cycle, that is working - transition - refuel - transition, or $T = \tau_w + \tau_t + \tau_f + \tau_t$. Therefore, the reward gained in the next cycle is the initial reward R_0 discounted by T and becomes $\beta^T R_0$. Subsequent rewards are again discounted by T and so the reward for the third cycle is $\beta^{2T} R_0$. The sum of all rewards if

working infinitely, that is choosing $\epsilon_{w\to f} = \tau_t \cdot k_2$, is

$$R_\infty = R_0 \sum_{i=0}^{\infty} \beta^{iT} = R_0 \frac{1}{1-\beta^T} \tag{6.3}$$

In practise no system will last forever, so this analysis is slightly biased toward infinite life histories.

If the robot chooses $\epsilon_{w\to f} = 0$ it gains the initial reward R_0 plus a one time bonus of

$$R_+ = \int_{\tau_f+\tau_t+\tau_w}^{\tau_r+\tau_t+\tau_w+\tau_t k_2} \beta^t dt = \beta^{\tau_f+\tau_t+\tau_w} \frac{\beta^{\tau_t k_2} - 1}{\ln(\beta)} \tag{6.4}$$

by spending the energy required for transitioning on working. The reward gained over the lifetime of the robot (which is fairly short) is $R_{rip} = R_0 + R_+$.

So the answer to Q2 (at what energy level should the agent switch back to refuelling?) is that the rational robot selects that threshold $\epsilon_{w\to f}$ that achieves the higher overall reward, so it picks

$$\epsilon_{w\to f} = \begin{cases} 0 & : \quad R_{rip} \geq R_\infty \\ \tau_t \dot{e_f} & : \quad R_{rip} < R_\infty \end{cases} \tag{6.5}$$

Since the discount function $\int \beta^t dt$ belongs to the class of memory-less functions, we only have to calculate Eq. 6.5 once, in other words if it is the best option to refuel after the first work cycle it is always the best option to do so and vice versa.

6.3 How much energy to store

We have shown how to determine a threshold for transitioning from work to fuelling. In this section we will analyze when to stop refuelling and resume work, or phrased differently, how much energy to accumulate before starting to work. Energy and time are interchangeable elements, provided that we know the rate at which energy is spent and gained. Since discounting is done in the time domain, our analysis equates energy with time for simplicity. Based on this, we can ask the time equivalent of Q1: 'how long should the robot refuel for?' We call this refuelling duration τ_f. To be rational, the robot must refuel long enough to gain enough energy to make the trip to the work site and back, that is $2\dot{e}_t \cdot \tau_t$, otherwise it would have to turn around before reaching the work site and thus will not gain any reward. Refuelling after the storage tank is full is time wasted that would better be spent obtaining a reward. Therefore the refuelling time is limited to $2\tau_t \cdot k_3 \leq \tau_f \leq \frac{E_{max}}{\dot{e}_f}$. In the following

we assume, without loss of generality, the robot starts at the refuelling site with an empty fuel tank. Assuming differently will just result in shift of the analysis by a constant factor, but will not change the overall conclusions.

6.3.1 Acyclic tasks

First we examine situations in which the robot has to refuel for a task that has to be done only once, that is the robot refuels, performs the task, and returns to the refuelling site. Depending on the time spent refuelling the robot obtains the following reward during the upcoming work period.

$$R(\tau_f) = \int_{\tau_f+\tau_t}^{\tau_f+\tau_t+\tau_w} \beta^t dt = \beta^{\tau_f+\tau_t} \int_0^{\tau_f k_1 - 2\tau_t k_2} \beta^t dt \tag{6.6}$$

Next we need to find the $\hat{\tau}_f$ that maximizes $R(\tau_f)$, which is

$$\hat{\tau}_f = argmax(R(\tau_f)) = \frac{\log_\beta \left(\frac{1}{k_1+1}\right) + 2\tau_t k_2}{k_1} \tag{6.7}$$

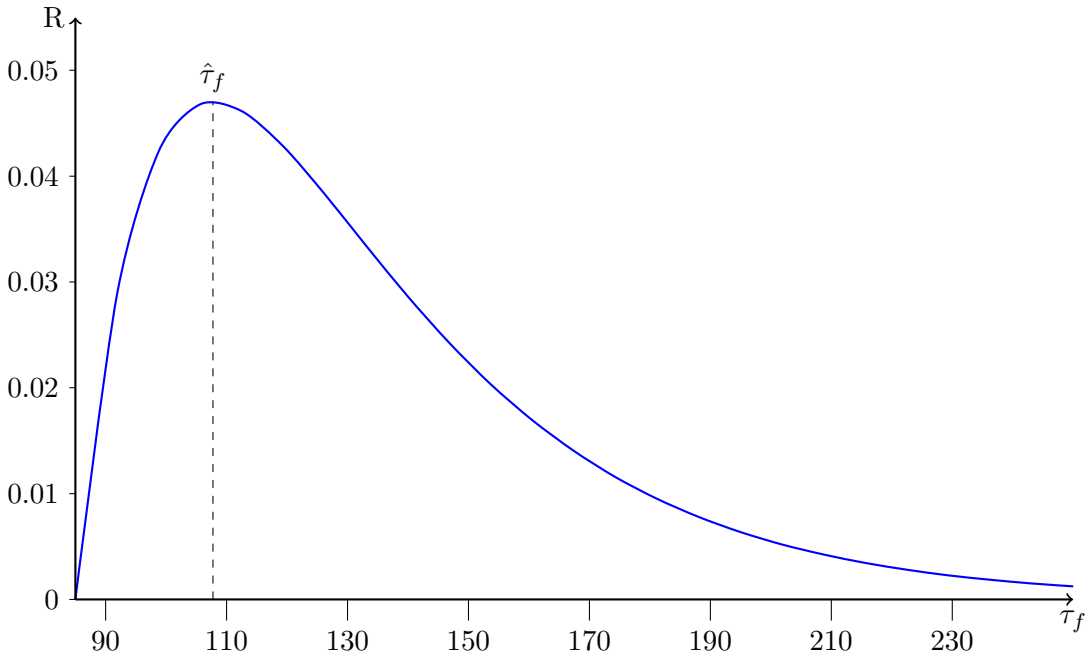


Figure 6.3: Reward depending on refuelling time with an example configuration $k_1 = 1.0, k_2 = 0.5, \beta = 0.97, \tau_t = 85s$

Figure 6.3 shows an example reward function (Eq. 6.6) depending on the refuelling duration τ_f. Using Eq. 6.7 we calculate that for this particular configuration the reward is maximized when the robot refuels for $\hat{\tau}_f = 107.756....$ If the fuel tank is filled before that time, the best the robot can do is return to work. This will give it the highest reward achievable, but the robot designer should keep in mind that there might exist a class of robots with a larger fuel tank that will achieve a higher reward. Note that if the robot stops refuelling at $\hat{\tau}_f$ even if its energy store is not full to capacity, and transitions to working, it earns the highest reward possible. To our knowledge this has not been stated explicitly in the robotics literature before Wawerla and Vaughan (2008) did so. It is generally assumed that robots should completely recharge at each opportunity, but this is not always the optimal strategy.

6.3.2 Cyclic Tasks

In cyclic tasks a robot is required to always return to work after resupplying with energy. Here the analysis is slightly different then in the acyclic case because the refuelling time of the current cycle not only influences the duration and length of the work period of this cycle but of all cycles to come. Hence, we should select a refuelling threshold that maximizes the overall reward. The overall reward is calculated by (see Fig. 6.2)

$$R_\infty(\tau_f) = R_0 \sum_{i=0}^{\infty} \beta^{iT} = \frac{(\beta^{\tau_w} - 1)\beta^{\tau_f + \tau_t}}{(1 - \beta^T)\ln(\beta)} \tag{6.8}$$

Unfortunately, it seems impossible to find a closed form solution to $\hat{\tau}_f = argmax(R_\infty(\tau_f))$. However, Eq. 6.8 can easily be evaluated for a given τ_f and so calculating the reward for each of a finite set of values for τ_f and selecting the one that maximizes the reward is quite practical. In any real application the number of τ_f to be tested is limited and possibly rather small, in the order of a few thousand. This is because any real robot will have a finite energy storage and any practical scenario will require only limited sampling due to the resolution of the fuel gauge, the uncertainty in the environment, etc. In the case of our Chatterbox Robot (see below), the nominal battery capacity is 2.8 Ah and the fuel gauge has a resolution of 1 mA, resulting in less than 3000 calculations for an exhaustive search.

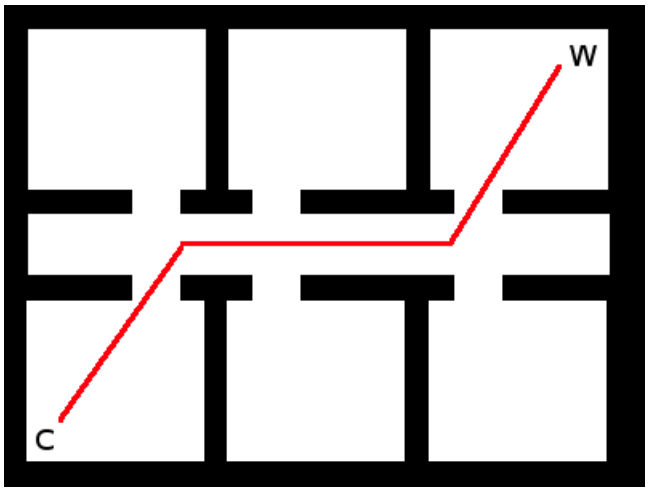


Figure 6.4: Office like environment with a charging station 'C' and a work site 'W'. The red line is the stylized path the robot travels on.

6.4 Experiments

In this section we present experiments to validate the theoretical results described in detail above. All experiments were performed using the robot simulator Stage[1] (Vaughan; 2008). The simulated robot uses simulated electrical energy, where we assume charging and discharging to be linear, with constant current for charging I_c, working I_w and driving I_d. We further ignore any effects caused by the docking mechanism, change in battery chemistry or ambient temperature.

In all experiments we roughly model a Chatterbox robot, a robot designed and built at Simon Fraser University based on an iRobot Create platform. This robot has a battery capacity of approximately 2.8 Ah and draws about 2 A while driving. We defined an abstract work task which consumes 4 A of current. Once at a charging station, the robot docks reliably and recharges with 2 A. The world the robot operates in is office-like with one charging station and one work site shown in Fig. 6.4. The obstacle avoidance and navigation controller drives the robot from the charging station to the work site and vice versa in approximately $\tau_t = 85s$. Due to naturally occurring noise in the experimental setup the travel time may vary by up to 6 seconds. While working, the robot receives one unit of reward per second, discounted by β. Discounting occurs on a one second basis.

[1] http://playerstage.sourceforge.net

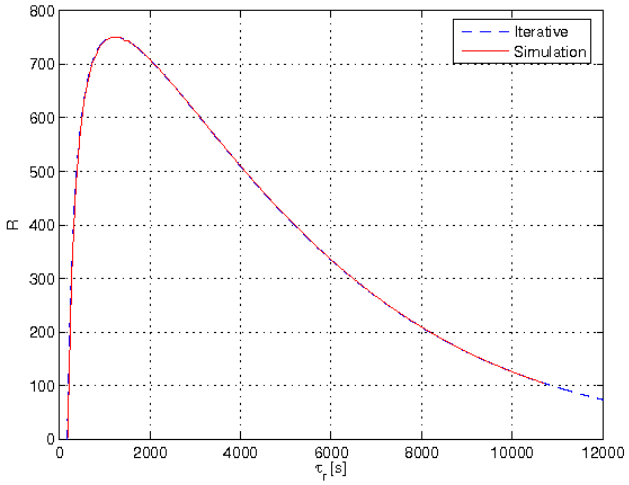


Figure 6.5: Comparing analytical and simulation results for accumulated reward from a **cyclic** task depending on refuelling time with an example configuration $I_c = 2.0, I_d = 2.0, I_w = 4.0, \beta = 0.9997, \tau_t \approx 85$

6.4.1 Cyclic Task Experiments

The goal of this experiment is to evaluate how closely our analysis from section 6.3.2 matches a robot in a simulated environment. In this experiment the robot's task is to recharge for some time τ_f, proceed to the work site, work until the battery energy drops to $\epsilon_{w\to f} = \tau_t I_d$, return to the charging station, and repeat the process. The reward for work is discounted by $\beta = 0.9997$. To find out which τ_f maximizes the reward we varied the threshold for leaving the charging station $\epsilon_{f\to w} = \tau_f I_c$ in each try. A trial lasted for 50000 seconds ($\approx$ 13.8 hours). Fig. 6.5 compares the accumulated reward gained over τ_f from the simulation and from the best solution obtained from the model by iterating over Eq. 6.8. The recharging time that maximizes the reward is predicted by the model to be $\hat{\tau}_f = 1219$. In simulation the reward was maximized by a charging time of $\hat{\tau}_f = 1170$. The difference comes from the variation in time, and therefore energy, the robot requires to travel between the charging station and the work site. Not only does this change the starting time of work which influences the reward, it also makes it necessary to give the robot a small amount of spare energy to ensure it would not run out of battery. This, in turn, delays charging and thereby influences the reward gained. However, the empirical results agree qualitatively with the values predicted by the model, and the optimal recharging time predicted by the model was within 4% of that observed in the simulation.

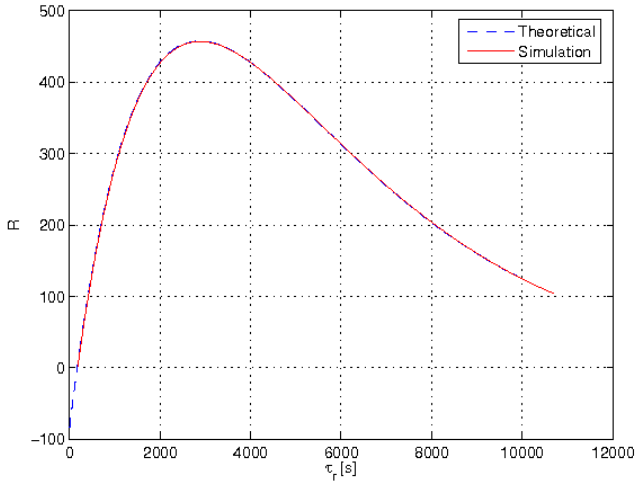


Figure 6.6: Comparing analytical and simulation results for accumulated reward from an **acyclic** task depending on refuelling time with an example configuration $I_c = 2.0, I_d = 2.0, I_w = 4.0, \beta = 0.9997, \tau_t \approx 85$

6.4.2 Acyclic Task Experiments

As before, we perform this experiment in order to compare the theoretical results with a simulation. The setup is the same as in the cyclic task experiment with the difference that the robot only has to perform one charge-work cycle. Figure 6.6 compares the simulation results to the analytical results. Where the general shape of the curve is similar to that in the cyclic task, it is worth pointing out that the maximum reward is gained with a larger charging threshold. This is intuitively correct as the robot has only one chance to obtain a reward. It can be (depending on the discount factor) beneficial to begin work later, but to work for a longer period. For our configuration, the most profitable theoretical charging time is $\hat{\tau}_f = 2872.7$ and the best simulation results were obtained with $\hat{\tau}_f = 2880$. Again the difference between the theoretical and experimental results, barely visible in the plot, are due to uncertainty in the robot simulation.

6.4.3 Once or Forever Experiments

In a further experiment we investigate the circumstances under which it is more profitable, and hence rational, for the robot to fully deplete its energy supply while working and when it is better to choose a perpetual refuelling policy. As in the previous experiments we use a simulated Chatterbox robot with the previously described parameters in the office-like

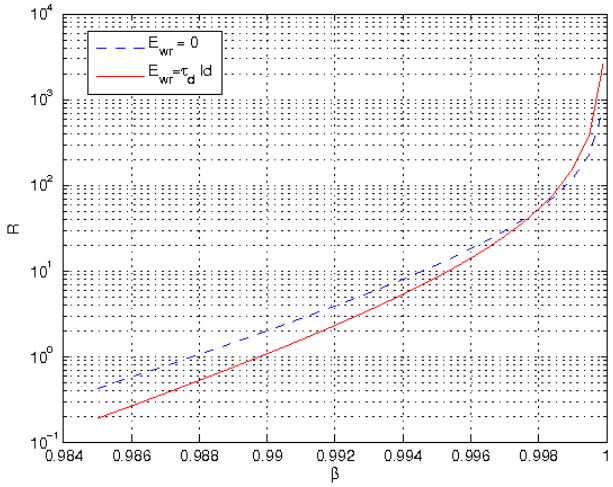


Figure 6.7: Reward obtained for different discount factors with two leave work thresholds. Configuration $I_c = 2.0, I_d = 2.0, I_w = 4.0, \tau_t \approx 85$

environment. For this scenario, we varied the discount rate between 0.9850 and 0.9999 in 0.0005 increments and ran two sets of simulations. In the first, the robot depletes its energy supply while working, that is, we choose the leave work threshold $\epsilon_{w\to f} = 0$. For the second set, we choose $\epsilon_{f\to w} = \tau_t \dot{e}_t$, a leave work threshold that causes the robot to keep performing work-fuel cycles forever. Since we change the discount rate we have to adapt the leave refuelling site threshold in order for the robot to earn the highest possible reward. For this we determine the optimal threshold in the same way as for the previous experiments. Figure 6.7 depicts the rewards obtained for different discount factors with each policy. As the graph further shows, for higher discounting (smaller discount rate), it is beneficial for the robot to choose a one time work policy. Conversely, for smaller discounting (higher β), it pays to keep working. The theoretical discount rate for switching the policy from one work period to an infinite work refuel cycle is $\beta = 0.9979$, which, as the graph shows, closely resembles the experimental result.

6.5 Summary

We outlined a theoretical analysis of when to refuel and for how long to refuel a robot in situations where the reward for the robot's objective is exponentially discounted. This discounting is, more often then not, ignored in robotics literature, although it is at the

very base of rational behaviour (Stuart and Peter; 2003). We took theoretical results and demonstrated that they apply to a simulated robot. In these simulations we assumed the location of and the distance between work and refuelling station to be known. This is reasonable given the state of the art in mapping and localization, for a wide range of scenarios. We further assumed the average energy spending rates to be constant and known, something achievable in most cases. One assumption made that simplifies a real-world robot scenario is constant refuelling rates. Gasoline-powered vehicles which refuel from a standard gas station have a constant refuelling rate, or close to it. However, the charging rate of a battery may depend on many factors including the charging method used, temperature, battery chemistry, and the present capacity of the battery. One useful extension of our model would be to include a realistic chemical battery recharge transfer function.

This chapter has presented and analyzed a core action selection problem for autonomous agents such as animals and mobile robots: how much to fuel before working, and when to abandon working and return to fuelling such that the value of discounted work is maximized. A simple model readily provides answers to these questions and closely predicts the observed behaviour of a robot simulation. While the model is simple, it is very general, and these results suggest that it could be of practical as well as theoretical interest.

Chapter 7

Patch Switching

In the Chapter 6 we introduced an example for the WF-model and showed how a robot controller can be derived that lets the *Fungus Eater* make optimal task-switching decisions for this scenario. The decisions to be made were (1) when to stop working and obtain more energy and (2) how much energy to accumulate before returning to work. The assumption was that the gain rates are constant.

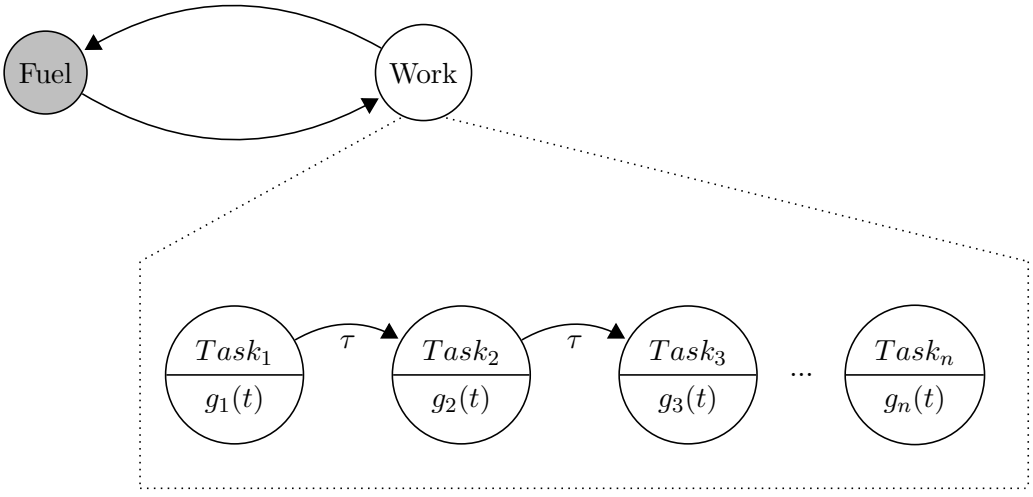


Figure 7.1: Patch Model: The work task consists of n task, each with its own gain function $g_i(t)$, possible subject to diminishing returns

In this chapter we investigate a different kind of decision. Given several resource locations, when should the *Fungus Eater* leave the current location and move to another

location ? Note it is not important whether this resource is related to work or energy, ore or fungi respectively in the *Fungus Eater* analogy. The problem we study is shown in the dashed box in Fig. 7.1 and can be applied to work or fuelling. For simplicity we will reference this problem from a work context, so the robot is purely concerned with work right now. Once the robot started to perform the first task, it has to decide at every time step whether to stay and keep performing the current task or to switch tasks and perform a new task. While performing a task the robot gains a reward according to a task specific gain function $g_i(t)$. The index i refers to the i-th task. The gain function can be dependent on the task execution time. Switching tasks comes at a cost that can be an actual decrease in reward or come in the form of lost opportunity, e.g. by travelling between job sites. In our case, travelling takes on average τ units of time. What makes this decision a difficult and interesting one is the fact that a large number of resources are subject to diminishing returns (see Sec. 7.1).

Our task is foraging for atomic units of resources that are consumed as they are encountered, instead of delivered centrally. A reward is instantly obtained once a resource is consumed. This models self-feeding, for example. Crucially, units of resource are not distributed uniformly, but exist in regions of locally high density known in the biology literature as *patches*. Danchin et al. (2008) defines a patch as "an homogeneous resource containing area (or part of habitat) separated from others by areas containing little or no resources". The advantage of patches is two-fold: (1) patches give the forager *a priori* information about the likelihood of finding resources. For example we expect to find valuable apples only on apple trees, and not on pine trees; (2) patches reduce the complexity of the decision process: instead of making decisions about each apple, it is sufficient to only make decisions about each patch. As the number of patches is usually significantly smaller than the number of resource units, we can expect reduced complexity of decision-making.

However, considering patches instead of atomic units of resource introduces the issue of the reward rate per patch, usually called *patch quality*. This may not be known *a priori*, and may require the forager to forage in the patch for some time to determine a good estimate of the patch quality. This sampling cost inevitably harms overall performance.

The task illustrated in Fig. 7.2 has the properties of interest. A robot collects apples in an orchard, and is rewarded as it collects each apple. As a tree (patch) is depleted of apples, the marginal cost of picking the next apple increases. In other words collecting apples is subject to diminishing returns. How does the robot decide to abandon the current tree and

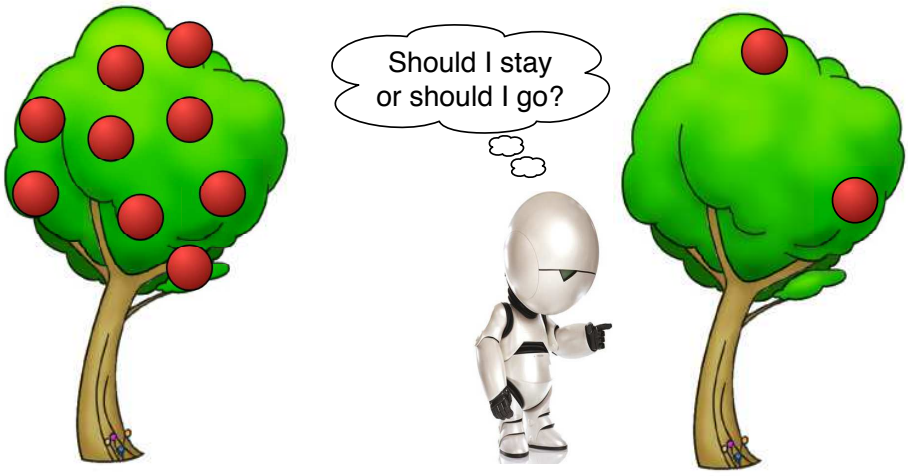


Figure 7.2: The robot must maximize global collection rate in an environment of multiple work sites, each with locally diminishing returns.

move to the next, such that the overall rate of apple collection is maximized?

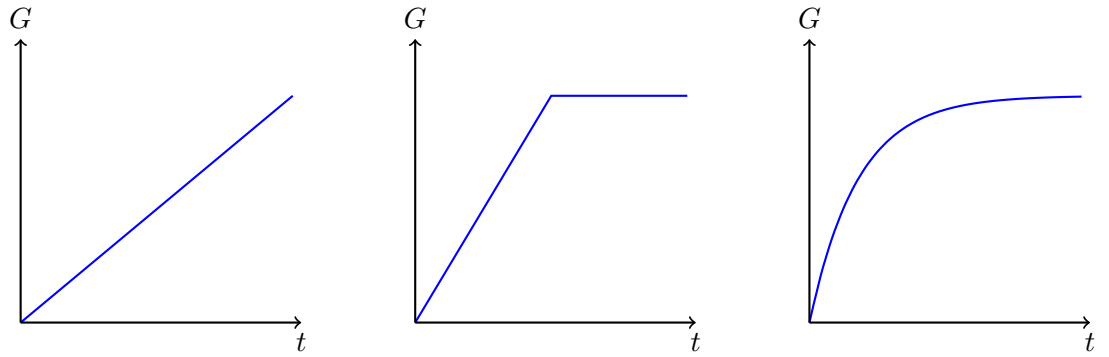
The problem we consider is related to foraging, which is well-studied in behavioural ecology (Stephens and Krebs; 1986; Stephens et al.; 2007) and is a canonical task in autonomous robotics (Cao et al.; 1997). This often means multi-agent and *central place foraging* (Stephens et al.; 2007), where foraged items are delivered to single privileged location. Other focal points of robotics foraging are to dynamically allocate the optimal number of workers to a given task (Liu et al.; 2007; Ulam and Balch; 2004; Wawerla and Vaughan; 2010) or method of reducing interference between foraging robots by separating them in space in order to improve the system's performance (Lein and Vaughan; 2008, 2009; Østergaard et al.; 2001; Shell and Matarić; 2006). For more details on robot foraging see Sec. 3.5.1.

Almost all previous work examines role allocation and interference reduction in collaborative multi-robot systems and very little work has been done on high-performance solitary foraging in robots, though this is well-studied in animals.

An exception, and the most similar previous work, is by Andrews et al. (2007) which explores the use of the Marginal-Value Theorem (see Section 7.2) for task switching. Andrews et al. (2007) investigates mathematical models of a robot. In contrast, this work and the experiments in this section are grounded in a complete low-level robot controller in a sensori motor simulation. Another important difference is that Andrews does not address the problem of how to measure the instantaneous gain rate in situation where rewards are

obtained in atomic units.

7.1 Diminishing Returns



(a) Constant gain function

(b) Gain function of a depleting patch, without patch depression

(c) Exponential gain function

Figure 7.3: Example gain functions

Diminishing returns refers to a decrease in gain rate with exploitation time. In this case the gain function $g_i(t)$ is negatively accelerated with task execution time t. Usually more work gives more reward, e.g. transporting pucks from a source to a sink (provided the source produces a sufficient number of pucks) yields a reward per puck transported, thus the more pucks we transport the higher our reward. The gain function of such a task is shown in Fig. 7.3a. If we drop the assumption that the source constantly produces pucks and instead let the source eventually run out of pucks, the reward from this task will reach an upper limit. Figure 7.3b shows a gain function for this case. Yet other tasks may become more and more difficult as we perform the task. For example picking apples from an apple tree is initially relatively easy, as long as the apple tree is full of apples. As the number of apples decreases it becomes more and more challenging to reach the remaining apples. Therefore the apple gain rate will decrease over time, as shown in Fig. 7.3c. This is characteristic of many robot tasks, such as mining, de-mining or foraging.

Causes for diminishing returns are:

- Finite resources: as pointed out early, as resources become sparser it often become increasingly difficult to obtain them.

- Robot environment interaction: the behaviour of the robot influences the shape of the gain function. Clearly the faster the robot works, the steeper the gain function. More interesting are effects that alter the gain function from linear to exponential (diminishing returns case). Assume an obstacle free office with uniformly distributed pucks. By systematically searching for pucks e.g. in a meander pattern, the robot gains pucks with an approximately constant rate. If the search is random on the other hand, the robot is likely to again search areas already cleared of pucks. As the number of cleared sub areas increases it become more and more likely for the robot to search those cleared spaces. Thus the gain rate deceases with work time. Figure 7.5 illustrates these effects based on a foraging simulation.

7.2 Marginal-Value Theorem

In behavioural ecology the task switching problem is often discussed in terms of optimal foraging theory (Stephens and Krebs; 1986) as a patch leaving decision. In this context patches are subject to diminishing returns and thus require the forager to make decisions about changing patches. In this case the task switching cost is analogous to the inter-patch travel cost. An important result of optimal foraging theory is the Marginal-Value Theorem (MVT). Charnov (Charnov; 1976b; Charnov and Orians; 1973) proposed the MVT to model foraging decisions made by animals.

To derive the MVT Charnov (1976b) argued that an optimal forager should maximize the longterm average gain rate, given by

$$R = \frac{\sum \lambda_j \cdot g_j(t_j) - \tau \cdot E}{\tau + \sum \lambda_j \cdot T_j} \tag{7.1}$$

where λ_j is the proportion of visited patches that are of type j, $g_j(t_j)$ is the net gain function for a patch of type j, τ is the average inter-patch travel time, E the rate of energy expended while switching patches and t_j is the time spent in a patch of type j. The objective of a forager is to select all patch residence times $\Gamma_j^* = t_j$ such that R is maximized.

Charnov shows that R is maximized if $\frac{\partial g_j(t_j)}{\partial t_j} = R$. Graphically this is easy to do with the help of a rather unusual graph. In Fig. 7.4 we plotted two quantities on the abscissa, the patch residence time increasing to the right and the inter-patch travel time increasing to the left. The optimal patch residence time Γ_j^* is found by constructing a tangent to the gain

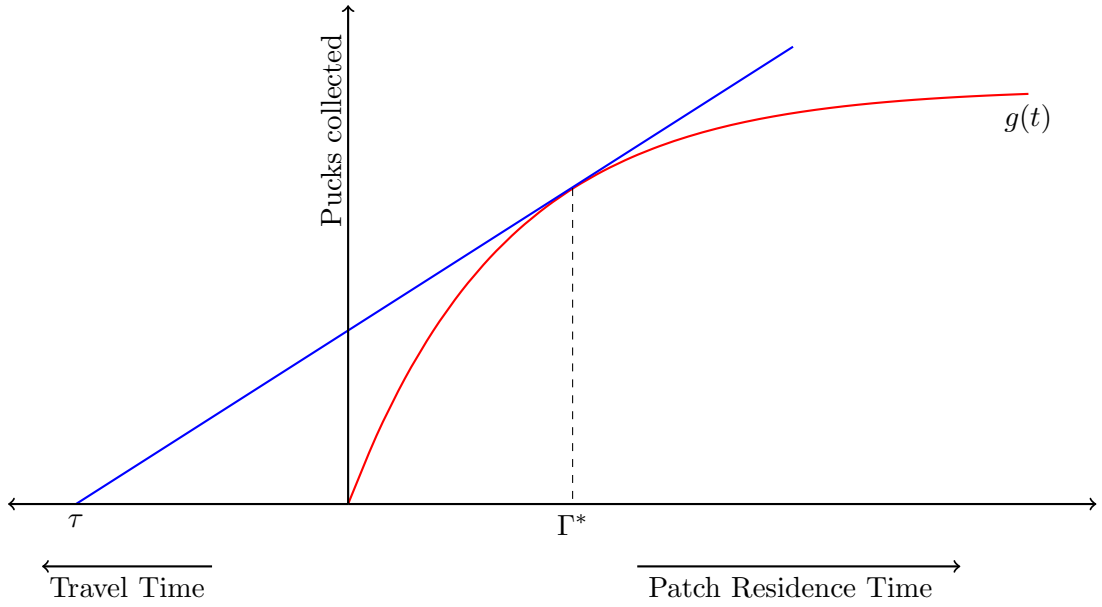


Figure 7.4: Typical MVT plot with two quantities on the abscissa: travel time increasing to the left, and patch residence time increasing to the right. The optimal patch residence time Γ^* is found by constructing a tangent to the gain function $g(t)$ that begins at the patch switching time τ on the travel time axis.

function that begins at the patch switching time τ on the travel time axis (see (Stephens and Krebs; 1986) for more details).

Charnov's key result is the following patch leaving rule: "*when the intake rate in any patch drops to the average rate for the habitat, the animal should move on to another patch*" (Charnov and Orians; 1973). This rule has two consequences: (1) the marginal rate of leaving must be the same for all patches in a habitat; and (2) as the cost to switch patches increases the forager should exploit each patch longer.

The simplicity of this rule makes it very appealing as a task-switching rule for robots, but the theorem and its validity has been widely and controversially discussed, for example by Green (1984); McNamara (1982); Stephens and Krebs (1986). Some of these issues make a direct implementation of the MVT as a robot task switching policy impossible. The main problems are:

- How to measure the marginal gain rate if the reward comes in discrete lumps. Andrews et al. (2007) suggest calculating the slope of the gain function between the last gain function change and the one two changes prior. In our tests (not shown) this method

proved insufficient due to the stochastic nature of puck encounter during random foraging in patches with randomly placed pucks. In one of the proposed control policies we use the expected value of a beta distribution over time-steps in which the robot found a puck and those in which it did not, as a proxy for the instantaneous rate. While we are able to build a task switching policy around this estimated gain rate, it is not the instantaneous gain rate. Thus leaving a patch once this estimated gain rate equals the long-term average rate does not maximize the long-term gain rate. We show a solution to this problem.

- The true long-term average gain rate for a given environment is usually unknown to the forager, all it can know is the average gain rate it experiences. But this experience is a result of the foragers behaviour, yet the MVT requires the forager to base it's patch leaving decision on the obtainable long-term average gain rate. This circular dependency necessitates that the forager explores the action-space in order to find the maximum long-term average gain rate. In the first control policy (see Sec. 7.3) we use the circular dependency and turn the foraging task into a multi-armed bandit problem. We use standard ϵ-greedy methods to tackle the exploration-exploitation trade-off.

Stephens and Krebs (1986) summarizes this as "*The MVT survives not as a rule for foragers to implement, but as a technique that finds the rate-maximizing rule from a known set of rules*". Behavioural ecologists proposed other patch-leaving rules. (1) **number rule**, "leave after catching n items" (Gibb; 1958); (2) **fixed residence time rule** "leave after being in a patch for t time" (Krebs; 1973); (3) **give up time rule** "leave after t time has elapsed since the last encounter" (Krebs et al.; 1974); (4) **rate rule** "leave when the instantaneous intake rate drops to a critical value r" (McNamara; 1982). Rules 1-3 have the advantage that the decision is based on values that are easily measurable by the forager. The rate rule is an extension of the MVT in that it copes with variance in patch sub-types, but it does not address the two issues mentioned above. All of these rules have in common that they do not address the question of how to obtain the magic number on which the decision is based.

Although our work is inspired by the behavioural ecology literature, we are not suggesting that the control policies or decision strategies presented in this chapter are a model of animal behaviour nor that animals employ any of these methods. We merely use the MVT

to develop a robot controller for the problem of task switching in situations of diminishing returns.

Next we introduce two different robot controllers for task/patch switching that are based on the MVT. The first controller explores the action-space in a greedy fashion. This controller was also described by Wawerla and Vaughan (2009). The second controller uses the ideas of the MVT more directly and makes task-switching decisions based on the derivative of the estimated long-term average gain rate.

7.3 Robot Controller: Action-Value

We use a generic mobile robot model in the well known simulator Stage (Vaughan; 2008). It is equipped with a short-range colour blob tracker to sense pucks, our unit of resource. The robot knows (or equivalently can detect) the boundaries of puck patches. Patches are 620 times the size of the robot, and contain 10, 30 or 50 pucks placed uniform randomly. A minimum distance between pucks is enforced to avoid overlap. To exploit a patch, the foraging robot can use one of two foraging policies:

1. Under the **random foraging policy** the robot drives straight until it comes to the patch boundary, where it chooses a new heading that brings it back into the patch, at random. When pucks are detected, the robot servos towards the closest puck and collects it.

2. Under the **systematic foraging policy** the robot employs a regular square-wave-like search pattern from one side of the patch to the other side. The distance between legs in the pattern is small enough to guarantee that the whole patch is searched.

These policies exhibit different reward dynamics. To illustrate this we tested each policy on 100 patches of 50 uniform random placed pucks each. Fig. 7.5 shows the average time required to collect a certain number of pucks under the two policies. The random foraging policy suffers from diminishing returns, since over time it is increasingly more likely to revisit previously cleared, now unproductive, parts of the patch. The systematic forager has an approximately constant collection rate. While we might prefer the systematic forager because of this property, it may be much more costly to implement than the random forager. Real-world low-cost robots like the iRobot Roomba use a randomized method that suffers from diminishing returns.

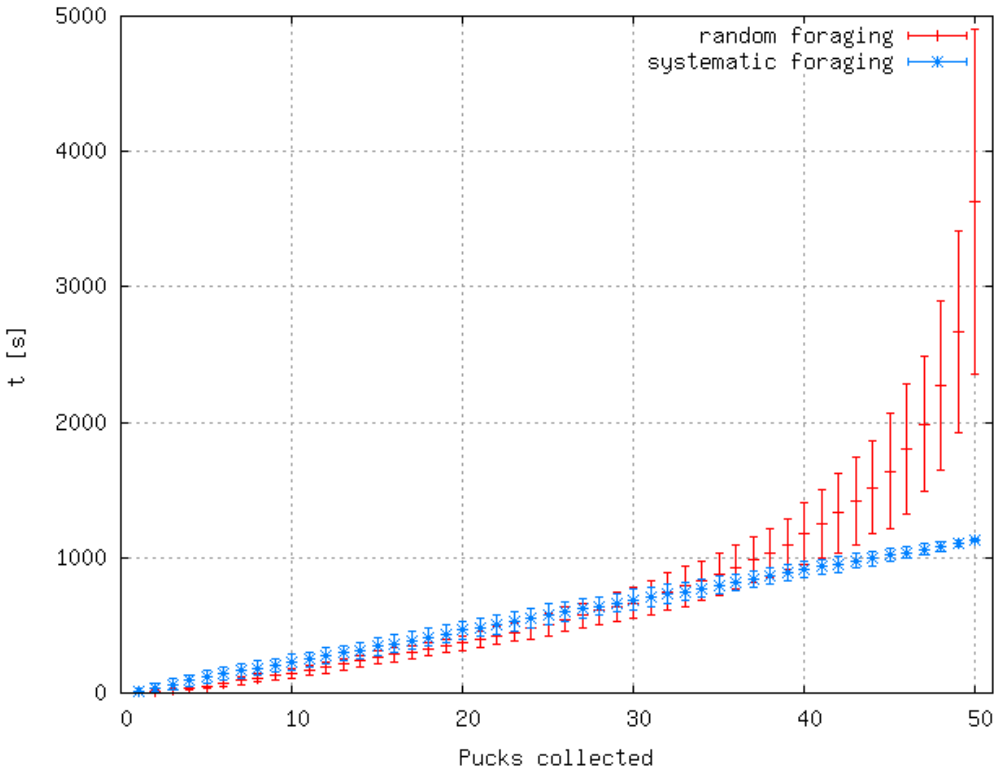


Figure 7.5: Time required to collect all 50 pucks in a patch using random search and systematic search.

The point here is not to choose the best policy, but to illustrate that low-level robot control policy can fundamentally influence the nature of the overall optimization task, and to show that single-robot random foraging is subject to diminishing returns (see Sec. 7.1) and thus provides a suitable model for all tasks of this domain.

Given this randomized foraging policy, the robot needs to determine when to abandon the current patch and move on. As suggested by Charnov's MVT (Charnov; 1976b), our robot abandons a patch once the instantaneous gain rate drops below a threshold. To do this we must (1) determine the instantaneous gain rate of atomic items (pucks) encountered at every time-step and (2) select a leaving threshold that maximizes the long-term average gain rate.

7.3.1 Instantaneous Gain Rate

Measuring the instantaneous rate of atomic events is impossible. Andrews et al. (2007) suggest calculating the slope using the current and the previous two gain function values. This approach may work for continuous gain function, but fails in situations where the gain function is non-continuous and stochastic such as in unit-based foraging in uniform random patches. So we resort to an alternative representation. We use the expected value of a beta distribution over time-steps in which the robot found a puck and those in which it did not. Equation 7.2 gives this expected value of the beta distribution which we use as a proxy for the instantaneous rate $\hat{\lambda}_i(t)$.

$$\hat{\lambda}_i(t) = \frac{p_i(t)}{p_i(t) + q_i(t)} \tag{7.2}$$

Where i refers to the i-th patch, t is the amount of time spent in a patch, $p_i(t)$ is the number of time-steps in which the robot collected a puck and $q_i(t)$ is the number of time-steps during which the robot did not collect any pucks. This method has an interesting problem: until the first puck is found, the number of time-steps in which the robot collected pucks is zero ($p_i = 0$), and thus our estimate of the rate is zero ($\lambda_i = 0$). Hence the robot will immediately leave the patch. We avoid this by initializing $p_i(0)$ and $q_i(0)$ with a prior ϕ_p and ϕ_q respectively. These priors represent the robot's expectation of the initial gain rate of a patch. They can either be set to an environmental constant (if known) or they can be determined at run time by experience from the previous patch. The ratio of the priors represents the expected initial gain rate, the magnitude of each is an indication of the forager's confidence in these priors. The higher the magnitude the more experience is required to modify λ, i.e. change the robot's belief about the prevailing rate.

Because puck encounter is a random and infrequent process, the value of $\hat{\lambda}_i(t)$ is very variable in practise, particularly for small values of $p_i(t)$ and $q_i(t)$. We apply a low-pass filter to smooth this slightly. These steps are combined into Algorithm 7.1.

7.3.2 Patch-Leaving Threshold

Recall that the MVT predicts that a forager should leave a patch once λ drops to the environment's global average. In the general case it is impossible for the robot to know this true global average rate. All the robot can measure directly is the long-term average gain rate it experiences: a value which depends on the robot's past behaviour. The experienced

```
1  Algorithm:forage(θ)
2  init  φ_p, φ_q
3  t = 0
4  p(0) = φ_p
5  q(0) = φ_q
6  λ̂_filt(0) = p(0)/(p(0)+q(0))
7  repeat
8      t = t + 1
9      randomly forage for one time-step
10     if puck found then
11         p(t) = p(t − 1) + 1
12     else
13         q(t) = q(t − 1) + 1
14     end
15     λ̂(t) = p(t)/(p(t)+q(t))
16     λ̂_filt(t) = λ̂_filt(t − 1) + k_1(λ̂(t) − λ̂_filt(t − 1))
17 until λ̂_filt(t) < θ
18 return (t, p)
```

Algorithm 7.1: Forage in the current patch until a proxy for the instantaneous rate $\lambda(t)$ drops below the threshold θ

average gain rate $\mu(\theta)$ for foraging in n patches is given by

$$\mu(\theta) = \frac{\sum_{i=1}^{n} g_i(\theta)}{\sum_{i=1}^{n}(t_i + \tau_i)} \tag{7.3}$$

where $g_i(\theta)$ is the gain function that gives the total number of pucks collected in patch i when the patch is abandoned according to Algorithm 7.1, t_i is the time spent in the i-th patch, and τ_i is the patch switching duration. The objective of the forager is to maximize μ by selecting θ.

$$\theta^* = \text{argmax}(\mu(\theta)) \tag{7.4}$$

Unfortunately $g_i(\theta)$ is unknown and difficult to obtain. The function depends precisely on the robot's sensori motor interaction with the environment, the patch quality and the distribution of pucks in the patch. Hence a closed–form solution of eq. 7.4 is not obtainable. Fig. **??** shows the long-term average gain rates over a range of values of θ for different environmental conditions. From these graphs we can see that the observed average gain rate varies widely under different circumstances. The variance in average gain rate observed

is high, but error bars are omitted for clarity.

Since we only have an approximation of the instantaneous gain rate, we cannot, as the MVT requires, leave a patch once this rate drops to the long-term average. Instead we resort to maximizing the long-term gain rate by recursively improving the leave threshold θ based on previous experience.

In practise we found that the observations have such high variance (evident in the error-bars of Fig. 7.5), that local gradients are not very useful and thus simple gradient descent methods resulted in poor performance. A more sophisticated heuristic approach is required in our scenario.

Action-Value methods have been shown to be effective for n-armed bandit problems; maximizing the return of a discrete set of unknown process. Sutton and Barto (1998) give an overview of various action-value methods and Vermorel and Mohri (2005) give an empirical comparison. We can adapt this framework to our continuous action-space by finding a discrete set of values of θ that approximate the input-output mapping of the true gain function. We continuously refine this set by exploring θ space, while frequently using the current best estimate of θ^* to obtain good performance as we go.

Since no generally optimal method is known, we use here a combination of *softmax* and *ε-first* adapted for continuous action-spaces. (See (Sutton and Barto; 1998) for details on these methods). A comparative analysis of the various possible algorithms is beyond the scope of this document. Our objective is it to demonstrate that it is possible to develop a robot controller based on the underlying principals of the MVT. The complete method is shown as Algorithm box 7.2. The parameters θ_{min} and θ_{max} are the minimum and maximum patch leaving thresholds. These are determined by the environment and the robot specification, e.g. the maximum travel speed. N denotes the number of bins into which the action-space is discretized. More bins potentially allow a better approximation, at the expense of longer start-up time. Constant k_1 is used to smooth the weight update, k_2 is the *temperature* for *softmax* action selection and k_3 determines how much we are willing to modify the leave rate threshold between adjacent patches.

To initialize our discrete approximation of the gain function we must fill the N bins. They can be initialized with prior expected values if available, but here we obtain them empirically with a start-up phase. For the first N patches visited we use the discretized leave rate threshold (lines 5-7 in *ε-first*). Here $1 \leq a \leq N$ denotes the bin number. $\Theta(a)$ describes the centre of bin a and $w(a)$ its weight. For the remaining patches we select a

```
1  Algorithm:adaptive()
2  init θ_min,θ_max, N, k_1, k_2, k_3
3  δ = (θ_max − θ_min)/N
4  for i=1 to ∞ do
5      if i ≤ N then
6          a = i
7          θ = i · δ
8      else
9          draw a with probability ∝ e^{w(a)/k_2} / Σ_{j=1}^{N} e^{w(j)/k_2}
10         draw θ uniform randomly from [Θ(a) − k_3δ, Θ(a) + k_3δ]
11     end
12     (t, p) = forage(θ)
13     μ = p/(t+τ)
14     if i ≤ N then
15         Θ(a) = θ
16         w(a) = μ
17     else
18         Θ(a) = (w(a)·Θ(a) + μ·θ)/(w(a) + μ(a))
19         w(a) = w(a) + k_1(μ − w(a))
20     end
21 end
```

Algorithm 7.2: Adaptive leave rate selection

bin using *softmax* (line 9). The leave rate threshold θ is uniform randomly chosen from an interval centred around the centre of the bin (line 10).

The robot now forages with the new leave rate threshold θ using Algorithm 7.1. This results in some patch residence time t and in p pucks collected. Therefore we can calculate the average gain rate for the patch μ as the number of pucks collected divided by the sum of patch residence time and patch switching time τ (line 13). Next we update the weights and the centre of the bin. For the first N patches the weight is simply the average gain rate μ. The centre of the bin is the leave rate threshold used (lines 15-16). For the remaining patches the centre of the weight is the moving average of the average gain rate and the centre of the bin is the weighted sum of the previous centre and the just explored leave rate threshold (lines 18-19), where the previous centre is weighted by the weight of the bin and the latest leave rate threshold is weighted with the average gain rate that resulted from the

use of this threshold.

Summarizing the action of Algorithm 7.2: *Softmax* selects the best bin in a greedy fashion while it keeps exploring the other bins with probability proportional to the weight of each bin. Since the weight is an estimate of the expected yield rate, it explores higher paying areas of the action-space more frequently then lower paying areas. The discretization step might choose bin centres that are suboptimal; to over come this problem we perform a local random walk of the bin centres. Over time this will move the bin centres towards better values.

7.3.3 Experiments

To investigate the effectiveness of this approach, we conducted a series of simulation experiments consisting of two phases (1) generate foraging data and (2) test our adaptive task (patch) switching policy on the generated data.

To generate the foraging data we used the method described in Section 7.3. For each of the three puck qualities (10, 30, 50 pucks per patch) we generated 100 samples and recorded for each time-step whether the robot collected a puck or not. Note that in this phase no patch-leaving decisions are made, we just recorded data while the robot simply collected pucks until each patch was exhausted.

In the second phase we ran our adaptive patch switching policy on the recorded data and compared it against the best solution obtained by exhaustive search over the same data set. Decoupling the data generation from the analysis of the control policy allowed us to cancel any noise in the performance analysis caused by the random generation of the patches. It also made brute-force search for approximately optimal solutions feasible.

To obtain a benchmark with which to compare our online adaptive method, we choose a finite discrete set of rate thresholds over a range of θ between our preset maximum and minimum. For each threshold we had the robot forage in each patch until the instantaneous rate dropped to that selected leave rate threshold (Alg. 7.1). On leaving the patch, the robot takes some time in which no pucks are collected before arriving at the next patch: the patch–switching cost. This way we found the leave rate that maximizes the long-term average gain rate for the recorded data. Fig. **??** shows the long-term average gain rate over the leave rate threshold for patches of different quality and different switching costs. The prediction of the MVT that the patch residence time should increase with increasing switching cost is clearly visible. The graphs also give an idea of the search space for the

adaptive algorithm. Especially low switching costs (fig. 7.6a) exhibit a sharp performance peak which is difficult to adapt to, in this situation it is desirable to stay on the side with the smaller slope.

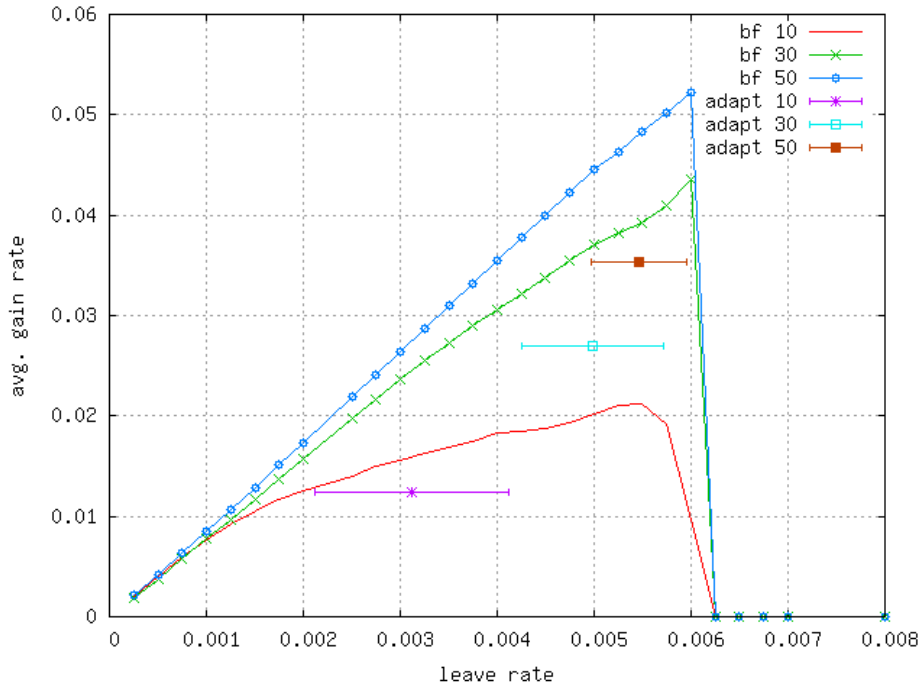
Patch quality is one factor that determines the environmental gain rate. The other is the average switching time between patches. To investigate the influence of the switching time τ we analyzed the system with 4 different switching times, $\tau = \{10, 100, 500, 1000\}$ seconds.

The performance of the adaptive method is presented as mean and standard deviation of observed gain of 20 trials of 100 patches each, compared to the estimated optimum obtained by exhaustive search. All algorithm parameters required were set manually and kept constant without attempting to optimize them, with one exception. The confidence of the priors for the instantaneous gain rate were lowered for patches with 10 initial pucks, so that the estimate of the gain rate would change faster. The original values gave poor performance in this challenging scenario.

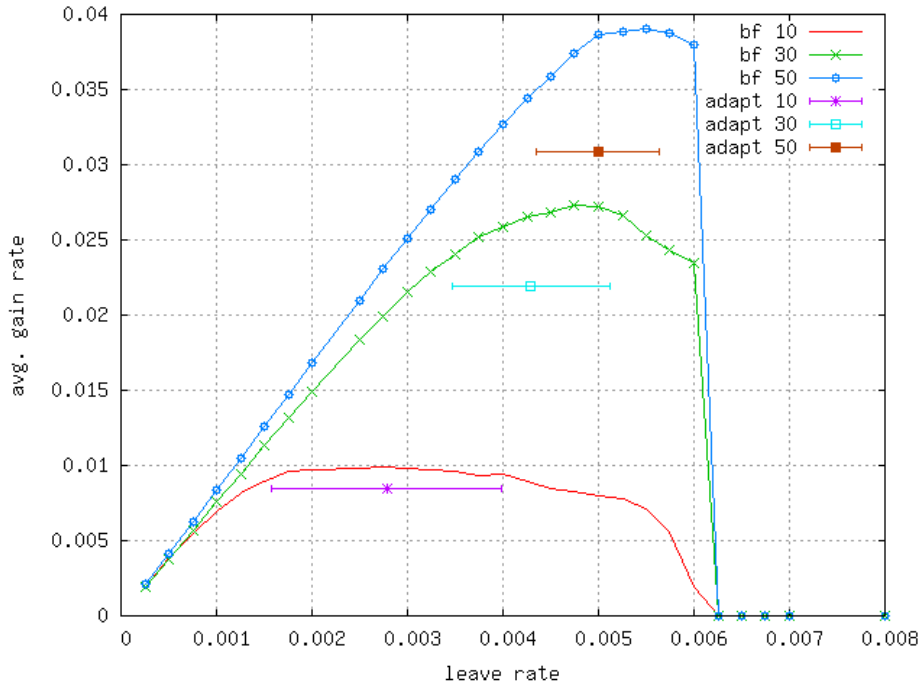
	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	60.2	3.72	84.6	2.68	76.7	3.40	66.6	3.89
30	62.0	0.76	79.9	2.20	82.8	2.75	81.4	2.03
50	68.3	0.64	79.3	1.11	85.4	1.60	86.3	1.38

Table 7.1: Adaptive patch switching and random foraging

Table 7.1 shows the results of the first experiment in which we analyzed the adaptive patch switching policy for 3 different patch qualities and 4 different switching costs. The results are given in percentage of the bruteforce long-term gain rate. The small standard deviation of about 2.5% indicates that the system performs fairly consistently despite the high noise levels in the data. Compared to the bruteforce method we expect a lower performance since the system, like any ϵ-greedy method, has to trade off between exploration and exploitation. Our method performs a fixed number ($N = 10$) of initial exploration trials and subsequently performs continuous exploration with a small probability. Especially during the initial exploration we expect some trials with very poor performance because the entire action-space is sampled. This does harm the overall performance.

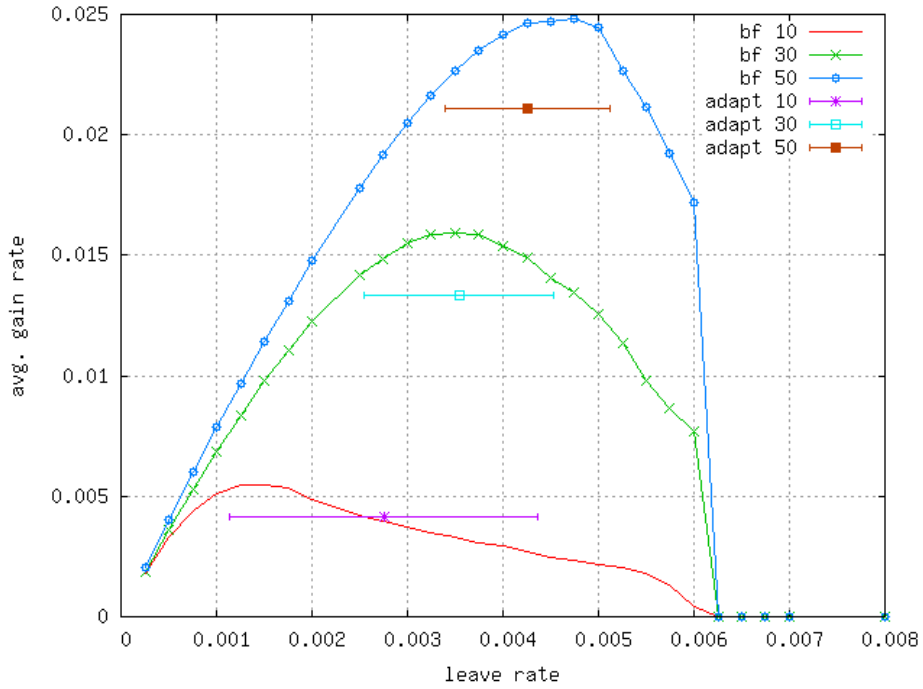


(a) Average gain rate with patch switching time 10 sec.

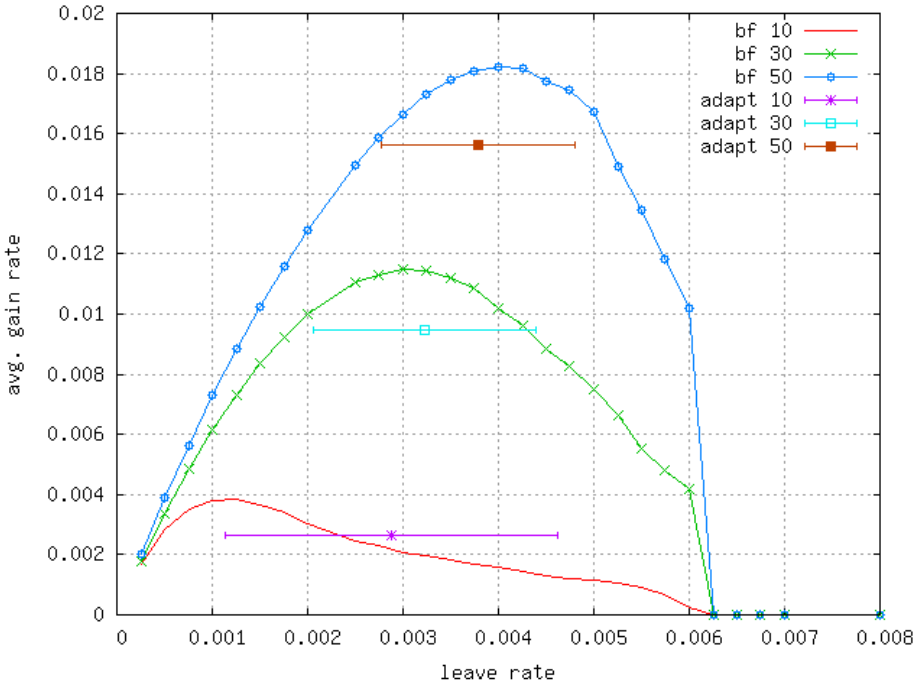


(b) Average gain rate with patch switching time 100 sec.

Figure 7.6: Long-term average gain rate observed versus leave rate threshold for different patch switching cost and patch qualities (errorbars in y-axis omitted to improve readability). The sharp drop-off at $\tau = 0.006$ occurs because the leave threshold is too high for the robot-environment interaction, the robot leaves a patch before it encounters the first puck



(c) Average gain rate with patch switching time 500 sec.



(d) Average gain rate with patch switching time 1000 sec.

The actual performance appears to be quite good: between 75 and 85% of the brute-force benchmark solution as long as the patch switching cost is not too small. The performance drops with the very small patch switching cost of 10 seconds. As Fig. 7.6a indicates, this is a challenging situation since the gain rate function has a very sharp peak. Another reason why this is a difficult situation is that the small switching cost requires the forager to leave the patch very early. For example, the best threshold found by brute-force causes the robot to stay for only an average of 35 seconds in the 10 puck patch. Any error the adaptive method makes is relatively large compared to the small switching cost.

Nevertheless as Fig. **??** shows, the mean of the leaving threshold selected by the adaptive system is usually near the optimal threshold and always on the side of caution by being on the side of the smaller slope.

	switching cost variance							
	0		30		50		80	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	84.6	2.68	82.6	3.30	79.5	4.79	80.0	2.97
30	79.9	2.20	80.0	1.48	79.5	2.55	80.8	3.01
50	79.3	1.11	79.5	1.16	79.9	0.90	80.9	1.18

Table 7.2: Adaptive patch switching and random foraging under varying patch switching times with mean 100 sec

In a second experiment we investigated the performance of the system when the patch switching time varies from one patch to the next. For this experiment we drew patch switching times from a normal distribution with a mean of 100 seconds and variance of 30, 50 and 80 seconds respectively. As in the previous experiment we obtain a near-optimal threshold by brute-force search as a benchmark and compare it with the average of 20 instances of the adaptive system. Table 7.2 shows the results. The performance is around 80% for all situations with low deviation, indicating that the system is not sensitive to variance in the switching cost.

To analyze the system under changing patch quality conditions we set up two simulations in which patch quality changes from 30 pucks per patch for the first 100 patches, to 50 pucks per patch for another 100 patches and vice versa. We ran these simulations with a patch switching cost of 10, 100, 500 and 1000 seconds respectively. We compare the performance

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
30→50	82.0	2.87	92.8	3.02	90.9	2.92	90.8	1.67
50→30	80.0	2.44	90.7	2.44	91.3	1.50	89.3	1.50

Table 7.3: Adaptive patch switching for random foraging under changing patch quality

of our method with that observed when switching between the best fixed thresholds for 30 and 50 pucks obtained by brute force search. The results are shown in table 7.3. The data suggest that the system handles the patch quality switching well. The performance seems slightly better than in the stationary situation in table 7.1. This is probably due to a relatively longer exploitation duration: the cost of the initial 10 patch exploration is amortized over 190 patches rather than 90. Despite that fact that the bruteforce method is given the advantage of actually knowing when the patch quality switch occurs, something our method has to detect and adapt to, the system performs very well.

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	55.2	4.54	81.2	6.32	84.4	6.50	83.4	7.03
30	80.7	3.46	80.5	3.76	79.1	3.89	79.2	2.05
50	76.5	1.80	80.1	1.49	84.9	2.64	83.7	2.79

Table 7.4: Adaptive patch switching for systematic foraging

In a last experiment we investigated the adaptive method's performance in situation in which the forager searches each patch systematically. Recall that under this foraging policy, the instantaneous gain rate is constant until the patch is empty and the gain rate drops to zero. In this situation, the robot has to decided not which leave rate maximizes the overall reward, but only to determine when the patch is empty. To see how our system copes with this situation we compared the brute-force and the adaptive method under a systematic foraging policy. As table 7.4 shows the system can also handle situations in with the instantaneous gain rate is given by a step-function.

7.4 Robot Controller: Marginal Gain Rate Task Switching

The task-switching policy from the previous section required the robot to perform exploration steps in order to evaluate the average quality of the available tasks. We showed that the performance of this policy is about 80% of that of a bruteforce near-optimal method. In this section we propose a recursive task switching policy based on locally available information only, hence no explicit exploration phase and thus no exploration/exploitation trade-off is required.

In Sec. 7.2 we introduced Charnov's patch model and outlined his argument for why a forager should maximize the longterm average gain rate. For convenience we restate Eq. 7.1:

$$R = \frac{\sum \lambda_j \cdot g_j(t_j) - \tau \cdot E}{\tau + \sum \lambda_j \cdot T_j} \tag{7.5}$$

where λ_j is the proportion of visited patches that are of type j, $g_j(t_j)$ is the net gain function for a patch of type j, τ is the average inter-patch travel time, E the rate of energy expended while switching patches and t_j is the time spent in a patch of type j. The objective of a forager is to select all patch residence times t_j such that R is maximized.

Without loss of generality we ignore the energetic cost of travel $\tau \cdot E$, since it is independent of the decision variables, so Eq. 7.1 reduces to

$$R = \frac{\sum \lambda_j \cdot g_j(t_j)}{\tau + \sum \lambda_i \cdot T_j} \tag{7.6}$$

The gain function $g(t)$ depends on (1) the actual patch quality, which varies from patch type to patch type but can also be variable within a patch type, e.g. if the pucks are placed randomly and (2) on the robot environment interaction, e.g. sensor range, search strategy, motor control etc. Thus foraging in two equally sized patches, initially containing the same number of pucks, may result in two very different gain functions and there is no way a forager can predict the gain function of a particular patch. Figure 7.6 shows two exemplar gain functions and the average gain function over 100 patches (patch with initially 50 pucks). One can see how quite different the gain functions can be. Thus as McNamara (1982) argues, the sub-patch type variance has to be considered. This immediately raises the question how does the forager determine the type of patch she is currently foraging in? In some scenarios the patch type might be detectable by an external cue, e.g. large apple trees vs. small apple trees. But in general this is not possible and the forager is required

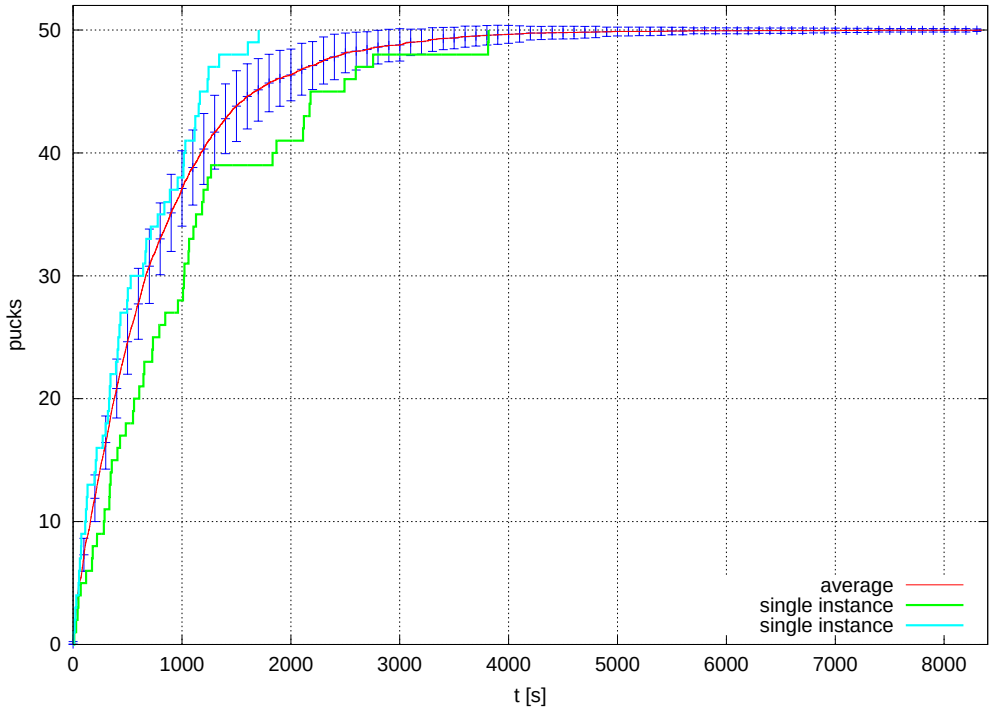


Figure 7.6: Average gain function (thin line) for random foraging in a 50 puck patch, error bars depict the standard deviation. Two instance of the gain function (thick lines) for patches with the same initial number of pucks.

to forage in the patch in order to obtain information about the patch. This adds a patch discrimination problem to the decision process.

To overcome these issues, we suggest dropping the notion of patch types and treating each patch as its own type. (In the following we still use the phrase "patch type" to mean patches with the same initial number of pucks, but we do not perform rate maximization based on the notion of patch types.) For unique patches the long-term average gain rate is

$$R = \frac{\frac{1}{n}\sum_i^n g_i(t_i)}{\tau + \frac{1}{n}\sum_i^n t_i} \tag{7.7}$$

We replaced the patch type index j with index i referring to unique patches. The advantage of not having to distinguish patch types and not having to deal with patch subtype variance comes at the disadvantage of having a possibly very large planning horizon n. In the original MVT equation we had to optimize over the set of possible patch types, now we optimize over all patches. Since the robot cannot predict what patches it will encounter in the future,

we avoid the large planning horizon by recursively maximizing Eq. 7.7 based on only past experiences and ignoring possible future changes. Then our approximation of the long-term average gain rate at patch i is

$$\tilde{R}_i = \frac{g_i(t_i) + G_i}{t_i + \tau + T_i} \tag{7.8}$$

Where G_i is the sum of collected pucks and T_i the total time (patch residence plus travel time) from all previous patches $0..i-1$. G_0 and T_0 can be used as prior that provides the robot with an initial estimate of the average patch quality. Both G_i and T_i are a simple statistic of the average patch quality of the environment. This information (except the prior) is gained by the forager during exploitation. Hence a forager encountering only one patch type will actually maximize Eq. 7.6. But a forager first encountering a series of only low quality patches and then a series of high quality patches will maximize a very different average gain rate function then an omniscient forager. But an uninformed forager maximizing Eq. 7.8, will do as well as possible given the limited available information.

7.4.1 Control Algorithm

The core of our task switching method is to maximize Eq. 7.8. This is done by numerically calculating the derivative of R_i at every time step and leaving the patch once the the derivative becomes zero. Since the gain function is assumed to be negatively accelerated, a maximum is found this way.

```
1  Algorithm:patchMax
2  init G_0, T_0, τ̃, k_1, k_2, k_3, k_4
3  i = 1
4  forall patches do
5      enter patch i
6      t = 0
7      g(0) = 0
8      repeat
9          t = t + 1
10         randomly forage for one time-step
11         if puck collected then
12             g(t) = g(t − 1) + 1
13         else
14             g(t) = g(t − 1)
15         end
16         r(t) = (g(t) + G_i) / (t + τ̃ + T_i)
17         if t == 1 then
18             r_filt(t) = r(t)
19         else
20             r_filt(t) = (1 − k_3) r_filt(t − 1) + k_3 r(t)
21         end
22         if r_filt(t) − r_filt(t − 1) ≤ 0 then
23             c = c + 1
24         else
25             c = 0
26         end
27     until c > k_1 and t > k_2
28     move to next patch in τ_i time
29     G_{i+1} = G_i + g(t)
30     T_{i+1} = T_i + t + τ_i
31     τ̃ = τ̃ + k_4 (τ_i − τ̃)
32     i = i + 1
33 end
```

Algorithm 7.3: Task switching algorithm

Algorithm 7.3 summarizes our task switching method. The robot forages for one time-step and if it collected a puck, the local gain function $g(t)$ is incremented (line 10-15). Next we calculate an approximation of the long-term gain rate based on the experience from previous patches (G_i, T_i), an estimate of the travel time $\tilde{\tau}$, and the value of local gain function at the current time. Because of the stochastic and noisy nature of the gain function, the estimate of the long-term gain rate has to be smoothed. In our implementation we use a low-pass filter (line 17-21). Other methods maybe substituted, however, it performs well enough for our purpose. As mentioned earlier the patch leaving decision is based on checking if the derivative of the long-term gain rate is equal to zero. Again because of the stochasticity of the gain function we might experience local maxima. A simple counting step helps to overcome those undesired local maxima (line 22-27). As with the low-pass filter, any suitable method may substituted. The actual patch leaving decision is made in line 27. A patch is left once a maximum is found and a minimum amount of time has been spent in the patch. This minimum patch residence time is helpful during the initial time in a patch. Until the first puck is found the current value of the gain function is zero $g(t) = 0$, this would cause the robot to leave the patch immediately.

Once the robot leaves the patch it travels to the next patch. This travel takes τ_i time. Before starting to forage in the new patch the estimates for the environment quality G and T and the estimate of the switching time $\tilde{\tau}$ are updated (line 29-30).

7.4.2 Experiments

To investigate the effectiveness of our approach, we conducted a series of simulation experiments consisting of two phases (1) generate foraging data and (2) test our task (patch) switching policy on the generated data. To generate the foraging data we used a generic mobile robot model in the well known simulator Stage (Vaughan; 2008). The robot is equipped with a short-range colour blob tracker to sense pucks, our unit of resources, in its vicinity. The robot knows (or equivalently can detect) the boundaries of a puck patch. Patches are 620 times the size of the robot, and contain initially 10, 30, 50, 100, 200 or 300 uniform randomly placed pucks. A minimum distance between pucks is enforced to avoid overlap. To exploit a patch, the robot randomly forages for pucks, by driving straight until it comes to the patch boundary, where it chooses a new heading that brings it back into the patch, at random. When a puck is detected, the robot servos towards the closest puck and collects it. Collecting a puck takes one simulation time step, so there is virtually no handling time.

At each simulation time step we record how many pucks the robot has collected so far in the current patch: this is the gain function.

As mentioned earlier the gain function is not only dependent on the initial number of pucks per patch but also on the robot/environment interaction. To get a good sample of the distribution of gain functions, we randomly generate 100 patches of each of the six patch types and record the gain functions from the robot foraging in those patches. Note that at this point in the experiment no patch leaving decisions are made. The robot simply forages until the patch is exhausted and the simulation is terminated. Testing our approach on this recorded data set rather than during the robot simulation allows us to compare approaches on exactly the same data and it makes it feasible to determine a near-optimal solution by bruteforce solution search.

As a baseline for comparison we need to find a t_i for each patch such that the long-term gain rate is maximized. No closed form solution is known to this problem, and the gain functions are available as data points only. So we employ a bruteforce search. Since each patch is unique this technically requires us to solve Eq. 7.7 for all possible combinations of patch residence times. Because this is computationally prohibitive we resort to calculating the average gain function over all 100 instances of a patch type. Then we find the best patch residence time by solving Eq. 7.7 for all possible t ($0 \leq t \leq T_{patch\_exhaused}$) and selecting the t that maximizes the average gain rate. In case of multiple patch types we calculate the long-term gain rate for each combination of residence times on the average gain function. This is only feasible since the number of patch types considered is small.

	initial pucks per patch					
	10	30	50	100	200	300
μ [s]	1858.1	2909.9	3631.8	4556.0	5171.0	5475.5
σ	825.2	1184.3	1271.3	1337.0	1206.0	1208.7

Table 7.5: Mean and standard deviation of the time required to exhaustively forage patches

In all of the following experiments we used the obtained long-term average gain rate as a metric for comparison. All algorithm parameters required were set manually and kept constant without any attempt to optimize them. The priors G_0 and T_0 were set to zero. To investigate our task switching method under a wide range of conditions we altered the task (patch) switching time τ from very short 10 seconds to very long 5×10^6 seconds (≈6 days). To put this in perspective we report the mean and standard deviation of observed times

required to exhaustively forage patches in Table 7.5. The spectrum reaches from almost no switching cost to a switching cost about 200 times the average time required to exhaust a patch.

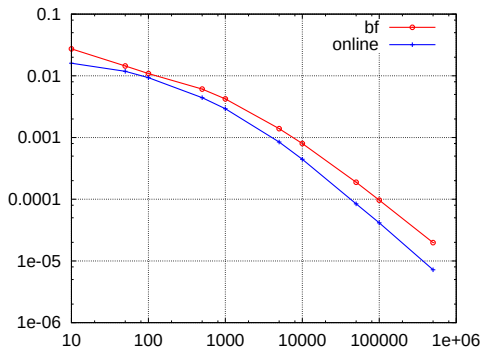
Single Patch Type

In a first experiment we had the robot forage in a series of 100 patches with the same initial number of pucks. Figure **??**(a)-(f) shows the achieved long-term average gain rate for each patch type over a variety of switching times compared to the bruteforce solution. From the graphs we can draw three conclusions. (1) If the task switching times are short (ie. much lower then the patch residence times) the performance of our method is in general lower then that of the near-optimal bruteforce method. The MVT predicts short patch residence times in situations where patch switching is cheap. But because of the various filters (filter parameters kept constant for all experiments over all conditions) our method's responsiveness is too slow in these short residence time situations. We say the performance is lower, but it is still above 78% (except in the 10 puck patches, where the performance drops to 50%). (2) Under low patch quality situations (10 pucks, 30 pucks) our method performs noticeably poorer than the bruteforce method. Again the reason is in the choice of parameters. The filters are too slow for the optimal, short patch residence time. (3) The method described in this paper achieves similar long-term rates as the bruteforce method in all other cases examined. Recall that it uses only locally obtained information, in contrast to the omniscient bruteforce method.

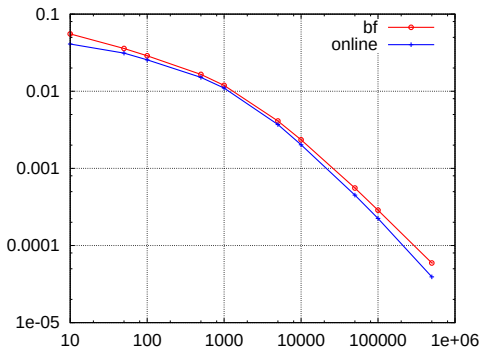
Multiple Patch Types

A more challenging problem is the case where patches of very different quality are encountered. As the MVT predicts the patch leaving decision is not only dependent on the quality of a given patch but on the global quality. In other words a patch residence time that is optimal under a single patch quality condition is no longer optimal for that patch type if patches of different type are encountered.

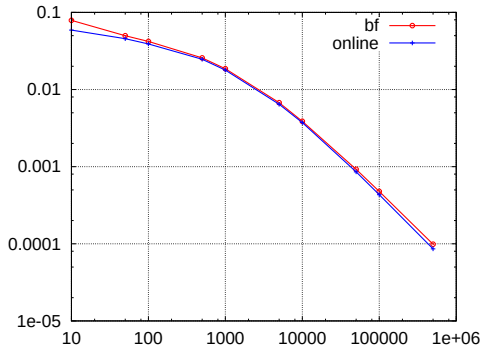
To investigate our system under these conditions we conducted a series of experiments. First we had the robot encounter 100 patches of type A and 100 patches of type B in a random order. Figure 7.7g and 7.7h show the averaged results over 20 trials for patch configurations 50:100 pucks and 50:300 pucks respectively. Errorbars were omitted because



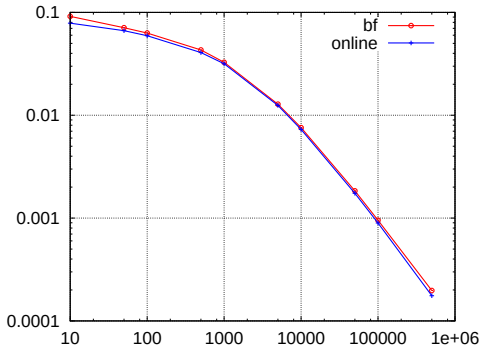
(a) Patch type 10 pucks



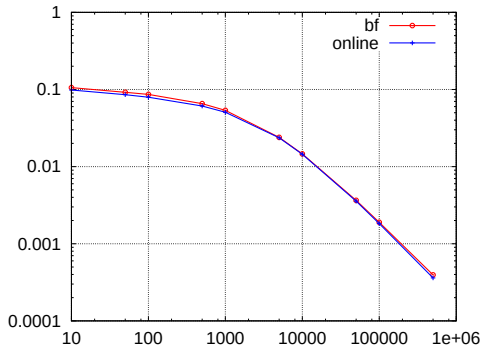
(b) Patch type 30 pucks



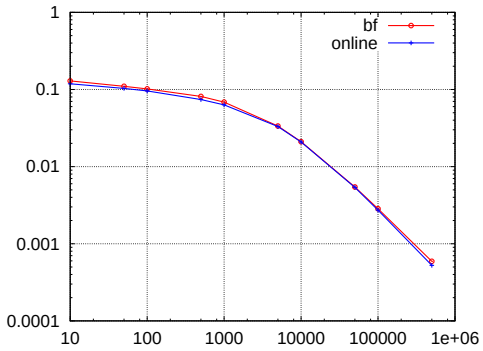
(c) Patch type 50 pucks



(d) Patch type 100 pucks

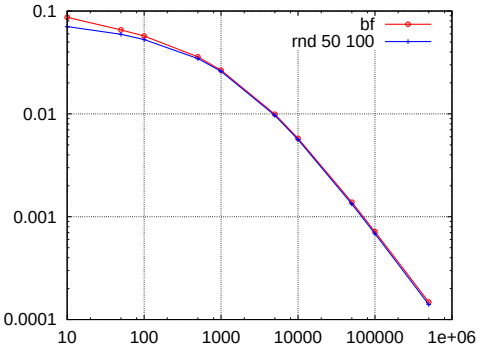


(e) Patch type 200 pucks

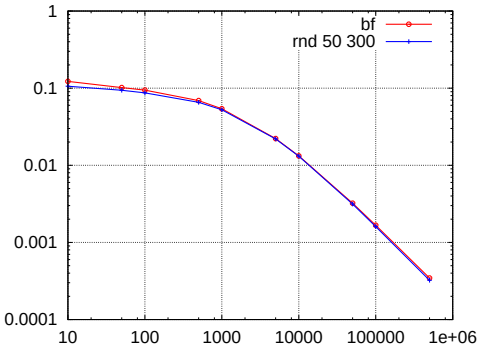


(f) Patch type 300 pucks

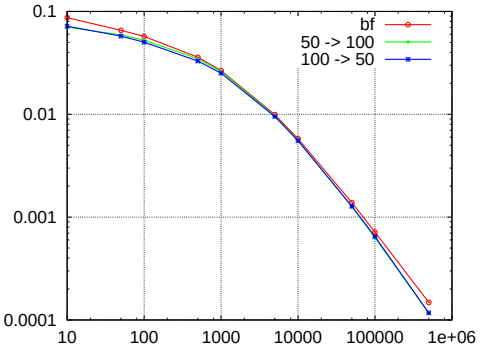
Figure 7.7: Long-term average gain rates achived by the bruteforce method (red line with circle) and our online method (green line with cross, blue with asterix). Inter patch travel time τ in seconds on the x-axis and long-term gain rate in pucks per seconds on the y-axis. More details in the text.



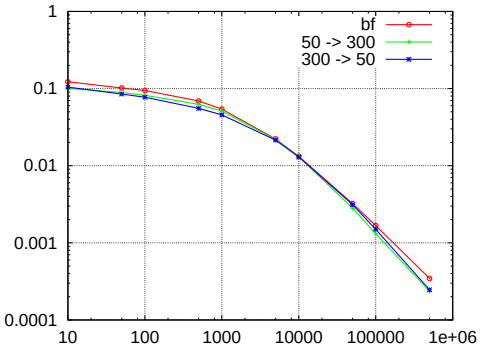
(g) Random sequence of patches with 50 and 100 pucks



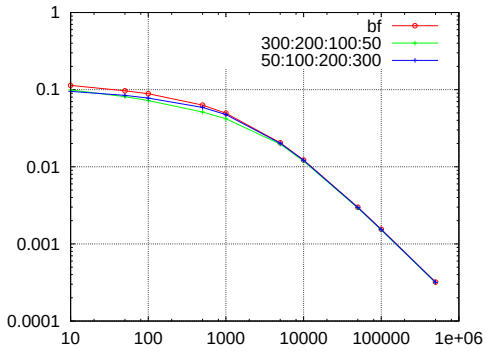
(h) Random sequence of patches with 50 and 300 pucks



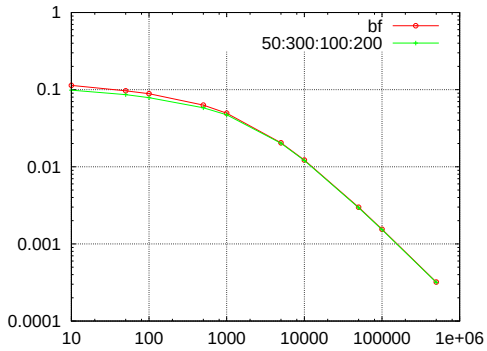
(i) Stepwise sequence of patches with 50 and 100 pucks



(j) Stepwise sequence of patches with 50 and 300 pucks



(k) Stepwise sequence of patches with 50, 100, 200 and 300 pucks



(l) Stepwise sequence of patches with 50, 300, 100 and 200 pucks

of the small standard deviation. As in the single patch type experiments and for the same reasons, the performance is somewhat lower under short switching time conditions, but in the general the graphs show that our method copes well with randomly encountered patches of different qualities.

An even harder problem is to encounter a longer series of patches of type A followed by a series of patches of type B. Where the forager does not know anything about type B patches while it forages in type A patches. In this experiment the robot was faced with a series of 100 patch of one type followed by 100 patch of a different type. The results for 50:100 and 50:300 patches with a stepwise change in both directions is shown in Fig. 7.7i and 7.7j respectively. Here the bruteforce method is at a significant advantage because the patch leaving decisions are derived with full knowledge of the future patch change. Our method does not/cannot anticipate the patch quality change and thus for the first 100 patches acts under the "assumption" of a constant environment. The error resulting from this "assumption" grows with the difference in patch qualities. That is why the performance difference in the 50:300 scenario (Fig. 7.7j) is larger then in the 50:100 case (Fig. 7.7i). Despite the limited amount of information our task switching method performs generally well.

Figure 7.7k shows the results for a stepwise sequence of 50:100:200:300 puck patches and the reverse ordering. The results are qualitatively very similar to those discussed previously. In one last experiment of this type we choose step wise patch encounter with larger step sizes. The ordering chosen was 50:300:100:200. Results are shown in Fig. 7.7l. The performance results are again qualitatively similar, suggesting the our method handles this type of variance well.

Variable Switching Cost

So far we tested different switching costs but kept them constant in the single patch type as well as multi-patch type experiments. To investigate varying inter-patch travel time, we conducted an experiment in which the travel time between patches was drawn from a normal distribution with mean 1000 seconds and standard deviation 100, 500 and 700 seconds respectively. Table 7.6 shows the results in percent compared to the long-term gain rate of the bruteforce solutions. Because of the computational complexity, the bruteforce solution was only calculated using the mean and not the actual randomly drawn travel times. As in the previous experiments we see generally good performance and the usual

drop in situations with low patch quality.

	initial pucks per patch					
σ	10	30	50	100	200	300
100	74.0	92.2	96.0	96.4	95.3	92.2
500	76.3	90.2	93.9	94.5	89.7	92.3
700	67.8	89.5	96.9	92.6	88.7	90.1

Table 7.6: Percent performance for variable patch switching time with mean 1000 sec. and standard deviation $\sigma = \{100, 500, 700\}$

7.5 Summary

Task switching under diminishing returns is daily routine for many animals and important for many conceivable autonomous robots. Maximizing the long-term average gain or reward rate under these conditions requires the robot to have knowledge of future gain functions. This is not achievable by a robot relying solely on information obtained by its own actions. To the best of our knowledge no solution to this problem is known. In this chapter we have argued that the MVT is not implementable because an instantaneous gain rate is meaningless in the case of rewards obtained in chunks. It also requires a continuous exploration phase in order to find the global maximum rate, but the MVT itself does not explore the action-space.

We proposed two task switching methods, one based on exploring the action-space in a greedy fashion, and the other one based predicting the gain rate using all past experiences as a predictor for the global gain rate. In Sec. 3.4.1 we discussed how the way the average is calculated has an influence on the behaviour. There we outlined the argument for a memory window made by Bateson and Kacelnik (1996). As future work it would be interesting to investigate how such a limit on memory size would effect performance. For example a low-pass filter over the experienced gain rate could be used as such an estimator.

Experimental Data

In accordance with the Autonomy Lab's policy on code publication, the foraging data and the implementation of the experiments are made available online at `git://github.com/`

`jwawerla/tsw_experiment.git`. The exact data that led to the presented results can be accessed via the commit hash `4f84a82d09f2c181df57ab5d7faa2e53cc3348f3`.

Chapter 8

Team-task Allocation in a Multi-robot Transportation System

So far in this thesis we looked at the single robot domain. We focused on solitary robots because their decision processes are generally less complex. Yet multi-robot systems have their advantages, often more work can be done per unit time and the combined sensor range can greatly increase the situation awareness of the overall system. One of the biggest challenges with multi-robot systems is the interference between robots. The result is that with increasing number of robots performance initially increases and at some point the performance decreases because the robots are in each others way (Vaughan et al.; 2000a) (See Sec. 3.5.1 for more details on interference reduction methods). From a performance point of view interference is negative, but interference, if we can measure it, also contains information. Here we show how interference can be used to adopt a near optimal allocation of robots to tasks. This chapter was also published as Wawerla and Vaughan (2010).

8.1 Problem Statement

Transporting goods is a natural task for autonomous mobile robots and in the future robots will perform more and more transportation tasks, from moving goods around in warehouses (Guizzo; 2008), delivering mail, bringing semi-finished products from one processing facility to the next, supplying factories, to world-wide shipping. To achieve this, we need

systems that assign transportation tasks to robots. Several versions of the problem can be stated, e.g. fixed or changing pickup or delivery points, changing rates at which goods have to be transported and varying fees.

In this chapter we consider a multi-robot transportation system with fixed and known source and sink locations and constant fees. The source produces goods at a certain, variable rate, but cannot store more than one unit at a time. Fig 8.1 shows an example of this case. The objective of the robots is to transport goods, considered abstractly as pucks, from a source to the corresponding sink. Pucks may not be transported to any other sink. To be economically viable the task-allocation has to be cost effective. We use energy expended by the robots as our unit of cost and pucks transported as our work metric. To reduce the cost, the task-allocation system can deallocate a robot temporarily by sending it to a depot, marked D in Fig.8.1. There the robot is on stand-by and does not require any energy.

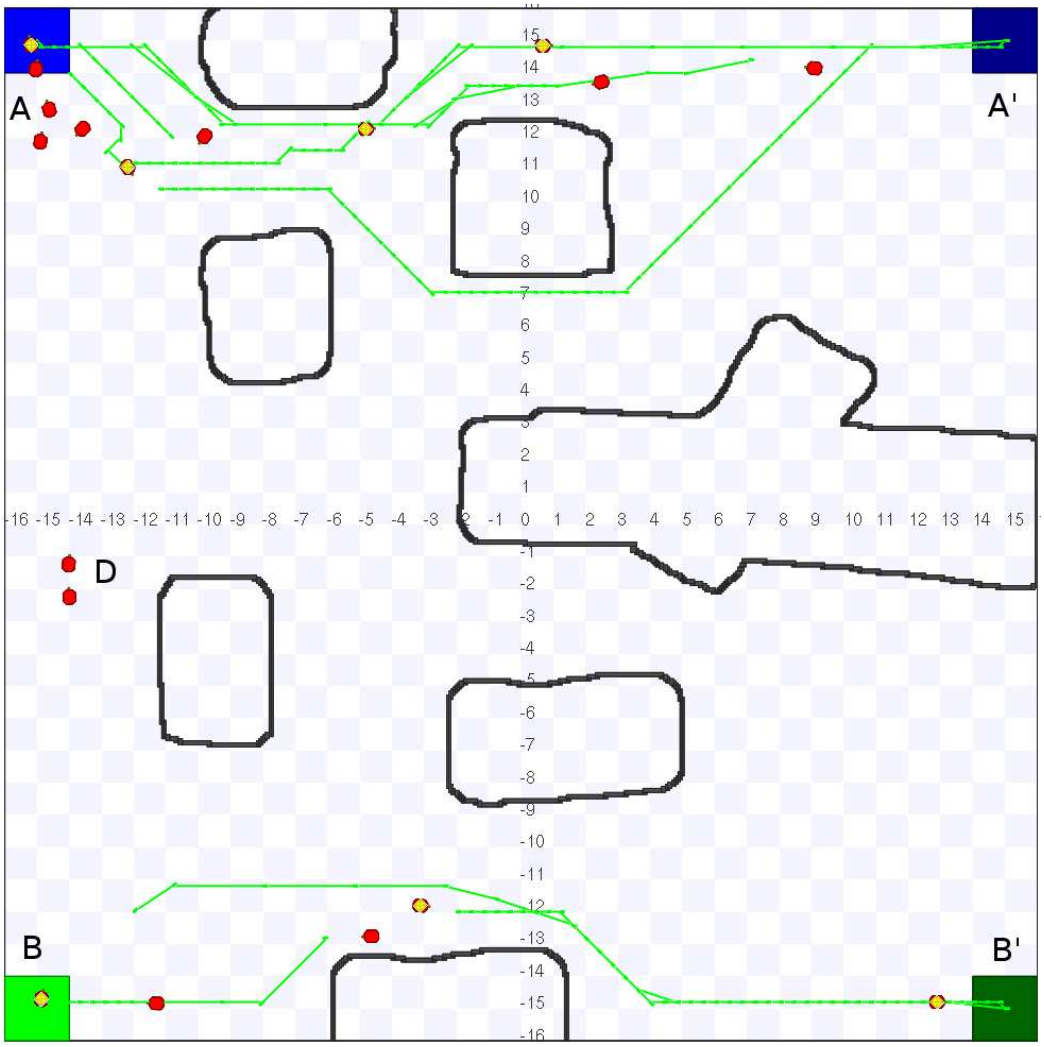


Figure 8.1: Screenshot of a Stage simulation with two tasks, transporting pucks from A to A' and from B to B'. D marks the robot depot for unassigned robots. The thin (green) lines show the path planed by each robot. Robots carrying a puck are displayed with a (yellow) diamond on top.

We propose two task-allocations methods, (1) based on a central, omniscient planner and (2) governed by a heuristic, using only locally sensed information and minimal and infrequent communication between the robots. The challenge for both methods as well as any other task-allocation algorithms lies in correctly dealing with interference, inherent to all multi-robot systems. We show that a naive solution creates an unintended feedback loop

due to interference, that leads to a drop in performance. We suggest a simple yet effective solution to this problem.

8.1.1 Related Work

Robot task allocation is a widely studied field. It can be broadly classified into two classes: planner based systems and systems that rely on individual robots making individual task allocation decisions. Planners are often based on auction mechanisms in which robots bid for tasks. Gerkey's MURDOCH system (Gerkey and Matarić; 2002) is one such example. As we will show, the problem we study does not require the sophistication of this class of planners.

Local decision mechanisms are attractive because they are redundant and by definition do not rely on a centralized agency. Labella et al. (2004) use a minimalistic rule originating from modelling ant behaviour. Each robot's probability to leave the nest is increased by a small δ if it found food on the last foraging trip and decreased by the same δ if the robot returned empty handed. Liu et al. (2007) present a similar central place foraging system but with the explicit goal to minimize energy expenditure. The decision each robot makes is based on two thresholds. One for the amount of time spent resting and one for the time spent foraging. The thresholds are adapted by certain cues. Colliding with other robots makes a robot more likely to rest and successfully retrieving food makes the robot more likely to keep foraging. In an experiment with up to 12 Khepera robots Krieger and Billeter (2000) use a similar threshold based task-allocation mechanism to keep the swarm's energy reserve at a safe level. Krieger and Billeter (2000) also provide an extensive list of references to studies about task allocation in social insects. All three methods have been shown to arbitrate between work and rest periods but it is not clear how to select between two work tasks and resting.

Jones and Matarić (2003) propose a probabilistic rule that is shown to work for at least two tasks. The robots calculate the probability of switching tasks based on a local estimate of the number of robot performing the same task and on an estimate of the available work load. This requires the robots to communicate what task they are currently engaged in. Jones explicitly prohibits an idle task. McFarlands's Cue×Deficit-rule (McFarland and Spier; 1997) allows robots to select between two types of resource depending on the robot's deficit and encounter rate of a particular resource. McFarland models situations in which the robot is in need of both resources, e.g. needing water and food. In the transportation problem

considered in this chapter a deficit of a certain task is not meaning full. McFarlands's method also requires encountering both resources at least every once in awhile in order to update the cue estimates. This cannot be guaranteed in a transportation problem.

Some work uses machine learning techniques to estimate the utility of tasks. A recent example is Dahl's vacancy chain scheduling (Dahl et al.; 2009). Reinforcement learning and ϵ-greedy action selection is used to have robots choose between two transportation tasks. The scenario is similar to ours but less challenging due to the absence of fixed obstacles in the world. Also energy is not considered in the performance metric. The method we propose is significantly less complex then that proposed by Dahl and does not suffer from any explicit trade-off between exploration and exploitation as Q-learning does.

Aside from robotics, this transportation problem is related to queueing theory. Queueing theory is the mathematical study of waiting lines. Reviewing the entire theory is beyond the scope of this document, the interested reader is referred to introductory books on the subject like (Gross; 2008; Stidham; 2009). In general queueing theory aims at calculating performance measures including the expected number agents receiving service, the average waiting time, and the probability of encountering a system in a certain state, e.g. queue full or empty. Queueing theory is usually used to design a queueing system, for example telephone switches or checkout counters in a supermarket. For that the behaviour of the customer is assumed fixed and often modelled with a Poisson process. In this chapter we face the inverse problem, the service is fixed and we can control the behaviour of the customers. The other major difference between our work and queueing theory is that at least for the local heuristic the decision process does not have full knowledge of the production and service rates. The centralized approach of queueing theory necessitates to have this knowledge.

Queueing theory is a mathematical technique originally developed to design telephone services in the 1920s. Since then numerous attempts have been made to extend it to more general problems. But as van Dijk (2002) points out "[..] the application of queueing theory for daily-life problems has remained rather restricted". Taha (1981); van Dijk (2002) argue that mathematical models of real world queueing systems are very often too complex to be solvable. Instead either simulations or hybrids (combinations of queueing theory and simulation) are used to model and analyze practical queueing systems. This is not to say that our problem cannot be model using queueing theory, we just want to point out possible limitations. In any case a mathematical model of the system does not give us a decentralized control policy, the goal of our work.

8.2 Robot System

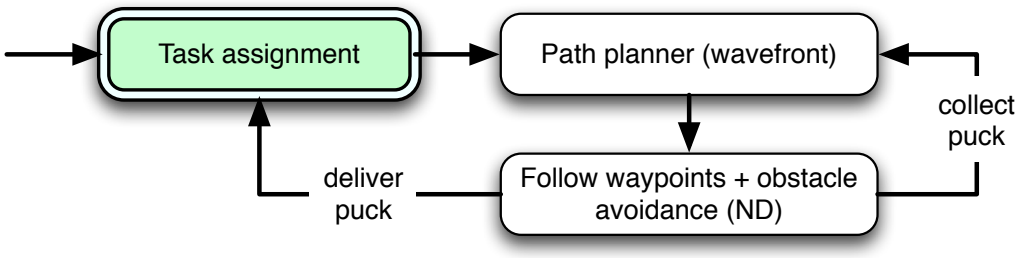


Figure 8.2: Overview of the robot system

The robots used in this work are generic models of a differential drive robot simulated in Stage (Vaughan; 2008). Each robot is equipped with a 360° field of view laser range finder providing 64 samples with a maximum range of 5 meters. Each robot can localize itself with an abstract localization device. This models a GPS[1] module or any form of SLAM implementation. To transport pucks each robot is capable of sensing pucks, picking them up and dropping them off. The cargo capacity is limited to one puck per robot. In order to communicate with other robots or with the centralized planner the robots are equipped with a generic communication device. We assume reliable communication, but as we will show communication used by our task allocation policies is minimal.

As energy expended by the robots is one of our performance measures the energy model of the robots is important. Two types of energy expenditure occur, (1) energy used for computation and sensing is expended at a constant rate and (2) energy used for locomotion is expended at a rate proportional to the speed of the robot. The first energy model is fairly accurate. The second model does not take acceleration into account and is thus not fully realistic. Acceleration and deceleration occur frequently in high interference situations, therefore our energy model does not penalize interference as much as one would expect. The actual energy rates are modelled after a typical Pioneer 3DX robot. Robots in the depot are in a stand-by mode and do not use any energy.

The robot control software is split in two parts, (1) the common part that handles navigation, obstacle avoidance, picking up and delivering pucks etc. and (2) the task allocation part that decides which task to perform next. An overview is shown in Fig. 8.2.

[1]Global Positioning System

Path-planning is done using a wavefront planner (WP). This widely used technique discretizes the world into a grid and builds a gradient of shortest distances starting at the goal location. The shortest path is then given by gradient decent from the robot's current position. (see (LaValle; 2006) for details). In our implementation we augment the map data with sensor data from the past n time steps. This enables the planner to incorporate knowledge about temporary changes in the world such as congestion. Figure 8.1 illustrates this feature, one robot has planned a seemingly longer path around the south side of the obstacle at (0,10). While four other robots planned the shorter path along the often more congested north side of the same obstacle. Emphasis was put on reliability of the planner and not so much on interference reduction. For example the introduction of simple traffic rules such as "always stay on the right side of corridors" may reduce interference noticeably. But as highways in any major city during rush-hour are evidence, interference cannot be avoided as the number of agents increases. In fact to investigate our task-allocation methods we want to explore the configuration space under low and high interference conditions.

The planned trajectory is used by a sensor-based obstacle avoidance routine to navigate the robot along the trajectory to the goal. We use Minguez's Nearness Diagram (ND) (Minguez and Montano; 2004) an extension of the Vector Field Histogram (Ulrich and Borenstein; 1998) approach. Minguez's implementation is available on his homepage[2].

The key contributions of this chapter are the task-allocation methods. We investigate and compare two methods, both using the same navigation and control routines described above.

8.2.1 Allocation Re-Planner

The allocation re-planner is a centralized planner that has full knowledge of the world, the production rate of the sources and the current robot task assignments. An optimal assignment is achieved if the puck production rate of a task equals the sum of the transport rate of all assigned robots. Only under this condition is there no unused transport capacity nor are pucks waiting at the source. The optimal number of robots is thus given by

$$N_i = \frac{\dot{p}_i T_i}{c} \tag{8.1}$$

[2] `http://webdiis.unizar.es/~jminguez/code/sources_ND_english.zip`

where the index i refers to the i-th task, $\dot{p}_i$ is the production rate of the task, T_i the round trip time the robot needs to drive from the source to the sink and back, and c is the puck carrying capacity of the robot ($c = 1$ in our case). It is fair to assume that a central planner has accurate information about the robot's carrying capacity and the production rate for each task. But knowledge of the round trip time is difficult to obtain. The round trip time can be estimated from the distance between source and sink and the speed of the robot. The distance between source and sink maybe accurately obtained from a path-planner e.g. the wavefront planner. The actual speed of the robot is a function of the robots hardware, the control software, as well as the interference between the robot and other robots or fixed obstacles in the environment. Modelling this interference and the resulting change in speed is difficult. For example, in the environment in Fig. 8.1 task A is subject to a higher degree of interference then task B because of the narrow corridor between the obstacles at (0,10) and (-8,14). Modelling interference is generally a difficult endeavour. A simple model is proposed by Seth (2002). Given an interference factor Seth's model uses this factor and the number of agents to calculate the change in performance. This approach has two limitations, (1) we do not know the interference factor and (2) since this model calculates the effect of interference on the average performance and not on a sensori motor level, it is too high level for our purpose. Instead of modelling interference, we take the embodied approach. A naive solution is to average over the actual round trip time as experienced by the robots, and base the planner's allocation on this time estimate. Unfortunately this method has an undesired feedback loop. Randomly occurring interference will increase the estimate of the round trip time, causing the planner to allocate more robots to the task and thus eventually increasing the interference. This in turn results in even more allocated robots.

What is needed is a way to distinguish between the "true" round trip time and delays caused by interference due to randomly occurring congestion. As a simple measure for interference we use the speed of the robot. We sum up the time steps during which the robot's speed is below a certain threshold. The robots then report the experienced round trip time corrected for the time wasted due to interference and the time spent waiting at the source and sink $\hat{T}_i$.

Once a robot completes a task the planner may assign it a new task using Eq. 8.1 and a low-pass filtered estimate of the, interference corrected experienced round trip time. Alternatively the robot might be sent to the depot if it is no longer needed. The complexity of this planning step is $O(k)$ where k is the number of tasks. In the depot, robots are waiting

in a queue and only the head of the queue queries the planner for a new assignment.

8.2.2 Allocation Heuristic

The allocation heuristic has no access to any information other than the location of sources and sinks. Since the production rates are unknown, robots initially select a task at random with equal probabilities. Every time a robot returns from delivering a puck it faces one of three possible situations:

1. robots are already waiting in a queue to pick up a puck
2. no robots are waiting and a puck is available for pick up
3. no robots are waiting and no puck is available yet

Situation (1) can be evidence that the number of robots assigned to the task is too high. It can also be a result of an unfortunate spatial clustering of robots. Situation (2) might indicate that the number of robots assigned to the task is too low since the production rate appears to be higher then the transportation rate. But again this could also be a consequence of a brief congestion somewhere along the transportation route. Situation (3) is, at least from this particular robots point of view, the ideal case. On one hand the absence of a robot waiting queue indicates that there are not too many robots assigned to the task. On the other hand pucks not already waiting indicates that the task has recently been serviced by another robot.

The allocation heuristic is entirely based on these three situation, which are easily identifiable by the robots. A robot in situation (1) randomly chooses a maximal time it is willing to wait in the queue according to

$$t_{wait} = max(t_{minwait}, \hat{T}_i \cdot r) \tag{8.2}$$

where $t_{minwait}$ is a minimal waiting time, $\hat{T}_i$ the corrected round trip time and r is a uniform random number in the interval $[0, 1]$. If the robot is able to enter the loading bay before the waiting time is up, it will continue with the current task. Otherwise it stops performing any task and returns to the depot.

A robot in situation (2) broadcasts a worker recruitment message for the current task with a probability p_b. The first robot in the depot will respond to the allocation call by starting to perform the task in need of an additional worker. In the case that there are

no robots in the depot, the allocation message remains unanswered. Situation (3) does not require the robot to alter the allocation, thus it simply continues to perform the task. Because production rates may drop to zero, a robot in situation (3) will only wait for a puck to be produced for a maximal duration. We choose the last experienced round trip time to be the maximal waiting time before returning to the depot.

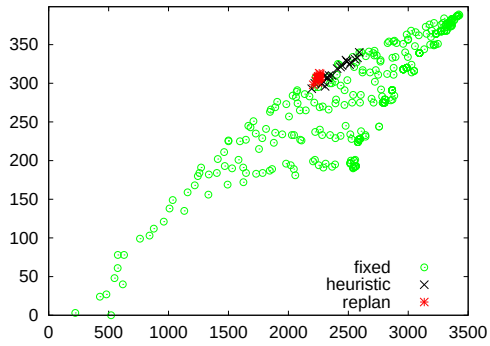
Note that at no point are the robots obtaining any information about the task production rates, the number of assigned robots per task, or the number of unassigned robots. The group allocation is entirely based on locally observed information and a single value broadcast message, used infrequently.

8.3 Experiments

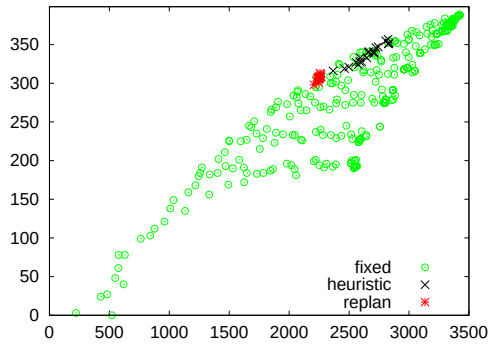
All experiments reported in this chapter were conducted with 18 simulated mobile robots in Stage (Vaughan; 2008) a sensori motor level multi-robot simulator. The robot controller was the same during all experiments, just the task selection component was swapped. The simulation environment is shown in Fig. 8.1. Initial starting positions of the robots, the location of source and sink and the total sum of production rates were fixed. Only the ratio of the production rate of the two tasks was altered. Performance was measured in two dimensions: total number of pucks transported in a fixed amount of time and total energy expenditure during this time. These values are not directly convertible into a single performance criterion. The result of any form of linear combination is dependent on the weighting and is thus application dependent and not suitable for a general comparison. Due to the cost of interference, ratios like pucks transported per unit energy, generally favour allocations where only one robot is performing the task while the remaining robots are in stand-by. So we report the results of our experiments in a energy-work graph, similar to a precision-recall graph in machine learning. Results closer to the upper left corner of a energy-work graph should be considered better. Less energy is then expended and more work performed. All experiments were run for 2 simulated hours.

8.3.1 Constant Production Ratios

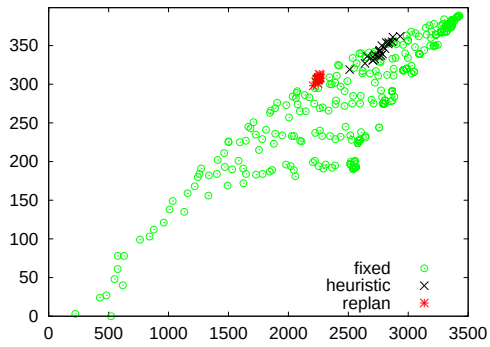
In a first set of experiments we kept the production rate constant at 1:1, 2:1 and 10:1 respectively. Each configuration was tested 20 times. Fig. 8.2 shows the results in energy-work graphs for the re-planner and the heuristic allocation for a selection of broadcast



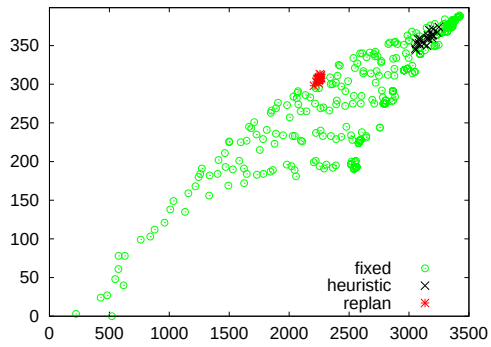
(a) Ratio 1:1, broadcast probability 0.0



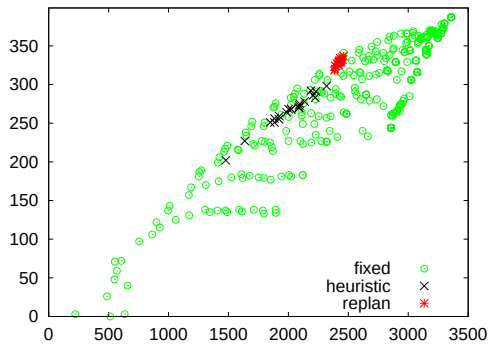
(b) Ratio 1:1, broadcast probability 0.1



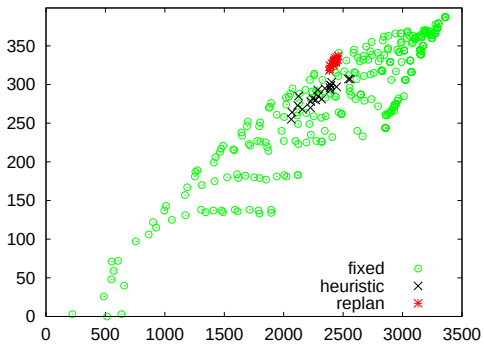
(c) Ratio1:1, broadcast probability 0.2



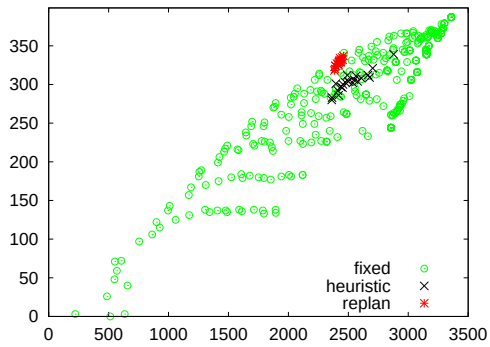
(d) Ratio 1:1, broadcast probability 1.0



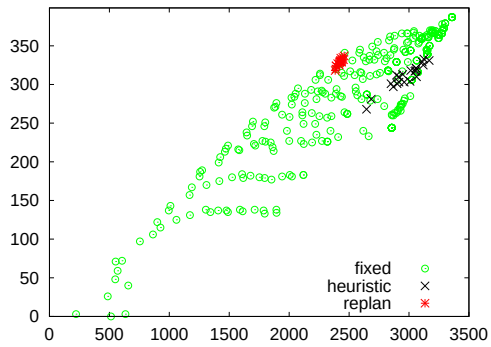
(e) Ratio 2:1, broadcast probability 0.0



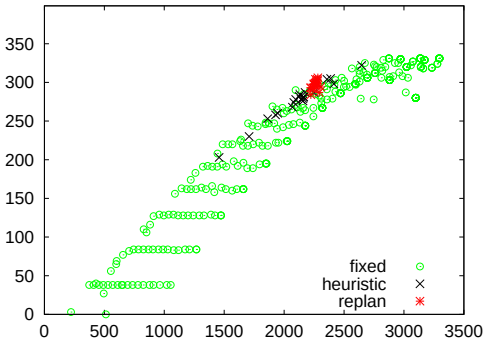
(f) Ratio 2:1, broadcast probability 0.1



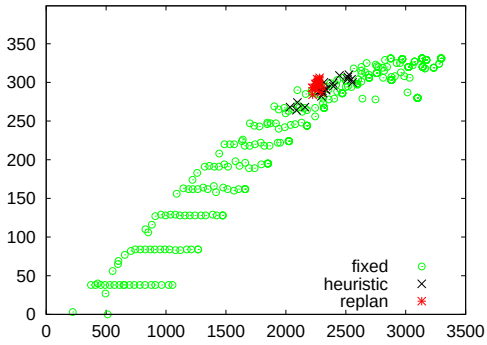
(g) Ratio 2:1, broadcast probability 0.2



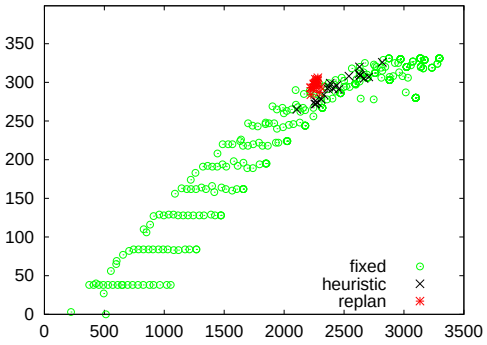
(h) Ratio 2:1, broadcast probability 1.0



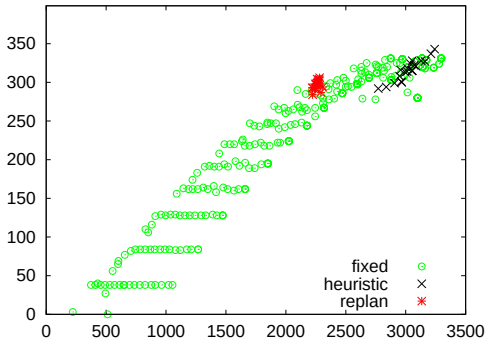
(i) Ratio 10:1, broadcast probability 0.0



(j) Ratio 10:1, broadcast probability 0.1



(k) Ratio 10:1, broadcast probability 0.2



(l) Ratio 10:1, broadcast probability 1.0

Figure 8.2: Energy-Work graphs for different production rate ratios and broadcasting probabilities. Energy expended is plotted on the x-axis and pucks transported on the y-axis. The green circles mark results from fixed robot assignments to visualize the possible performance space. The performance of 20 trials of the re-planner is shown in red asterisks and the performance from 20 trials of the allocation heuristic is shown in black crosses.

probabilities. We tested probabilities with 0.1 increments, but due to space constraints only show the most interesting ones.

In order to not only compare our two task-allocation methods with each other and to allow the reader to see where our solutions lie in the possible energy work space, we also characterize what this space looks like. Therefore, we assigned a fixed number of robots to each task and kept this assignment constant during the course of a 2 hour simulation run. We repeated this process for all possible ways to assign up to 18 robots to two tasks, where the unassigned robots return to the depot, just like in the case of the other two policies (a total of 189 experiments).

As mentioned earlier it is application dependent whether we wish to minimize energy expenditure or maximize puck transportation or find some middle ground, but in general we wish to minimize wasting energy. Therefore, control policies that achieve the same number of pucks transported while spending less energy must be considered better policies. From the fixed assignment we can see the line of good performance, it is the left hand side of the fixed assignment energy-work tuples. From Fig. 8.2 we conclude that the re-planner performs very well, since it's results are on the line of good performance for all configurations tested.

The results from the allocation heuristic are subject to a higher degree of variance. This is to be expected due to the stochastic nature of the heuristic. Further we observe that a low broadcast probability yields results closer to the line of good performance. The reason is that a higher probability causes a higher frequency of re-assignments and these cost energy as the newly allocated robot travels to the worksite. The key to the heuristic proposed is to trade-off re-assignments and adaptation to the task configuration. From the graphs it is apparent that this adaptation is more difficult in the 2:1 case. The 1:1 case is easier, because of the initial random task selection, the ratios of robots already closely matches the production rates and "only" a reduction of workers has to be achieved by the system. The cues used by the heuristic are more pronounced in the 10:1 case, because the ratios are so much more different.

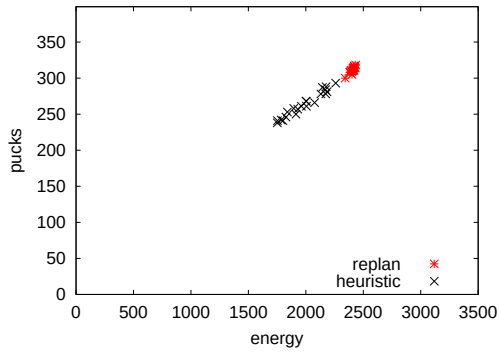
Summarizing, the allocation heuristic performs remarkably well considering that no information about the production rates or the allocation of other robots is known to the heuristic.

		Rate Ratio	
		3:1	1:3
re-planner		8.0 : 3.0	3.1 : 8:0
heuristic with broadcast probability	0.0	7.4 : 3.6	3.4 : 3.6
	0.1	7.8 : 4.0	3.6 : 5.3
	0.2	8.5 : 4.1	3.7 : 6.3
	0.3	9.1 : 4.1	3.7 : 8.1
	0.4	9.4 : 4.2	3.8 : 9.0
	0.5	9.5 : 4.1	3.7 : 9.7
	0.6	10.0 : 4.2	3.9 : 9.7
	0.7	10.1 : 4.1	3.8 : 10.4
	0.8	10.4 : 4.3	3.8 : 11.0
	0.9	10.4 : 4.4	3.9 : 11.2
	1.0	10.6 : 4.4	3.8 : 11.7

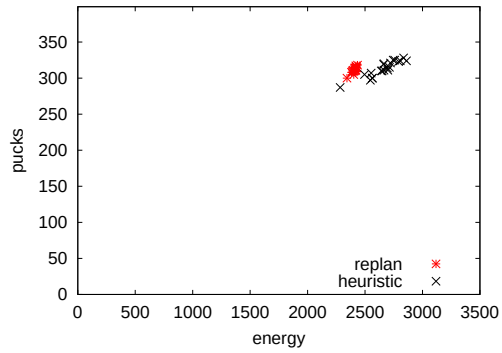
Table 8.1: Average number of robots assigned between two tasks with stepwise change of production rates from 3:1 to 1:3

8.3.2 Changing Production Ratios

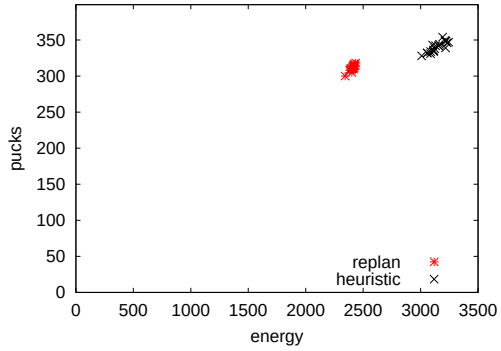
Next we investigate how the two policies perform in situations of changing production rates. We conducted an experiment in which we set the production ratio to 3:1 for the first hour and then reversed it to 1:3 for the second hour. Unlike in the constant rate situation, a fixed assignment obviously does not show us the possible energy-work space. So we restricted our analysis to comparing the re-planner with the heuristic. The results are summarized in energy-work graphs (Fig. 8.3) and the average robot assignment numbers in Tab. 8.1. The assignment numbers are the average number of robots assigned to a task during a production rate period. As expected, with zero broadcast probability, the heuristic just de-allocates robots once the production rate of the first task drops. With probability 0.3 there is enough recruitment so that the ratio of assigned robots matches that of the re-planner. The number of assigned robots is generally higher, because of the overhead of constant re-assignment. As the probability increases we observe an increase in overhead and thus higher assignment numbers but the ratio is similar to that of the re-planner. Disabled recruitment (prob. 0.0) means fewer workers after the ratio change and that results in fewer pucks transported (Fig. 8.3a) and less energy spent. Using a broadcasting probability of 1.0 does not yield more pucks transported but results in more energy wasted on frequent re-allocation (Fig. 8.3c), this coincides with our analysis of the assignment numbers. More



(a) Ratio 3:1 → 1:3, broadcast prob. 0.0



(b) Ratio 3:1 → 1:3, broadcast prob. 0.3



(c) Ratio 3:1 → 1:3, broadcast prob. 1.0

Figure 8.3: Stepwise change of production rates from 3:1 to 1:3. Energy expended is plotted on the x-axis and pucks transported on the y-axis. The results from the centralized planner are shown as back crosses and the heuristic performance is shown as red asterisks.

interesting is Fig. 8.3b, which shows that the work rate achieved by both policies is about the same (broadcast probability of 0.3), but the heuristic requires more energy. As in the first experiments, lack of information has to be paid for with energy.

8.4 Summary

For energy to be used in a performance analysis of simulation experiments an accurate model of the robot's energy consumption is required. Mainly this model must be able to capture possible differences in energy requirements of the sensori motor activity caused by different control policies. To test our methods under more realistic circumstances we are currently working on real robot experiments. An issue not addressed in this work and left

for the future is refuelling. Our current implementation can handle refuelling in principle but more, and especially longer, experiments are required for a thorough investigation.

The sources of the transportation task used throughout this chapter were only able to store one puck at a time. In the future we plan to investigate systems in which pucks accumulate at the source if the robots do not pick them up in time. Another question left for the future is how does task priority, e.g. expressed in terms of task-dependent rewards, influence the allocation policies?

This chapter introduced two task-allocation strategies for a multi-robot transportation system. The computational complexity of both strategies is small. In simulation experiments we compared the strategies in the form of energy-work graphs. It is not possible to generally say which method performs better because we lack a unifying metric. But we can conclude that the re-planner's performance is on the line of good performance. The performance of the heuristic allocation is often slightly below the line of good performance. Yet the heuristic still performs well, considering that it uses no information about the production rates nor about the task allocation of other robots. This simple rule based allocation policy yields remarkable results entirely relying on local information and minimalistic broadcasting.

Experimental Data

In accordance with the Autonomy Lab's policy on code publication, the source code and analysis scripts of the experiments are made available online at `git://github.com/rtv/autolab-fasr.git`. The exact data that led to the presented results can be accessed via the commit hash `ed47f9212cf3b72f6a083b48bc2d39a1b79006af`.

Chapter 9

Conclusion

We began this thesis by introducing the *Fungus Eater*, a well-known conceptual robot introduced by the psychologist Toda to investigate embodied decision-making. In order to implement a controller for this conceptual robot, we first had to clarify some of Toda's definitions which were imprecise. Specifically, there was an obvious lack of details about the environment of the robot and little specificity about the reward structure of ore collection. Another obstacle when examining robot controllers in the framework of *Fungus Eater* is the potential for an infinite planning horizon, or at least a potentially very long planning horizon based on the lifetime of the robot. This is computationally intractable. After reviewing the relevant literature on action-selection in robotics and behavioural ecology, we were able to introduce the notion of behavioural cycles, a refinement of utility cycles. An example of such a cycle would be work, look for energy, refuel, look for work. The obvious benefit of such a cycle is that the robot always returns to a similar location in the state-space. Since rational agents are expected to make the same decisions under the same circumstances, this permits us to plan just a single cycle if the environment is constant.

Armed with this newly defined framework, we presented solutions to a set of self-sufficiency scenarios: discounted labour, rate maximization, and patch switching. We drew insight from the field of behaviour ecology to solve these subproblems because this field provides descriptive models of animal behaviour which closely resemble the self-sufficiency problems discussed in this thesis. We implemented robot controllers based on the principles behind these descriptive models for optimal animal behaviour. A direct implementation may not be possible because the models are often too abstract or describe behaviour from

an observer point of view and not from an embodied and situated angle. That was the situation we faced when looking for suitable models for the *Fungus Eater* problem. We successfully developed control policies to solve these subproblems and showed that the resulting behaviour was near-optimal.

Expanding on the solitary *Fungus Eater*, we examined control policies for task allocation in multi-robot systems. Specifically, we evaluated the performance of a centralized planning system capable of allocating robots to different tasks. We then looked at a distributed method based on the ability of robots to utilize local cues to perform task switching. One key issue with evaluating the performance of both of these control policies was that a good, concrete utility function is not available and is always domain dependent due to the trade-off between work and energy.

In summary, our ultimate goal is to develop control policies for self-sufficient robots and we have used the *Fungus Eater* problem as a framework to do so. By looking at behavioural ecology models from a sensori motor point of view, we showed that our control policies, developed to solve aspects of the *Fungus Eater* problem, indeed generate the desired behaviour based on the limited sensor information of a physical robot. We want to reiterate that our policies do not necessarily closely resemble the mechanisms that control animal behaviour.

Have we succeeded building a *Fungus Eater*? A fair answer is, no, we have not built a *Fungus Eater*, but we have successfully created controllers that bring us closer to the ambitious goal of self-sufficient robots and helped to further define the problem space. This thesis also demonstrates the rich potential for solutions to problems with building autonomous robots by looking to other fields, especially behavioural ecology. While we have found solutions for the example scenarios presented here, many more, and especially more complex ones, are conceivable. As evident in the vast behavioural ecology literature, we are still unable to fully understand, and therefore cannot implement, optimal behaviour for self-sufficient agents in the general case.

In particular this thesis contributes:

- a refinement of Toda's *Fungus Eater* scenario to create a formal robotics problem (Ch. 2).
- a framework to characterize task-switching for self-sufficient robots. The framework extends McFarland and Spier (1997) framework by explicitly stating the switching

costs and the gain rates of each possible task (Ch. 4).

- a robot control policy for a chain of point style work sites based on rate-maximization. Previously the state of the art in robot recharging was to employ a fixed energy remaining threshold policy to determine when to recharge. In our work we showed that an adaptive threshold policy performs better than the traditional fixed threshold policy. By employing a rate-maximization policy we were able to improve performance again, compared to the adaptive threshold policy (Ch. 5).

- recharging strategies for discounted labour. A key result from this work is a method to determine if the robot should be semelparous or iterparous. Traditionally iterparity was assumed to be optimal for robots, but we showed that under certain circumstances it pays to not recharge at all (Ch. 6).

- two patch switching methods based on the marginal-value theorem. To the best of our knowledge diminishing returns have not been explicitly considered previously in the robotics literature. By overcoming the practical problems associated with the MVT, be were able to derive two sensori motor based control policies for task-switching under diminishing returns. We analyzed the performance of these policies using a foraging scenario, but it is worth noting that the task-switching algorithms are not limited to foraging. Our work also showed that the principals behind the MVT are usable by a situated agents, interesting future work would be to investigate if we could thereby explain under- or overmatching observed in animals, e.g. with certain filter constants (Ch. 7).

- a minimal, centralized planner for allocating a team of cooperating robots to tasks with the objective to maximize overall team efficiency

- a decentralized task-allocation heuristic for a team of robots with the objective to maximize overall team efficiency (Ch. 8).

9.1 Future Work

There are numerous open questions regarding control policies for self-sufficient robots. In this section we highlight the ones that seem the most pressing for us at the moment.

In Sec. 4.2.7 we introduced a scenario with diminishing returns and capacity limits. We argued that the standard approach to handle diminishing returns, the MVT, is not applicable in this case. The reason being that the MVT does not consider any capacity limits and thus may switch patches even when the onboard storage is almost full and thus only a very small amount can be gathered from the new patch. An extension to the MVT is needed to cope with this issue.

Another scenario left unsolved is the resource chain model depicted in Fig. 4.7. We discussed possible solutions for some special cases of this problem and pointed out future research direction for the general case in Sec. 4.2.8.

All the scenarios we considered assume that the rate at which energy is expended during work is constant. This is clearly a simplification since different tasks usually require different energy rates, e.g. programming robots requires a smaller energy rate than digging a hole.

We only investigated cases in which the exchange rate between resources is zero, in other words we did not allow trading one resource for another. Surely in a more general case at least some resources should be exchangeable. The *Fungus Eater* could for example trade some of the collected ore with another robot for fungi. Or a robot like the Slugbot (Kelly and Melhuish; 2001) that usually feeds on slugs, could be charged with the task to pick apples, say in addition to slugs it could also feed on apples (maybe not as efficiently), what decision should it make? Presumably the decision depends on the exchange rate and the switching cost.

Mapping is an important field of robotics research, which typically means generating maps, ideally on the fly, and using them to localize the robot (for more details on mapping see Monemerlo and Thrun (2007); Thrun et al. (2005)). An open question is how we include information about resources into maps and reason about resource maps. Since the primary objective of the robot is to obtain a reward and not to generate a resource map, we have to address the exploitation/exploration trade-off. Part of this question is deciding when the map is accurate enough to make an informed decision and constantly evaluating if the map is outdated and needs updating.

In this thesis we only considered resources and tasks that did not change without any action by the robot. In the future we have to investigate situations in which resources are subject to change due to other reasons. This might be other competing robots, environmental effects like weather, or cyclic behaviour of the resource it self, e.g. circadian or annual periods. Some animal species are clearly capable of such decisions. McFarland

(1977) reports herring gulls capable of learning the weekly work schedule of a landfill. The birds would forage for food in the freshly delivered garbage from Monday to Friday and they would be absent on weekends (forage elsewhere). McFarland interprets the fact the gulls would appear at the landfill on public holidays as an indicator that the gulls did not utilize any auxiliary cues (like engine noise) to determine their behaviour. Robot controllers that can handle temporal changes in resource distributions would most likely have to have some model of the temporal change of the resources. The question then is how to learn such a model? An issue that comes hand in hand with uncertainty in the resource availability is short-fall risk, which was briefly discussed earlier. Conceptually one approach could be to learn the probability distribution of the resource availability and use it as a basis for any decisions about energy reserves. In order for the robot to make this kind of decision, we would have to specify the degree of risk we want the robot to accept. The challenge with learning survival critical information is that the learning system must not make any mistakes.

In chapter 8 we presented two algorithms for team task-allocation. The objective was to maximize the efficiency of the whole team. This cooperative setting included deallocating robots entirely. A very different problem for the same general setting is to maximize the efficiency of each individual robot. This is different in two main aspects, (1) robots cannot be deallocated and (2) it may be beneficial for robots to not cooperate with each other. From Fretwell (1972) we know that the robots should reach an ideal-free distribution (Sec. 3.4.4). What is unknown is what situated and embodied control policy would lead to this distribution or an approximation of it. Because of the switching cost, we would expect deviation from an IFD to some degree. An interesting additional action-selection problem worth investigating is to what degree the robot should cooperate. For example, how much is a robot willing to exchange information with other robots? This could lower its uncertainty about other work locations which would be beneficial. On the other hand it could also increase interference at the current work site if the information exchange attracts others. Under circumstances where the production rate of the work site is a lot higher than the work rate of the robot, recruiting additional robots will not be harmful. Attracting more robots if the production rate is lower than the work rate of the robot would be disadvantageous.

Multi-disciplinary research, as presented in this thesis, can provide new insights by looking at a problem from a different angle. The behavioural ecology literature provided a great deal of inspiration and information to take a well-known robotics problem into a new

direction. This multi-disciplinary approach should be extended to include research fields like operations, psychology, and economics, to name just a few. It is not simply a one-way inspiration from other fields into robotics. Robots, and task-switching research in particular, could be used to as an embodied system for testing theories in behavioural ecology. Further, the practical applications of robots able to make refuelling and work decisions, and especially manage multi-robot task allocation, has obvious benefits for automation. In summary, robot task-switching remains an important, exciting, and interesting research topic.

Bibliography

B. W. Andrews, K. M. Passino, and T. A. Waite. Foraging theory for autonomous vehicle decision-making system design. *Journal of Intelligent and Robotic Systems*, 49:39–65, 2007.

D. Austin, L. Fletcher, and A. Zelinsky. Mobile robotics in the long term - exploring the fourth dimension. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 613–618, Wailwa, Hawaii, October 2001.

M. Bateson and A. Kacelnik. Rate currencies and the foraging starling: the fallacy of the averages revisited. *Behavioral Ecology*, 7(3):341–352, 1996.

W. M. Baum. On two types of deviation from the matching law: Bias and undermatching. *Journal of the Experimental Analysis of Behavior*, 22:231–242, 1974.

A. Birk. Learning to survive. In *Procs. of the 5th European Workshop on Learning Robots*, pages 1–8, Bari, Italy, 1996.

A. Birk. Robot-learning and self-sufficiency: What the energy-level can tell us about a robots performance. *Lecture Notes in Computer Science*, 1545:109–125, 1998.

V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1986.

J. L. Bresina, A. K. Jonsson, P. H. Morris, and K. Rajan. Activity planning for the mars exploration rovers. In *International Conference on Automated Planning and Scheduling*, 2005.

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

R. A. Brooks. A robot that walks; emergent behavior from a carefully evolved network. *Neural Computation*, 1(2):253–262., 1989.

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

R. A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139—-159, 1991.

S. Brown, M. Zuluaga, Y. Zhang, and R. Vaughan. Rational aggressive behaviour reduces interference in a mobile robot team. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 741–748, Seattle, Washington, July 2005.

R. R. Bush and F. Mosteller. *Stochastic Models for Learning*. Wiley, 1955.

Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226–234, 1997.

E. L. Charnov. Optimal foraging, the marginal value theorem. *Journal of Theoretical Biology*, 9(2):129–135, 1976a.

E. L. Charnov. Optimal foraging: Attack strategy of a mantid. *The American Naturalist*, 110:141–151, January 1976b.

E. L. Charnov and G. H. Orians. Optimal foraging: Some theoretical explorations. Unpublished manuscript `http://hdl.handle.net/1928/1649`, 1973.

E. L. Charnov and W. M. Schaffer. Life history consequences of natural selection: Cole's result revisited. *American Naturalist*, 107:791–793, 1973.

E. L. Charnov, G. H. Orians, and K. Hyatt. Ecological implications of resource depression. *American Naturalist*, 110(972):247–259, March 1976.

L. C. Cole. The population consequences of life history phenomena. *The Quarterly Review of Biology*, 29(2):103–137, Jun 1954.

J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2719–2724, May 1992.

D. A. Cristol and P. V. Switzer. Avian prey-dropping behavior. II. american crows and walnuts. *Behavioral Ecology*, 10(3):220–226, 1999.

T. S. Dahl, M. J. Matarić, and G. S. Sukhatme. Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6):674–687, 2009.

É. Danchin, L.-A. Giraldeau, and F. Cézilly, editors. *Behavioural Ecology*. Oxford University Press, 2008.

S. D. Fretwell. *Populations in a seasonal environment*. Princeton University Press, 1972.

B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-Robot Systems*, 18(5):758–768, 2002.

J. A. Gibb. Predation by tits and squirrels on the eucosmid *Ernarmonia conicolana*. *Anim. Ecol.*, 27:375–396, 1958.

J. F. Gilliam, R. F. Green, and N. E. Person. The fallacy of the traffic policeman: a response to Templeton and Lawlor. *American Naturalist*, 119(9):875–878, 1982.

L. Green and J. Myerson. Exponential versus hyperbolic discounting of delayed outcomes: Risk and waiting time. *American Zoologist*, 36(4):496–505, 1996.

L. Green, E. Fisher, S. Perlow, and L. Sherman. Preference reversal and self-control: choice as a function of reward amount and delay. *Behaviour Analysis Letters*, 1:244–256, 1981.

R. F. Green. Stopping rules for optimal foragers. *The American Naturalist*, 123(1):30–43, January 1984.

D. Gross. *Fundamentals of queueing theory*. Wiley, 2008.

E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, July 2008.

Y. Hada and S. Yuta. Robust navigation and battery re-charging system for long term activity of autonomous mobile robot. In *Proceedings of the 9th International Conference on Advanced Robotics*, pages 297–302, October 1999.

I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1996.

R. J. Herrnstein. Relative and absolute strength of responses as a function of frequency of reinforcement. *Journal of the Experimental Analysis of Behaviour*, 4:267–272, 1961.

R. J. Herrnstein. *The matching law: Papers in psychology and economics*. Harvard University Press, 1997. ISBN 0674064593. BF 319.5 R4 H47 1997.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1992.

O. Holland and C. Melhuish. Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5(2):173–202, 1999.

C. S. Holling. Some characteristics of simple types of predation and parasitism. *Canadian Entomologist*, 91:385–398, 1959.

A. I. Houston and J. M. McNamara. *Models of Adaptive Behaviour*. Cambridge University Press, 1999.

I. Ieropoulos, J. Greenman, and C. Melhuish. Imitating metabolism: Energy autonomy in biologically inspired robotics. In *Proceedings of the Second International Symposium on Imitation in Animals and Artifacts*, pages 191–194, Aberystwyth, Wales,, 2003.

A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

C. V. Jones and M. J. Matarić. Adaptive division of labor in large-scale minimalist multi-robot systems. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1969–1974, Oct 2003.

A. Kacelnik. *Characterizing human psychological adaptations*, chapter Normative and descriptive model of decision making: time discounting and risk sensitivity, pages 51–70. Number 208 in Ciba Foundation Symposium. Wiley, 1997.

I. Kelly and C. Melhuish. Slugbot: A robot predator. In *The 6th European Conference On Artificial Life*, University of Economics, Prague, Czech Republic, September 2001.

J. Krebs, J. Ryan, and E. Charnov. Hunting by expectation or optimal foraging: A study of patch use by chickadees. *Animal Behaviour*, 22:953–964, 1974.

J. R. Krebs. *Perspectives in Ethology*, volume 1, chapter Behavioral aspects of predation, pages 73–111. Plenum New York, 1973.

M. J. B. Krieger and J.-B. Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84, 2000.

T. H. Labella, M. Dorigo, and J.-L. Deneubourg. Self-organised task allocation in a group of robots. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, pages 371–380, 2004.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

A. Lein and R. T. Vaughan. Adaptive multi-robot bucket brigade foraging. In *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, Artificial Life*, August 2008.

A. Lein and R. T. Vaughan. Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 601–606, 2009.

K. Lerman and A. Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

S. L. Lima. Downy woodpecker foraging behavior: Efficient sampling in simple stochastic environments. *Ecology*, 65(1):166–174, 1984.

Y. Litus, R. T. Vaughan, and P. Zebrowski. The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 27–32, Roma, Italy, April 2007.

W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305, 2007.

P. Maes. How to do the right thing. Technical Report NE 43 - 836, AI-Laboratory, MIT, 1989.

M. Mangel and C. W. Clark. *Dynamic Modeling in Behavioral Ecology*. Princeton University Press, 1988.

M. J. Matarić and D. Cliff. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1):67–83, November 1996.

D. J. McFarland. Decision making in animals. *Nature*, 269:15–21, September 1977.

D. J. McFarland. Animal robotics - from self-sufficiency to autonomy. In *Proceedings From Perception to Action, IEEE Computer Society Press*, pages 47–54, Lausanne, Switzerland, September 1994a.

D. J. McFarland. Towards robot cooperation. In *Proc. of the Third International Conference on Simulation of Adaptive Behavior*, pages 440–444, 1994b.

D. J. McFarland and T. Bösser. *Intelligent Behavior in Animals and Robots*. The MIT Press, September 1993. ISBN 0262132931.

D. J. McFarland and A. I. Houston. *Quantitative Ethology - The state space approach*. Pitman Advanced Pub. Program, Boston, USA, 1981.

D. J. McFarland and E. Spier. Basic cycles, utility and opportunism in self-sufficient robots. *Robotics and Autonomous Systems*, 20:179–190, June 1997.

J. M. McNamara. Optimal patch use in a stochastic environment. *Theoretical Population Biology*, 21(2):269–288, 1982.

F. Michaud and E. Robichaud. Sharing charging stations for long-term activity of autonomous robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2746–2751, 2002.

J. Minguez and L. Montano. Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.

M. Monemerlo and S. Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Springer, 2007.

J. Nakahara and M. Toda. Optimal strategies and human behavior in fungus-eater game 4. Technical Report ESD-TDR-64-237, Decision Sciences Laboratory US Air Force, L.G. Hanscom Field, Bedford, Massachusetts, February 1964.

N. Nilsson. Shakey the robot. Technical Report Technical Note 325, SRI International, Menlo Park, CA, 1984.

E. Østergaard, G. S. Sukhatme, and M. J. Matarić. Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the International Conference on Autonomous Agents*, pages 29–30, May 2001.

G. Parker and R. Zbeda. Learning navigation for recharging a self-sufficient colony robot. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 734–740, October 2007.

R. Pfeifer. The fungus eater approach to emotion: A view from artificial intelligence. In *In Cognitive Studies: Bulletin of the Japanese Cognitive Science Society*, volume 1, pages 42–57, 1994.

R. Pfeifer. Building fungus eaters: Design principles of autonomous agents. In *From Animals to Animats, Cambridge, MA: MIT Press*, volume 4, 1996.

R. Pfeifer and G. Gómez. Interacting with the real world: design principles for intelligent systems. *Artificial Life and Robotics*, 9(1):1–6, April 2005.

R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press, 2001.

R. Pfeifer and P. Verschure. Distributed adaptive control: a paradigm for designing autonomous agents. In *Proceedings of the First European Conference on Artificial Life*, pages 21–30. MIT Press, 1992.

P. Pirjanian. Behavior coordination mechanisms - state-of-the-art. Technical report, University of Southern California, Institute for Robotics and Intelligent Systems Technical Report IRIS-99-375, 1999.

A. Raineri and H. Rachlin. The effect of temporal constraints on the value of money and other commodities. *J. Behav. Decis. Making*, 6:77–94, 1993.

A. Sadat and R. T. Vaughan. Blinkered lost: Restricting sensor field of view can improve scalability in emergent multi-robot trail following. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 947–952, 2010.

T. W. Schoener. Theory of feeding strategies. *Annual Reviews in Ecology and Systematics*, 2:369–404, 1971.

N. Schweighofer, K. Shishida, C. E. Han, Y. Okamoto, S. C. Tanaka, S. Yamawaki, and K. Doya. Humans can adopt optimal discounting strategy under real-time constraints. *PLoS Computational Biology*, 2(11):e152, 2006.

F. Sempe, A. Munoz, and A. Drogoul. Autonomous robots sharing a charging station with no communication: a case study. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS'02)*, volume 5, pages 91–100, 2002.

A. K. Seth. *On the relations between behaviour, mechanism, and environment: Explorations in artificial evolution*. PhD thesis, University of Sussex, 2000.

A. K. Seth. Competitive foraging, decision making, and the ecological rationality of the matching law. In *From Animals to Animats 7, 7th International Conference on Simulation of Adaptive Behavior, SAB*, pages 359–368, Cambridge, MA, USA, 2002. MIT Press.

A. K. Seth. The ecology of action selection: Insights from artificial life. *Philos. Trans. R. Soc. B*, April 2007.

C. E. Shannon. *Collected Papers*. IEEE Press, New York, 1993.

D. A. Shell and M. J. Matarić. On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2723, October 2006.

R. Sibly and D. J. McFarland. On the fitness of behavior sequences. *The American Naturalist*, 110(974):601–617, 1976.

M. Silverman, D. M. Nies, B. Jung, and G. S. Sukhatme. Staying alive: A docking station for autonomous robot recharging. In *IEEE International Conference on Robotics and Automation*, pages 1050–1055, Washington D.C., May 2002.

M. Silverman, B. Jung, D. M. Nies, and G. S. Sukhatme. Staying alive longer: Autonomous robot recharging put to the test. Cres-03-015, Center for Robotics and Embedded Systems, University of Southern California, 2003.

E. Spier. *Artificial Ethology*, chapter Robotic Experiments on rat instrumental learning. Oxford University Press, 2001.

E. Spier and D. J. McFarland. A finer-grained motivational model of behaviour sequencing. In *Proceedings of fourth International Conference on Simulation of Adaptive Behavior*, pages 255–263, 1996.

E. Spier and D. J. McFarland. Possibly optimal decision-making under self-sufficiency and autonomy. *Journal of Theoretical Biology*, 189(3):317–331, December 1997.

E. Spier and D. J. McFarland. Learning to do without cognition. In *From Animals to Animats 5, 5th International Conference on Simulation of Adaptive Behavior, SAB*, pages 38–47, 1998.

L. Steels. A case study in the behavior-oriented design of autonomous agents. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 445–452, Cambridge, MA, USA, 1994. MIT Press. ISBN 0-262-53122-4.

L. Steels. Discovering the competitors. *Journal of Adaptive Behavior*, 4(2):173–199, 1996.

L. Steels. A selectionist mechanism for autonomous behavior acquisition. *Robotics and Autonomous Systems*, 20(2-4):117–131, June 1997.

D. W. Stephens. The logic of risk-sensitive foraging preferences. *Animal Behaviour*, 29: 628–629, 1981.

D. W. Stephens. *Stochasticity in foraging theory: risk and information*. PhD thesis, Oxford University, 1982.

D. W. Stephens. Discrimination, discounting and impulsivity: A role for an informational constraint. *Philosophical Transactions: Biological Sciences*, 357(1427):1527–1537, November 2002.

D. W. Stephens and J. R. Krebs. *Foraging Theory*. Princeton University Press, 1986. ISBN 0-691-08441-6.

D. W. Stephens, B. Kerr, and E. Fernandez-Juricic. Impulsiveness without discounting: the ecological rationality hypothesis. *Proceedings of the Royal Society of London (B)*, 271: 2459–2465, 2004.

D. W. Stephens, J. S. Brown, and R. C. Ydenberg, editors. *Foraging - Behavior and Ecology*. University of Chicago Press, 2007.

S. Stidham. *Optimal Design of Queueing Systems*. CRC Press, 2009.

R. J. Stuart and N. Peter. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron. The design and development of a wireless robotic networked aquatic microbial observing system. *Environmental Engineering Science*, 24(2):205–215, 2006.

Z. Sun and J. Reif. On energy-minimizing paths on terrains for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 3782–3788, September 2003.

R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. MIT Press, 1998.

P. V. Switzer and D. A. Cristol. Avian prey-dropping behavior. I. the effects of prey characteristics and prey loss. *Behavioral Ecology*, 10(3):213–219, 1999.

H. A. Taha. Queueing theory in practice. *Interfaces*, 11(1):43–49, 1981.

A. R. Templeton and L. R. Lawlor. The fallacy of the averages in ecological optimization theory. *The American Naturalist*, 117(3):390–393.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

F. Thuijsman, B. Peleg, M. Amitai, and A. Shmida. Automata, matching and foraging behavior of bees. *Journal of Theoretical Biology*, 175(3):305–316, August 1995.

M. Toda. Design of a Fungus-Eater. *Behavioral Science*, 7:164–183, 1962.

M. Toda. *Man, Robot and Society*. Martinus Nijhoff Publishing, 1982. ISBN 089838060X.

P. Tompkions, A. Stentz, and W. R. L. Whittaker. Mission planning for the sun-synchronous navigation field experiment. In *Proc. of the IEEE international Conference on Robotics and Automation*, May 2002.

P. Ulam and T. Balch. Using optimal foraging models to evaluate learned robotic foraging behavior. *Adaptive Behavior*, 12(3-4):213–222, 2004.

I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1572–1577, May 1998.

N. M. van Dijk. To pool or not to pool? "the benefits of combining queueing and simulation". In *Proceedings for Winter Simulation Conference*, pages 1469–1472, 2002.

H. R. Varian. *Microeconomic Analysis*. W. W. Norton, 1992.

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

R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. Go ahead, make my day: Robot conflict resolution by aggressive competition. In *From Animals to Animats 6, 6th International Conference on Simulation of Adaptive Behavior, SAB*, pages 491–500, August 2000a.

R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. Whistling in the dark: cooperative trail following in uncertain localization space. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 187–194, Barcelona, Spain, June 2000b.

J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 437–448, October 2005.

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

R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1): 1–18, 2002.

J. Wawerla and R. T. Vaughan. Near-optimal mobile robot recharging with the rate-maximizing forager. In *European Conference On Artificial Life (ECAL)*, pages 776–785, Lisbon, Portugal, September 2007.

J. Wawerla and R. T. Vaughan. Optimal robot recharging strategies for time discounted labour. In *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, Artificial Life*, August 2008.

J. Wawerla and R. T. Vaughan. Robot task switching under diminishing returns. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5033–5038, 2009.

J. Wawerla and R. T. Vaughan. A fast and frugal method for team-task allocation in a multi-robot transportation system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1432–1437, Anchorage, AK, May 3 - 8 2010.

T. Wehrle. New fungus eater experiments. In P. Gaussier and J.-D. Nicoud, editors, *From perception to action*, pages 400–403, Los Alamitos, 1994. IEEE Computer Society Press.

D. Wettergreen, B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M. D. Wagner, and W. R. L. Whittaker. First experiment in sun-synchronous exploration. In *International Conference on Robotics and Automation*, pages 3501–3507, May 2002.

D. Wettergreen, P. Tompkins, C. Urmson, M. Wagner, and W. Whittaker. Sun-synchronous robotic exploration: Technical description and field experimentation. *The International Journal of Robotics Research*, 24(1):3–30, January 2005.

Wikipedia. Mars Exploration Rover - Wikipedia, The Free Encyclopedia, 2008. `http://en.wikipedia.org/wiki/Mars_Exploration_Rover` [Online; accessed 8-May-2008].

Wikipedia. Viking program - Wikipedia, The Free Encyclopedia, 2010. `http://en.wikipedia.org/wiki/Viking_program` [Online; accessed 18-Jan-2010].

R. C. Ydenberg and W. E. Davies. Resource geometry and provisioning routines. Unpublished Manuscript, 2010.

B. Zhang and G. S. Sukhatme. Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network. In *IEEE International Conference on Robotics and Automation*, pages 3673–3680, 2007.

M. Zuluaga and R. Vaughan. 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, August 2005.