Robotic Team Navigation in Complex Environments Using Stigmergic Clues

A thesis presented

by

Alfred Wurr

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science

The University of Manitoba

Winnipeg, Manitoba

July 2003

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES
*****
COPYRIGHT PERMISSION PAGE

Robotic Team Navigation in Complex
Environments Using Stigmergic Clues

BY

Alfred Wurr

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University

of Manitoba in partial fulfillment of the requirements of the degree

of

Master of Science

Alfred Wurr © 2003

Thesis advisor

John Anderson

Author

Alfred Wurr

Robotic Team Navigation in Complex Environments Using Stigmergic Clues

Abstract

Robotic agents in dynamic environments must regularly navigate reactively, relying solely on their immediate local perceptions to make navigation decisions. In environments with complex topography, features such as terrain undulation, geometrically complex barriers and similar obstacles form local maxima and minima that can trap and hinder agents navigating reactively. When agents are operating in groups, navigation is even more difficult, requiring agents to explicitly communicate to plan and coordinate their actions. The cost of explicit communication can be substantial, however, making it desirable to avoid its use in many domains.

Accordingly, in this thesis I present methods to allow a team of agents using reactive navigation to assist one another in their explorations through implicit communication. Agents accomplish this by modifying their environment in a manner meaningful to other agents in order to share knowledge about advantageous locations. The effectiveness of the various methods presented is shown in a number of experiments in which a team of agents is required to locate a goal situated somewhere in an unknown and complex simulated indoor environment.

i

Contents

Acknowledgments

Dedicated to my parents, Alfred and Beverley, for their love, encouragement and support.

Chapter 1

Introduction

The focus of this thesis is to increase the effectiveness of a team of robotic agents using reactive navigation - navigating using only the current perceptions available to the robot - to explore complex and dynamic environments. Robotic agents employing reactive navigation suffer from several problems. Local maxima and minima and a lack of goal-directed ability are among the more common drawbacks. Agents that are only reacting may be attracted to a local stimulus, but one that is really only a waypoint on the route to a goal - a local maximum. On the other hand, these same agents may attempt to move in a direct line to a visible goal, but be prevented from reaching it by an interposing barrier - a local minimum. These problems (explored more fully in Chapters 2 and 3) can hinder the explorations of agents using reactive navigation exclusively, preventing them from finding their goals in a timely fashion. To solve these problems, I explore how agents can help each other navigate by identifying desirable locations and sharing this knowledge with each other by modifying the environment.

This chapter begins with an introduction of terminology to establish a common vocabulary for later discussion. This is followed by a description and discussion of the research questions being studied and my approach to answering them. The chapter concludes by describing the structure of the remainder of the thesis.

1.1 Terminology

Before the topic and issues of this thesis can be discussed meaningfully, it is necessary to first define and describe some of the terminology used throughout this work.

A natural starting point is to first define the term *agent*. An agent in this work refers to a software construct situated in an environment in which it is capable of acting autonomously in pursuit of its goals [Russell and Norvig, 1995; Weiss, 1999]. This collection of software instructions is embedded in the body of either a physical or simulated robot. In the latter case, agents are also referred to as *softbots* to emphasize their non-physical nature. In either case, however, an agent is embodied in the sense that it occupies a space in an environment and affects that environment. Embodied agents consist of sensors used to observe their world and actuators (or effectors) manipulated to carry out actions within it, in addition to their cognitive components. The terms *agent* and *robot* are used somewhat interchangeably in this work (and much other work in Artificial Intelligence).

Embodied agents are mobile by nature, needing to move about their environment to locate and accomplish their objectives. For these agents, goals include interacting with or changing the state of other entities (physical objects or other agents) located

somewhere in the environment, and specific locations may also be associated with goals.

A *complex* environment is one in which the physical features of the domain are geometrically intricate and/or irregular in outline. This includes outdoor environments (e.g. forests, or rocky and hilly regions) and building interiors. Similarly, a *dynamic* environment is a domain that is subject to frequent changes in the position of objects and entities within it and possibly even changes in layout and structure.

The term *world model* is another term used frequently in this work. A world model refers to an internal map using some chosen representation that can be used by an agent to solve problems in the domain.

While additional terminology will be introduced throughout this thesis, these provide the basic definitions needed to understand the remainder of this work.

1.2 Motivation

Robotic agents are being used with greater frequency for specialized tasks such as rescue or security-related jobs (e.g. hostage situations) in dangerous and unpredictable settings [Baltes and Anderson, 2003]. Simulated robotic agents are also employed as characters in the dynamic and complex simulated worlds of computer games. When faced with an unknown environment or when using possibly out-dated or inaccurate maps (such as after a disaster), an embodied agent must navigate relying heavily or solely on local information available via the robot's sensors. Moreover, agents in embodied domains often need to make navigational decisions quickly. As a result, reactive navigation must sometimes be used to augment methods that use

explicit planning to navigate or to replace them entirely. Whether used exclusively or as part of a hybrid approach, improving reactive navigation is therefore desirable.

1.3 Method

My method for improving reactive navigation involves identifying useful locations and drawing attention to these locations by modifying the environment. Precisely what makes a location useful or significant is domain-dependent, but useful locations would include both positive elements (e.g. a goal is visible) as well as potentially negative elements (e.g. marking a cliff edge). By modifying the environment, it becomes possible for navigating agents to perceive a useful location at a distance and move toward it (or similarly avoid an undesirable location).

The marking of obstacles and helpful elements is commonly employed in human settings, such as modern highways where road signs and lines of the road provide drivers with information about what lies ahead. In a multi-agent setting, this approach allows benefits acquired by the exploration of one agent to be disseminated to others. While taking advantage of such distributed interactions in order to improve the performance of a collective is a common theme in multi-agent systems, most approaches to this require extensive explicit communication (see Chapter 2). The work presented here explores the improvement of navigation methods without the use of explicit communication, through the dropping of visible objects to serve as markers.

Using variations of this basic concept, I demonstrate methods that a team of agents using reactive navigation can employ in order to accelerate their search process. I illustrate methods to avoid local minima and maxima, as well as a method that allows

a goal, once discovered, to be located by others more quickly on subsequent trips. This is accomplished without the use of explicit communication between agents, and with no use of elaborate internal world models or high-level navigational planning mechanisms. These methods are also collaborative in nature - it is not simply one agent solely marking elements for the benefit of all colleagues, but a collaborative process between many individuals. Chapter 3 provides an in-depth description of the techniques that I employ to achieve these results.

1.4 Research Questions

This thesis addresses the following research questions:

- Will identifying locations of high utility without the use of explicit communication improve the exploration performance of a team of reactively navigating agents in an unknown, complex and dynamic environment?

- Will identifying locations in this manner allow reactively navigating agents to escape highly enclosed or complicated areas more readily?

- Will identifying locations in this manner suffice to draw agents to open areas of the environment and cause them to explore an area and locate a goal faster?

- Will cooperatively constructing a minimal trail of markers to preserve and reuse knowledge about the location of a discovered goal allow a team of reactive agents to more readily locate the goal on subsequent trips?

These research questions are examined in specific settings using a computer game environment as a simulator (described in detail in Chapters 3 and 4).

1.5 Summary

In this chapter I have defined terminology essential to discussing this work, introduced reactive navigation, including its problems and applications and the fundamental research questions with which this research is concerned. Having introduced the vital aspects of this work, I conclude this chapter by outlining the composition of the remainder of this document.

1.6 Thesis organization

This thesis is structured as follows:

- Related Literature

- Stigmergic Navigation

- Implementation

- Experimentation

- Conclusion

Chapter 2: Related Literature

Presents background and discussion of previous work in areas related to this thesis to give the reader an understanding of the issues involved. This includes an

overview of agent architectures, multi-agent systems, navigation, communication and stigmergy. The chapter concludes with a discussion of using computer games as a test-bed for exploring issues in artificial intelligence (AI), including those covered in this work.

Chapter 3: Stigmergic Navigation

This chapter describes the approaches that I use to deal with some of the problems associated with reactive navigation.

Chapter 4: Implementation

Chapter 4 outlines the specific implementation used to test and evaluate the efficacy of these methods. This includes a description of the agents and their environment.

Chapter 5: Experimentation

Chapter 5 presents the results of the experiments executed using the approaches described in Chapter 3 and an analysis of the results.

Chapter 6: Conclusion

Chapter 6 discusses the experimental results and conclusions that can be drawn from them. It also outlines possibilities for future work in this area that can further extend the ideas and methods contained in this work.

Chapter 2

Related Literature

This chapter provides an overview of research related to this thesis. As this work draws on many areas, it is necessary to present the various topics outward from the fundamentals. To this end, the reader is first presented with an overview of agent control architectures. This leads into a discussion of multi-agent systems with an emphasis on navigation in embodied domains. Following this is an overview of inter-agent communication, comparing and contrasting explicit and implicit communication mechanisms. The chapter concludes with a discussion of computer games in Artificial Intelligence research, including a survey of researchers that have used computer games to explore a variety of issues.

2.1 Agent Control Architectures

This section presents a review and discussion of agent control architectures including logic-based, behaviour-based and hybrid agents to give the reader an under-

standing of their strengths, weaknesses, and suitability to particular domains and applications.

2.1.1 Logic-based Agents

Traditionally, agents have been designed using a symbolic approach [Brooks, 1991a]. In this approach agents use logical reasoning and maintain an explicit symbolic representation of the world (usually referred to as a world model). Agents of this type can be programmed to reason about complex ideas and concepts at a high level of abstraction. In this paradigm, robot control is divided into sense, plan and act phases, with each phase handled respectively by sensing, planning and execution systems [Gat, 1997]. With each iteration of these phases, the sensing system examines the environment and updates an internally maintained world model accordingly. The planning system uses this information together with the agent's internal state to decide on a course of action that is designed to achieve the agent's goals [Weiss, 1999]. The execution system then carries out the actions specified by the planning system using the agent's effectors [Gat, 1997]. The cycle is repeated at regular and very small time intervals throughout the agent's operational period.

The advantage of agents that use this control approach is that they can plan actions to achieve their goals well into the future and utilize knowledge gained from past experiences when making decisions. As a result, logic-based agents can be adept at proactively working toward their goals by formulating plans to achieve them. Unfortunately, many of the embodied environments where agents might be used (certainly most real world environments) demand that agents make decisions with alacrity due

to a high rate of environmental change. Consequently, agents do not have the luxury of unduly pondering their current situation to achieve an ideal plan before acting on it [Weiss, 1999]. To make matters worse, the size of the state-space that must be searched in order to come up with an optimal or even reasonable plan in a complex domain can be prohibitively large. Therefore, it is likely that plans will be outdated before they are ready to be implemented (or during the course of their execution), requiring re-planning. In real-world environments, this can be aggravated by inaccurate sensory input, leading to greater deviations between the agent's view of the world and reality. As domains become more and more dynamic, these problems become more severe, eventually causing such agents based on a sense/plan/act cycle to become impractical [Brooks, 1991a].

2.1.2 Behaviour-based Agents

The limitations of symbolic methods inspired more reactive approaches to agent control in the form of behaviour-based or reactive agents [Brooks, 1986; Kube and Bonabeau, 1998]. The subsumption architecture proposed by Brooks [1986] is an early example of a behaviour-based control architecture [Gat, 1997], representing a radical departure from the logic-based approach. Rather than explicitly planning and world modelling, subsumption-based agents simply re-observe the world at every computational step and act on what they perceive at that instant. By doing so, discrepancies between the agent's view of the world and reality are limited (sensory inaccuracies notwithstanding) [Werger, 1999; Weiss, 1999]. As a consequence, even in the presence of sensory error, the effects of such errors are no longer cumulative.

That is, whether an agent saw something incorrectly last cycle does not directly affect the agent's options in the current cycle, since it is no longer working with the earlier erroneous information.

A subsumption-based agent's decision-making function is assembled from a number of simple behaviours that are triggered in response to sensory input. Each behaviour is represented as a separate layer of control designed to accomplish particular tasks, such as obstacle avoidance or movement toward a goal [Arkin, 1998]. Behaviours operate concurrently and asynchronously and receive sensory input directly from the agent's sensors. Behaviours react to sensory input and accomplish tasks by sending motor control signals to the agent's effectors [Arkin, 1998; Weiss, 1999]. Behaviour coordination is achieved by allowing higher priority layers to subsume lower priority layers when several fire simultaneously [Brooks, 1986]. This involves either suppressing sensory input signals before they reach a behaviour or by preventing control signals transmitted by the behaviour from reaching the effectors [Arkin, 1998].

The layering of behaviours allows an agent's control structure to be evolved incrementally. As more complicated layers are added, they are given higher priority, allowing them to subsume or inhibit the simpler behaviours that preceded them. The pre-existing behaviours do not require and do not have an awareness of the new behaviours. As a result, they do not require modification as a result of the addition of any new behaviours [Arkin, 1998]. Communication between behaviours is strongly discouraged, except through interaction with the environment [Brooks, 1991a; Arkin, 1998]. The simple priority-based behaviour arbitration and lack of complex planning allow agents to react quickly to situations that demand it.

Despite their profound impact upon the direction of autonomous agent research [Arkin, 1998], subsumption-based agents possess significant limitations [Balch and Arkin, 1993]. For one, the control system is hardwired into the agent's behavioural structure and inter-relationships. In many situations a lower-priority layer may in fact be more important than a higher layer, but only a single behaviour can take control at any given moment [Arkin, 1998]. As a result, alterations to the control system can require significant redesign of the entire system [Arkin, 1998; Moorman and Ram, 1992]. Scalability is another problem. As system complexity grows, the number of behaviours can increase dramatically and coordinating any significant number becomes problematic [Pirjanian, 1999; Brooks, 1990; Tsotsos, 1995]. Also, since these agents do not maintain state, they are unable to exploit previous experiences or learn from past mistakes [Moorman and Ram, 1992]. As a result, purely reactive agents can oscillate between different stimuli, preventing them from achieving their goals in a timely fashion [Balch and Arkin, 1993]. Finally, since the individual layers do not explicitly cooperate, they cannot work together to accomplish complex tasks (such as by sequencing behaviours).

As a result of the subsumption architecture's drawbacks, many other behaviour-based architectures followed in its wake, attempting to improve on the template laid down by Brooks. Among these new approaches was Ronald Arkin's *schema-based* architecture [Arkin, 1998]. This approach sought to allow multiple behaviours to influence an agent's actions simultaneously by utilizing the concept of *potential fields* (or superposition) [Pirjanian, 1999; Parunak et al., 2001]. In a potential field-based approach, an agent's attraction to or repulsion from an object can be given a value

at any particular location in the environment based on perceptions obtained from that location. The potential fields of a set of objects (e.g. goals and obstacles) can be combined into an overall potential field for the agent's surroundings. This allows the agent to follow a gradient across the field, avoiding obstacles and being attracted to goals. Potential fields are most commonly viewed graphically as a physical map (a potential field map, e.g. [Arkin and Balch, 1998]) showing the direction an agent would go in at any spot, given the combination of attractions and repulsions. However, a potential field can be represented as a function, a collection of rules, or any other representation that allows fields to be combined.

Schema-based behaviours divide control into *perceptual schemas* and *motor schemas* that are tailored to specific aspects of an agent's environment [Balch and Arkin, 1998, 1997, 1995; Pirjanian, 1999]. Perceptual schemas are associated with motor schemas, and are responsible for sensing specific aspects of the world and providing motor schemas with relevant stimuli. Perceptual schemas can themselves be used by other perceptual schemas to form more complex perceptual schemas and provide higher-level interpretations of the environment [Arkin, 1998]. Motor schemas, by contrast, are responsible for operating the agent's actuators to achieve these goals. On every computation cycle, each active schema computes an ideal movement vector based on its particular interests and input from relevant perceptual schemas. Together these vectors form a potential field that captures the agent's positive and negative attraction to objects and entities within sensory range from its current position in the environment [Arkin, 1998]. To proceed to a goal while avoiding collisions, the agent follows the path (gradient) of steepest descent by summing and normalizing

the vectors to compute a final heading and speed of travel.

Complex tasks are accomplished by grouping related behaviours together into collections (known as *assemblages*) and activating them in an appropriate sequence or by combining their outputs together [Mataric, 1997; Balch and Arkin, 1997]. For example, Mataric [1997] achieves flocking behaviour among a team of agents by summing the outputs of three behaviours (*safe-wandering*, *aggregation* and *dispersion*). Each assemblage is tailored to handle specific situations that the agent is likely to encounter in its lifetime. The influence of each of the individual behaviours that make up an assemblage is set by a gain value. This value is set according to each behaviour's relative importance, and can also be used to remove a behaviour's influence altogether when it is not appropriate to a particular state [Arkin, 1998].

Many other behaviour coordination mechanisms have also been used, such as voting and winner-take-all behaviour arbitration. In voting methods, each active behaviour votes for and against each of the various possible actions and the action with the highest weighted sum of votes is chosen and executed. Winner-take-all arbitration, by contrast, selects a single behaviour from the set of available behaviours and empowers the chosen behaviour to take whatever actions it deems necessary. Though many other mechanisms exist [Pirjanian, 1999], most have in common the ability to allow multiple behaviours to simultaneously influence an agent's actions [Mataric, 1997]. Pirjanian [1999] provides an in-depth survey and review of the many behaviour arbitration mechanisms for the interested reader.

In addition to allowing for greater flexibility in behaviour arbitration, more recent behaviour-based approaches relax many of the philosophical restrictions of subsumption-

based (and other *purely* reactive) agents. This includes endowing agents with greater representational power to permit them to carry out more complex tasks and achieve higher goals [Mataric, 1997]. This is accomplished by retaining *limited* information about the world and the agent's past experiences across computational cycles within the behaviours themselves [Balch and Arkin, 1993]. Balch and Arkin [1993], for example, avoid redundant exploration in a schema-based agent (discussed in Section 2.3) by recording visited locations in a spatial grid and avoiding them.

Although adept at *reacting* to situations, behaviour-based agents are not as well-suited to taking proactive action to achieve their goals. This is especially true of purely reactive agents because they do not reason or comprehend their surroundings at a high level, but rather respond automatically to their current situation [Gat, 1997]. It is also difficult to achieve even for behaviour-based agents that employ world models and retain state, though somewhat less so [Mataric, 1997; Arkin and Balch, 1997].

2.1.3 Hybrid Agents

In order to allow agents to be simultaneously reactive and proactive, agents have also been implemented using hybrid architectures. Hybrid agents blend a reactive subsystem that is responsible for the agent's short-term safety and a deliberative subsystem that performs higher level planning of long-term goals [Arkin and Balch, 1997].

Hybrid agents are typically composed of a planning module, reactive module and a control module that arbitrates and integrates the other two [Gat, 1997]. The planner

produces plans that are executed in the form of suggestions to the reactive layer, allowing the agent to perform complex and time-extended tasks. The reactive module attempts to follow the suggestions of the planner, while ensuring the agent's safety by reacting to events that require an immediate response [Gat, 1997]. The middle layer is responsible for mediating between the deliberative and reactive layers. It must deal with the differing representations and harmonize conflicting commands issued by each layer.

The central problem or challenge in this approach is achieving an optimal, or at least acceptable, balance between reactive and proactive behaviour [Weiss, 1999]. Agents that are too reactive are at the mercy of their impulses and the limitations of their sensor range. Conversely, an agent that is too deliberative may be too slow to respond to changing circumstances.

While hybrid approaches are a viable and proven alternative [Arkin and Balch, 1997; Gat, 1997; Reece and Kraus, 2000; Bonasso et al., 1997], hybrid agents are anything but parsimonious. Many researchers [Balch and Arkin, 1993, 1997; Werger, 1999] believe the limits of reactive and behaviour-based agent control can be pushed further before resorting to traditional symbolic reasoning and planning techniques. This research takes a similar view.

Regardless of the specific control methodology that agents employ, the difficulties in single agent control are multiplied when autonomous agents are working together as a team. Section 2.2, discusses the challenges and motivations for multi-agent systems in this regard.

2.2   Multi-agent systems / Teamwork

As problem scale and complexity increase, a single-agent solution to many problems quickly becomes impractical. Security patrolling, disaster rescue, and even much lower-level support tasks such as simple mapping of unknown terrain generally call for the use of a team of agents for reasons of redundancy, adequate coverage, efficiency, and specialization of expertise [Brooks, 1991a; Resnick, 1998]. In the event that a task requires extensive expertise or abilities in various areas, constructing a single robot able to perform all the necessary sub-tasks results in an overly complex individual that is difficult and costly to build and maintain [Brooks, 1991a]. In such a situation, a preferable alternative is to decompose a large or complex problem into many simpler sub-tasks that individuals with different abilities can work on separately as part of a larger team. Since these problems are much simpler than the overall global task, agent complexity can be reduced substantially [Arkin and Balch, 1998]. In addition to distributing expertise, using a group of agents allows us to exploit the inherent parallelism of naturally distributed tasks such as search and rescue or foraging, allowing global goals to be achieved many times faster [Burgard et al., 2000].

The degree of coordination and cooperation among a team of agents spans a broad range. At its most basic, teamwork can be incidental: agents simply occupy the same environment and share common goals with no explicit cooperation [Nwana et al., 1996]. Agents' actions can be moderately more coordinated, where they perceive one another and may cooperate, but do not *require* each other's help to complete tasks [Nwana et al., 1996]. This loosely coupled cooperation is suitable primarily for tasks such as foraging, where it is theoretically possible for a single agent to complete the

task on its own given enough time. The challenge in this setting is avoiding redundant effort and keeping agents from impeding each other (lest the additional agents worsen performance) [Buck et al., 2002]. The degree of cooperation falls into a spectrum, up to a point where each agent performs an explicit role in an overall scheme for achieving the system's goals [Nwana et al., 1996]. Here agents explicitly share goals and plans, communicate with one another and synchronize their actions in order to achieve improved overall system performance [Tambe et al., 1999; Veloso and Stone, 1998].

In order to achieve the necessary level of cooperative or global behaviour a varying degree of centralized versus decentralized control is required. Isolating decision-making and planning in a centralized controller lends itself well to highly coordinated group behaviour. However, it can also create a potential bottleneck as the load on a central controller grows. Consequently, it tends to scale badly as group size increases [Mataric, 1997]. This is partly because the communication between agents and the central controller also grows as the number of agents increases. Perhaps worse, if control is completely centralized, failure of the central controller implies a breakdown of the entire system [Kube and Bonabeau, 1998]. Beyond the issue of load on a central controller, the use of centrally created plans for a group also fares poorly as problems become more difficult. The additional complexity of planning for a team of agents slows the planning process and as a result, a centrally calculated plan for a large group is even more likely to be unusable by the time it is completed due to changes in the environment [Parunak et al., 2001].

The drawbacks of centralized control can be alleviated by allowing agents to make

decisions locally [Durfee, 1999]. Doing so allows the central controller to focus more on global performance of the team, leaving individuals to handle the details. By doing so it becomes possible for agents to function for varying periods of time without being in constant contact with a group controller or their peers [Stone and Veloso, 1999]. This has the added advantage of reducing communication overhead as well. The resulting multi-agent system is more robust and scalable as a result [Kube and Bonabeau, 1998]. The precise degree of centralized versus decentralized control is a design decision that is driven by the domain in which the multi-agent system is to be employed.

Regardless of the control method used, issues such as inter-agent interference, communication efficiency and planning are all of concern [Stone and Veloso, 1999; DesJardins et al., 1999]. As the agent population grows, the probability that an agent will obstruct another's path (in an embodied domain) or interfere with another's work grows. In order to minimize interference, agents can communicate to plan their actions, declare their intentions and collaborate in general. However, communication is slow and its overhead grows with team size. These and other costs can make its drawbacks greater than its benefits [Balch and Arkin, 1994], as will be discussed in Section 2.4.

2.3 Navigation and Embodied Domains

Navigation is an essential component of agents operating in embodied domains, since they need to move and explore their environment in order to achieve their goals [Nehmzow et al., 2000]. The challenge of navigation increases in tandem with the

topographical complexity of the environment. Navigating in a flat open area with uninterrupted sight lines and few obstacles is relatively straightforward. However, many domains that agents may be faced with are not so simplistic. For example, building interiors form complex environments containing many corridors, doorways, walls and obstacles. At its most basic, navigating within buildings involves finding and negotiating relatively narrow doorways and hallways and possibly locating and climbing stairs (or controlling an elevator). Navigation in both indoor and outdoor domains can become much more complex, as agents may be faced with terrain undulation, irregularly shaped obstacles, and winding maze-like layouts that make finding and navigating to a goal problematic.

Intelligent navigation in complex environments typically involves localization and path planning [Baltes and Anderson, 2003]. Localization is the process of determining an agent's position in its environment, while path planning is the process of calculating a collision-free path through the environment from the agent's current position to its desired destination. Path planning requires either pre-existing knowledge about the environment, such as a map and the locations of objects within it [Reece and Kraus, 2000] or the ability to discover this information. Accurate localization is also essential for path planning to be successful, because if the agent is in error about its current location, the path that is calculated to get it to its destination will be likewise incorrect [Verth et al., 2000].

A variety of localization techniques exist, including methods that use odometry, global-positioning, and radio-sonar positioning [Werger and Mataric, 1999]. In odometry-based localization agents begin in a known starting location and contin-

uously update their locations as they travel [Werger and Mataric, 1999]. Global positioning, on the other hand, works by dividing the environment into a grid where locations can be identified by coordinates. Radio-sonar methods triangulate position by sending out sonar pings and calculating position based on differences in ping times. Unfortunately, odometry-based navigation works only over short distances [Nehmzow et al., 2000] as cumulative errors in odometry can quickly result in discrepancies between the agent's actual and perceived location [Brooks, 1991b]. The latter two methods both require preparation of the environment ahead of time and complicated sensory equipment [Werger and Mataric, 1999].

In addition to accurate localization, path planning algorithms require and operate on symbolic representations of the environment. These may be provided *a priori*, constructed at runtime based on an agent's perceptions, or both in combination. Storing, constructing and maintaining these maps can require considerable resources in terms of both space and time [Balch and Arkin, 1993]. For this reason, in practice agents often possess only a simple map based on previous travels (if any map at all), rather than a detailed blueprint of the area they are to navigate [Fu et al., 1996]. To make matters worse, if the environment is dynamic and subject to extensive change, any dynamically constructed map may have only a limited viable lifetime. In such situations, path planning becomes limited in its utility, as the path planner is reduced to planning based only on the agent's immediate perceptions and/or based on a partial map that the agent builds as it navigates. Consequently, embodied agents must often navigate in environments that are partially (if not wholly) unknown for periods of time [Sgorbissa and Arkin, 2001].

Even with an accurate map of the environment and reliable localization, path planning algorithms can take too long to compute a path in a fast changing environment. Plans must be changed or replanned each time anything affecting the planned path changes. These drawbacks are even more severe when agents are planning and acting as a team [Mataric, 1997], because of the greater complexity of group plans and uncertainty about the possible actions of teammates. Agents must also be prepared to adjust to changes in the environment initiated by others. For example, an agent's intended path may be abruptly cut off by objects placed by others or by the bodies of other agents themselves. All of these factors contribute to making systems that use path-planning exclusively impractical in many real world scenarios [Brooks, 1991a].

As a result, local navigation decisions are sometimes better handled using reactive navigation techniques (as opposed to path planning) in unknown and rapidly changing domains [Sgorbissa and Arkin, 2001]. Instead of prescriptively navigating to a desired location via path planning, reactive navigation emerges from the agent's responses to its immediate local perceptions [Mataric, 1997]. This allows agents to react quickly to changing circumstances, does not require a perfect map of the environment, and since agents maintain at most a limited model of the world, localization is often not as crucial [Groner and Anderson, 2001]. Reactive navigation has also been referred to as *local* navigation as the agent, at each computation step, decides on the best direction in which to move to achieve its aims based only on its local perceptions [Sgorbissa and Arkin, 2001].

Unfortunately, reactive control mechanisms such as those based on potential fields are vulnerable to local minima. Local minima situations occur when an agent can

Agent (A) trapped in
a box-canyon with
only a single exit is
unable to perceive
the nearby goal (G)
and must wander
randomly until it
exits.
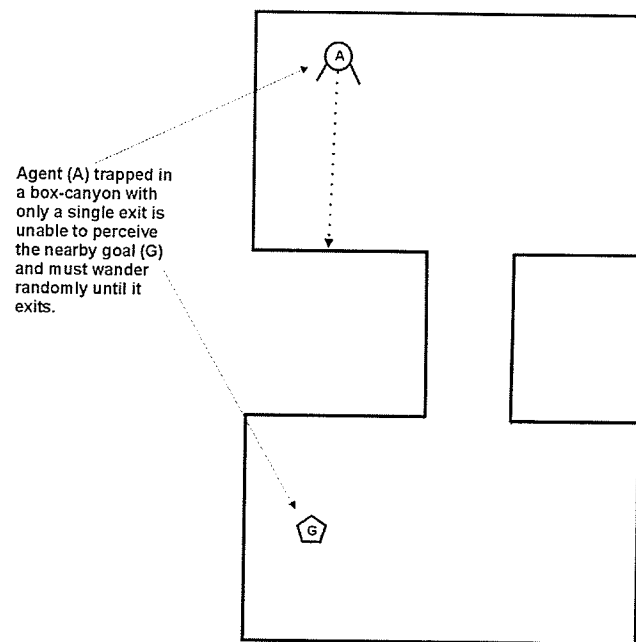
Figure 2.1: Box-canyon situation

see its goal, but must first move *away* from the goal to ultimately reach it [Pirjanian, 1999]. Typically, a certain amount of noise or randomness is added to the agent's chosen vector in order to deal with the problem of local minima [Balch and Arkin, 1995, 1994; Pirjanian, 1999]. Agents can similarly become trapped in local maxima where they lack the stimuli needed to choose a movement vector. For example, box-canyons (see Figure 2.1) are a common local maxima problem, where agents are unable to see a goal of any type [Sgorbissa and Arkin, 2001] and are reduced to wandering randomly before stumbling upon an exit [Balch and Arkin, 1993]. The lack of goal-directed ability makes purely reactive navigation unsystematic, and agents (as individuals and as a team) do not benefit from the knowledge they have gained by exploring an environment (unless they are constructing a detailed world model as

they explore) [Moorman and Ram, 1992]. Thus, if an agent trapped in a box-canyon finds its way out and gets trapped in the same box-canyon again, it must repeat its search for an exit. In addition to local minima and maxima, agents are also subject to cyclic behaviour, where they oscillate between multiple stimuli and never reach their goals (or at best waste time) [Balch and Arkin, 1997; Arkin and Balch, 1997].

Thus, an ideal agent navigation system is one that has the responsiveness of local reactive navigation and the proactivity of path planning methods. One solution to this problem is to allow agents to maintain state to remember experiences and places they have been [Balch and Arkin, 1993]. However, a fundamental design goal of purely reactive agents is that they maintain little or no state information [Brooks, 1986]. Even though behaviour-based agents relax this restriction it is still generally desirable to minimize state information in these agents for reasons of parsimony. Both methods can be used in agents, where local navigation is practiced by the reactive subsystem and path planning methods are employed in a deliberative controller, such as the many hybrid implementations (as discussed in Section 2.1.3) [Arkin and Balch, 1997; Fu et al., 1996]. In systems that employ path planning, reactive navigation allows robots to respond to the unexpected in real-time and handle aspects of the environment that have changed or were not known ahead of time [Balch and Arkin, 1993]. Consequently, improving local navigation is desirable, whether it is used exclusively or as part of the reactive sub-system of a hybrid agent. In either case, local navigation plays a part in an agent's navigation, making it desirable to improve it as much as possible. Moreover, if reactive navigation can be made more effective, it can be employed more widely without resorting to more expensive path planning techniques.

One such effort in improving local navigation is the work of Balch and Arkin [1993], which presents an exploration strategy that breaks the environment into a spatial grid stored in the agent's memory. In this work, each cell in the grid corresponds to a coordinate position in the environment. As the agent explores the environment, a spatial mapper records the number of times a location has been visited. An *avoid-past* motor schema is then used to cause the agent to move in a vector away from visited areas. By avoiding locations that have previously been visited, the agent naturally moves toward areas that have not been explored (or at least not thoroughly). When an agent became trapped in a box-canyon, it would gradually visit locations throughout the room and eventually cause the agent to move away from visited locations toward the exit. As a result, the robots in Balch and Arkin's experiments were found to handle box-canyons better than without the use of a spatial grid. Balch and Arkin's spatial grid approach is, however, intended to improve the explorations of a single agent, not a team of agents.

To improve the explorations of a team of agents, Burgard et al. [2000] implemented a team of robots that constructs a global map of the environment as they explore and assign probabilistic utility values to key target points in the environment. The utility of a target point is a function of the likely degree of unexplored area perceivable by the robot upon reaching that point (with higher degrees of unexplored areas being more valued), and the cost of reaching the point based on the robot's current location. Since unexplored areas have higher utility, the agents' search pattern minimizes overlap and the team is able to explore an area faster than otherwise. However, this technique depends on agents being able to reliably share information and requires extensive

communication. As well, localization is still necessary, and the challenges localization presents remain an issue.

Sgorbissa and Arkin [2001] attempt to improve local navigation for a team of reactive agents operating in an unknown and complex environment. Using line-of-sight communication, agents employ two basic local navigation strategies that share either an agent's state or goals. In the *state-sharing* strategy, robots that are in trouble are attracted by teammates that are not, in order to extract themselves from undesirable situations. Conversely, in the *goal-sharing* strategy, robots communicate their goals with one another and are drawn to teammates that can see or have seen their goal [Sgorbissa and Arkin, 2001]. Consequently, robots are drawn more directly to their goals.

Navigating in or as part of a group introduces another level of difficulty to the navigation task. When operating in a group, agents need to be prepared to adjust their movements to avoid other agents that cross their path. In addition, it is often desirable for a team of agents to move as a unit in formation. If agents are expected or designed to move and navigate in formation, they must *know* their place in the group and actively work to maintain that position. They must be able to avoid obstacles and quickly retake their proper position within the larger group [Balch and Arkin, 1995].

Balch and Arkin [1995, 1998] and Arkin and Balch [1998], for example, embed formation control in a team of Unmanned Ground Vehicles (UGVs). These automated scouts use four main formations (diamond, wedge, line, and column) when moving across terrain. Formation control is embedded in their schema-based reactive sub-

systems by addition of a *maintain-formation* behaviour. The formation behaviour is concerned with keeping the UGV in proper position relative to its teammates according to the currently active formation. The formation behaviour calculates the UGV's desired direction and velocity using a perceptual schema *detect-formation-position* to determine where it should be relative to its teammates. The motor schema maintain-formation then calculates a vector of travel toward this ideal position [Arkin and Balch, 1997].

Similarly, Fredslund and Mataric [2002] present a method that allows agents to travel in formation using only local sensing and limited communication. In this work, agents move in chains ordered according to numeric identifiers (ID's) assigned to each team member. A single agent adopts the role of *conductor* and is responsible for broadcasting formation changes to the team. Agents follow a teammate with an ID higher or lower depending on the formation. In *centered-formations*, the robot with an ID in the middle of the chain becomes the conductor. In *non-centered formations*, the robot with the lowest or highest ID is elected conductor. As the agents travel, they broadcast their identity and position to their teammates. Each agent is responsible for moving as required to keep a teammate with appropriate ID (depending on formation) at a particular distance and angle.

Werger [1999] produces formations among a team of robotic soccer agents by the addition of a *dispersion* behaviour. When a robot senses something near to its left or right side, it moves away from the object (robot or obstacle). This has the effect of causing agents to move in formation, reducing interference between them. Offensively, as agents follow the ball they are kept at a distance from each other by

the repelling effects of the dispersion behaviour. The behavioural tension between the dispersion and ball attraction behaviours was found to cause agents to advance in rough v-formation towards their opponent's goal, leaving them well-positioned to recover the ball if the teammate with ball possession were to lose it. Defensively, agents were found to orient themselves in a semi-circular formation by the combined attraction to the ball and repulsion from each other by the dispersion behaviour [Werger, 1999]. Mataric observed similar emergent global flocking behaviour using only local sensing and the interaction between three basic behaviours (avoidance, aggregation and dispersion) [Mataric, 1997]. Both Werger and Mataric's formations are distinguished from Balch's formations above, in that a specific geometric distance is not specified [Balch and Arkin, 1998].

Whether coordinating exploration strategies or moving in concert, varying degrees of communication are required to allow agents to share their perceptions, announce discoveries (such as the locations of a discovered goal) or in the event that the team becomes separated, to regroup [Arkin and Balch, 1998; Balch and Arkin, 1994; Burgard et al., 2000; Fredslund and Mataric, 2002; Groner and Anderson, 2001; Kube and Bonabeau, 1998; Mataric, 1997]. The formations of Werger [1999] and Mataric [1997] work primarily by observation and innate behavioural interplay, requiring minimal communication. Fredslund and Mataric [2002], by contrast, use communication to broadcast their position and identity to their teammates and to negotiate and announce formation changes [Fredslund and Mataric, 2002; Mataric, 1997].

The next section presents a brief overview of communication, and the strengths, weaknesses and trade-offs involved in employing communication in single and multi-

agent navigation.

2.4   Communication and Navigation

Explicit communication occurs as a result of a direct and purposeful information exchange between two or more individuals. It allows entities to share and negotiate plans, pool sensory information, and in learning situations allows agents to share reinforcement [Stone and Veloso, 1999]. It may also be needed when agents are outside visual range of one another. By communicating, agents can coordinate and collaborate and signal their intentions to avoid performing redundant work.

Explicit communication has been used to aid navigation in many implementations [Sgorbissa and Arkin, 2001; Vaughan et al., 2002, 2000b; Balch and Arkin, 1994]. Tambe et al. [1999] for instance, in their STEAM (a Shell for TEAMwork) system, use direct communication between team members to support plan formation and execution. STEAM uses an explicit teamwork representation based on joint intentions (the group's commitment to group actions) and a shared representation of group plans for a robotic soccer team. In STEAM, agents communicate changes in active team operators, such as when events invalidate an active team operator or enable a previously inactive one. By using explicit communication, this information can be passed to agents that may not be in a position to observe the events causing the operator change.

Most domains have a cost associated with explicit communication. At a minimum, explicit communication requires a common representation and means to communicate [Tambe et al., 1999]. For physical robots, specific communication apparatus is needed

to transmit and receive messages. For either software or hardware agents, software communication primitives are needed to process and interpret communicated messages [Balch and Arkin, 1994]. Even the mere existence of explicit communication (as opposed to the information in it) may endanger agents in some domains by giving away the presence of the agents performing communication [Tambe et al., 1999]. As well, if communicated information is not relevant to the recipient, time and effort processing this information is wasted [Werger, 1999].

Explicit communication may also be prevented by the nature of the environment (e.g. after disasters where infrastructure cannot be depended upon), and may be subject to physical delays (e.g. in distant space scenarios). This, along with the cost of actually generating and interpreting communication, provides ample reason to attempt to minimize explicit communication. Agent parsimony provides another motive: if we can adequately perform a task without explicit communication, it involves less effort and is less costly in terms of agent development [Arkin and Balch, 1998]. The amount of teamwork that can be performed without explicit communication is also interesting from a multi-agent systems standpoint, since it provides insight into the nature of communication and teamwork.

One approach is to make explicit communication more efficient and be more selective in its use. In STEAM, for example, agents consider the cost of communication before making a decision to do so, and avoid communicating information that they believe other team members can already observe on their own. Groner and Anderson [2001], on the other hand, minimize communication via *passive cooperative localization*. This technique allows agents to selectively exchange data on the positions of

navigational landmarks to allow them to re-localize in the face of ongoing uncertainty and sensor error while still performing useful work. In their work, communication is necessary, but is minimized in order to limit the amount of time agents spend communicating, freeing agents to continue with work more directly applicable to the task at hand.

Rather than simply minimizing explicit communication, another possibility is to replace it entirely with implicit communication. Implicit or indirect communication occurs by observation using an agent's pre-existing sensory apparatus (leveraging vision or sonar already needed for detecting the agent's surroundings). Information is passed through modifications to the environment and by observing the actions of others [Werger and Mataric, 1999]. Implicit communication is also referred to as *stigmergy* in the literature, a term used in biology to describe the influence that previous changes to the environment can have on an individual's current actions [Holland and Melhuish, 2000]. Stigmergy is a common control method in lower animals, especially social insects [Perez-Uribe and Hirsbrunner, 2000].

Ants, for example, have evolved to secrete pheromone when transporting food back to their nest. Other ants are attracted by the trail of pheromone that is created and follow it to the newly discovered food source. As more ants detect the pheromone trail and follow it, they lay down additional pheromone, making the trail more attractive to others [Holldobler and Wilson, 1990]. When food runs out, more ants wander away from the food source. With fewer ants on the trail, less pheromone is deposited, attracting fewer and fewer ants. Coordination of a primitive sort is thus achieved not by directly communicating ant to ant, but by modifying the environment in such a

way that others can use this information [Resnick, 1998].

Termites, by contrast, use stigmergy to produce complex structures without explicit coordination of their actions. Instead, worker termites are stimulated to take appropriate action at the proper time by observation of new structural features and the activities of other termites. Coordination and sequencing of the tasks necessary to complete a project is therefore directed by the structure itself, rather than explicit communication between workers [Holland and Melhuish, 2000].

According to Holland and Melhuish [2000], stigmergy is considered to have occurred whenever an earlier change to the environment affects the actions of one or more agents. This can happen by affecting the agent's choice of action, or by changing the location, frequency or strength of the agent's actions [Hammond et al., 1995]. Stigmergy can therefore be considered a form of *stabilization* [Kushmerick, 1994, 1996; Hammond et al., 1995], in that those employing it are modifying the environment to better suit their needs.

Humans stabilize their environments constantly. We structure the world we inhabit in order to allow others to expect regularities in their interactions and travels. In doing so, we often leave very sophisticated messages for one another. When driving down a modern highway, for example, much of its complexity is interpreted for drivers ahead of time and displayed on signs as the path is traversed [Agre, 1988]. People travelling through the woods mark places they have been in order to avoid wandering in circles. In a broader sense, we also stabilize our environments to make things easier on ourselves [Hammond et al., 1995]: we store things in regular places, have clearly marked containers, build paved sidewalks, and engage in a host of other

modifications in order to save ourselves the continual effort of life in less structured surroundings. In the same way, agents can be designed to modify their environment to better suit themselves and adapt to situations that differ significantly from what their designers anticipated. Though this is not stigmergy *per se*, there are intriguing similarities to it.

Findings such those of Balch and Arkin [1994] imply that implicit communication can be used as an effective alternative to the more costly explicit type. In studying the effects of communication, Balch and Arkin [1994] found implicit communication to be a significant factor in the robots' performance of a grazing task. This task involves a robot visiting every location in the environment (analogous to mowing or consuming grass). In this experiment, robots could simply observe the grazing swath left by others and avoid covering the same ground, allowing the team to complete the task faster as a result. They concluded that explicit communication is unnecessary for tasks where comparable implicit communication already exists [Balch and Arkin, 1994]. Consequently, the use of explicit or intentional communication can be avoided when similar information can be transmitted through modifications to the environment itself.

This has powerful implications for the design of multi-agent systems. It hints at the feasibility of creating highly coordinated teams without the use of explicit communication [Kube and Bonabeau, 1998]. By considering the synergies that can be obtained between agents' actions on the world and the world's actions upon agents, agents can be much simpler in design. This assumes, of course, that the method of implicit communication employed does not place undue burden on the agents expected

to recognize and parse the communication [Anderson and Wurr, 2002]. If the communication is too subtle or requires complicated sensory systems, then agent complexity can actually be increased. Ideally, implicit communication will be incidental to task performance rather than a deliberate and separate action. This has the advantage of not requiring any additional time and effort in order to communicate.

The potential benefits of using stigmergy as a communication method have led a number of researchers to explore its applications in single and multi-agent robotic systems. The next section provides an overview and discussion of work in this area, particularly with respect to navigation.

2.5 Stigmergy in Robotic Agents

Stigmergy is attractive to robotic agent designers for several reasons. By storing information in the environment rather than in an internal world model, agent memory requirements can be minimized. Moreover, maintaining consistency between the world and a world model is not required - the world serves as its own best model [Brooks, 1991a]. This makes it highly applicable in dynamic domains where the environment can change quickly, and in the physical world where a robot's perceptions are subject to sensor inaccuracies. In both situations it makes sense not to rely on an internal world model unless absolutely necessary - such a model may be stale or inaccurate due to the cumulative effect of sensory error [Werger, 1999].

In addition to minimizing agent memory requirements, the externalized information can theoretically be stored at the location that it is most useful [Parunak et al., 2001]. Indirect or stigmergic communication has the advantage of being *passive* in

that an agent can naturally integrate it with other sensory information and respond accordingly. In a behaviour-based agent, the presence of a modification to the environment can influence an agent through a behaviour designed to react to its presence appropriately and automatically. For these agents, the relevance of communicated information is determined by an agent's position in the environment, and does not require a long chain of logical reasoning [Parunak et al., 2001]. Communicated information is therefore less likely to interfere with or disrupt the activity of agents to whom the communication is not important. Furthermore, communicated information can be more time-extended, if transcribed into a stable medium, than a simple information transfer between an initiator and one or more recipients.

To date, most work with stigmergy has dealt with its effects on simple homogeneous robots collectively performing foraging or sorting tasks (e.g. [Werger and Mataric, 1999; Holland and Melhuish, 2000; Balch and Arkin, 1994; Parunak et al., 2001]) based on that of the natural creatures (e.g. ants and termites) that inspire the model. Social insects such as ants are homogeneous and redundant, which means that sub-tasks necessary to complete a larger task do not have to be completed by a single agent: the presence of environmental cues alone eventually ensures that a complete sequence of actions is executed, even if the sequence is performed by different individuals [Holland and Melhuish, 2000]. However, the potential benefits of modifying the environment to affect the actions of others can be readily applied to more complex domains and purposes.

Balch and Arkin [1994] have studied the effect of communication in three different multi-robot tasks: foraging for objects (gathering and returning them to a home loca-

tion), consuming objects (finding and consuming in place) and grazing (as described in Section 2.4). Most intriguingly, external communication mechanisms were shown to improve performance of the first two tasks, but not the latter (already discussed in Section 2.4).

Werger and Mataric [1999, 1996] use a form of stigmergy in which robot bodies are substituted for chemical pheromone. In their work a team of robots searches an area without global positioning and only limited sensors by forming robot chains. One end of the chain stays in contact with a home position, while the other end extends outward. Communication is passed down the chain using simple physical taps. Robots are able to briefly leave the chain and search for items before returning to the chain. It is possible to use these taps to re-orient the chain toward newly discovered items of interest. The primary drawback with these robot chains is that close contact between robot bodies limits parallel action. Most of the team members are prevented from actually doing useful work by the necessity that they keep their position in the chain. In addition, the area of exploration is bounded by the maximum length of the robots' bodies laid out in a line.

Balch and Arkin [1993] have explored using limited local spatial memory to deal with the box-canyon problem (this problem was described in Section 2.3) in reactive schema-based robotic agents. Agents remember parts of the environment that they have visited by recording this information internally in a 2-dimensional integer array. Each entry in the array corresponds to a portion of the environment (a cell of arbitrary size). The value stored in each element records the number of times that a given location has been visited in the past. The more often a location has been visited

in the past, the greater its repulsive force on the agents. This can be considered to be "stigmergy in the head", where the robot's movements are recorded in internal memory rather than externally in the environment.

By avoiding previously visited locations, a single robot was found to be better able to deal with box-canyons and was not as dependent on noise (randomness added to its movements) to extract itself from local minima. Though navigation was improved, Balch and Arkin discovered that the robot could become trapped by visited locations and find no direction in which to move. However, they believed that this problem could be solved by introducing a decay mechanism to the internal memory, similar to the natural decay of ant pheromone trails. Finally, since the stigmergy depends on an agent maintaining a rudimentary world model, this model occasionally became inconsistent with the world [Balch and Arkin, 1993] causing a divergence between the agent's perceived and actual location. This underscores the value of storing this information in the environment (as opposed to in memory), as doing so ensures that there can be no discrepancy between the world and the recorded location. At the same time, it does not increase agent computational complexity and memory requirements.

As an alternative to stigmergy, the use of stigmergy-inspired modelling methods in conjunction with external communication has also been explored. Vaughan et al. [2002, 2000b,a], for example, developed a team of physical robots that could locate a supply of resources within an initially unknown complex environment and return it to their home base. The robots are physical entities operating in the real world, but generate and share waypoint coordinates via radio communication, which they maintain as an internal list of *crumbs* and *places* that form trails for the agents to

follow.

In Vaughan's approach, a waypoint (*crumb*) is a data structure that records the *place* (P) the crumb refers to, a position (L) in the agents' shared *localization space* (i.e. a spatial or topological representation of the agent's own position in the environment [Vaughan et al., 2002]) which refers to the coordinates of the goal, an indication of the distance to goal (d) and the time when the crumb was created (t). Robots create trails by generating a virtual *place* record when they experience an event (typically discovery of a goal object), which records the event type and the robot's current estimated location.

Each robot is responsible for broadcasting a temporary trail at a regular interval that contains a listing of all *places* (event locations) in the robot's trail, an indication of the most recent place visited and a single crumb that represents the agent's current location and distance from the most recent place. Robots add to their trails by integrating trails communicated by other robots into their main trail. In this way, robots share and benefit from the explorations of their teammates. The robots utilize the trails so produced by checking the list of crumbs that fall within their sensory radius and moving towards the crumb with the lowest estimated distance to the goal. Using this algorithm, the robot moves progressively closer to the location of the event by following the trail of crumbs with decreasing distance estimates.

Like Balch and Arkin's work, this is simulated stigmergy - all the disadvantages of explicit communication are still in place. A reason for the use of simulated stigmergy here is that marker clutter would be a significant problem, since multiple agents drop crumbs at regular intervals. Another limitation of this method is that it depends on

agents being able to localize themselves within their environment (via odometry or another method). The severity of this drawback is clear in that the method was found to fail after about 28 minutes of operation when the cumulative error between the robots' coordinate records and the real world became so great that their respective trails could not be shared effectively. Another difficulty is that the robots also need to be capable of accurately measuring the distance to the goal. In a complex and unknown environment with many twists and turns this can be very difficult, since the distance is rarely based on simple line of sight. It is quite possible that a goal location might be only a few feet from an agent in a straight line, but the agent might have to travel hundreds of feet to get there (because of walls or other obstacles).

Parunak et al. [2001] and Sauter et al. [2002] employ ant-inspired stigmergy in a software environment that uses synthetic pheromone to coordinate unmanned aircraft that use potential field-based navigation (potential fields were described in Section 2.1.2). The method described by Parunak et al. [2001] and Sauter et al. [2002] uses a distributed network of *place* agents that are used to record and control what they cite as the three key aspects of ant stigmergy: pheromone aggregation, propagation and evaporation. Aggregation refers to the accumulation of agent pheromone to reinforce paths. Propagation refers to the diffusion of pheromone to nearby locations. Lastly, evaporation describes the natural decay process by which old or obsolete pheromone trails are removed from influence naturally.

Each place agent is responsible for a particular region of physical space, divided into hexagonal pieces and connected to others via wireless network connections. Individual mobile agents are represented by a single *walker* agent that spawns *ghost*

agents to traverse the network of place agent nodes in search of targets in the network. Place agents act as physical bodies to allow software ghost agents to pass through allowing travel across a physical environment. When a ghost agent locates a target it heads back over the network of place agents to its associated walker agent leaving deposits of pheromone at each place agent visited. The walker agent for its part follows the pheromone trail integrating its attraction to the trail of place agent nodes into its overall movement vectors until reaching the target. As trails become obsolete due to targets being eliminated or having moved, they dry up as the place nodes gradually decrease their level of attractiveness over time.

This technique allows agents to move through a battle region dynamically using potential field-based navigation. However, it requires very extensive infrastructure to be in place in order to operate. The place agents must first be evenly distributed throughout a predetermined region, be able to communicate reliably with each other via a wireless network, and must also have appropriate sensory apparatus to detect hostile targets that are of interest to the ghost agents that traverse the place network. These are strong conditions to attempt to place on an unknown environment, let alone a battlefield, and consequently the approach seems unsuitable for allowing agents to autonomously navigate such an environment.

Kube and Bonabeau [1998] study ant-inspired coordination in a transportation task using a team of robots with decentralized control. In their work a robot team cooperatively performs a box-pushing task without explicit communication, and using locally sensed information only [Kube and Bonabeau, 1998; Kube and Zhang, 1995]. When robots involved in the task detect a lack of progress (i.e. the box is not moving),

they re-orient their position until box motion is once again observed. Thus, if robots are pushing from opposite ends and hinder one another's efforts, they gradually realign themselves. Directed box-pushing to a position specified by a spotlight is also achieved without explicit communication. This is accomplished by designing robots to push the box only when they are in contact with the box and when the goal spotlight is not detectable. This has the effect of aligning the robots on the side of the box furthest from the goal, causing them to push it in the desired direction. Though this work focuses on stagnation recovery behaviours for a team of reactive robots, it is relevant to this thesis in that the robots sense progress or lack of progress (caused by the incompatible movements of the other agent) in the environment and adapt accordingly through performance of the task itself. This emphasizes the feasibility and validity of using stigmergy to improve agent performance as individuals and a team.

Rumeliotis et al. [2000] present a somewhat different ant-based approach to navigation that uses landmarks rather than pheromone for navigation. Desert ants navigate using visual landmarks and path integration. Path integration is the process of keeping an accurate idea of the direction of one's starting point relative to one's current position by updating this global vector as one travels with angles steered and distances covered [Rumeliotis et al., 2000]. Even for human-beings, path integration in unfamiliar or confusing areas can be quite difficult and easily subject to error.

Consequently, these agents learn and integrate a series of local vectors between a number of landmarks to navigate [Rumeliotis et al., 2000]. This allows them to follow meandering and complex paths to their nest or outward from it by breaking

the journey into discrete vectors between recognized landmarks. Though the agents are not explicitly creating the trail, they are actively interpreting their environment to their advantage. Howard et al. [2002] use stigmergy in a similar manner. In their work, the bodies of a team of mobile robots are used as landmarks to allow robots to localize themselves based on the relative range, bearing and orientation of other agents in a dynamic and/or hostile climate. While this shows the potential for using stigmergy by actively creating landmarks when none exist, the problem with this approach is the use of the robots themselves. A robot serving as a landmark is not likely doing useful work in such an environment.

2.5.1   Stigmergy and Teleautonomy

Stigmergy can also be used as a form of teleautonomy [Anderson and Wurr, 2002], which involves remotely issuing instructions to one or more otherwise autonomous agents to influence their actions and behaviour. In many cases, teleautonomous control occurs via a single operator issuing radio signals while otherwise autonomous agents attempt to integrate fulfillment of these instructions into their ongoing actions [Anderson and Wurr, 2002]. While this is useful, it would be desirable for reasons discussed above to have such instructions provided without explicit communication and to have persistence.

Arkin and Ali [1994] demonstrate a human operator influencing the behaviour of agents either as another schema or by changing the behavioural settings of individual robots remotely. This allowed a human being with a global perspective of the environment to steer robots out of box-canyons and local minima. A weakness

in their approach was that the human operator always influenced the entire group and could not restrict influence to only the individuals of interest. A dilemma with teleautonomous control is that time-extended concentration (as well as the global perspective) may not feasible.

Using stigmergy as a mechanism for issuing teleautonomous instructions is attractive for several reasons. Were a human to control a robot dropping markers, these markers could serve as instructions just as a broadcast network signal could. Moreover, stigmergy is inherently context-specific, influencing only those robots that it is relevant to (i.e. those agents close enough to observe it) [Arkin and Ali, 1994]. A group of individuals can be influenced collectively, without targeting the *entire* group at the time the command is issued. Using stigmergy as a mechanism for teleautonomy can also be used to reduce perceptual load on agents by having a human mark out features of the environment that agents may be ill-equipped to detect. Given that this can be accomplished with only the infrastructure already in place for stigmergic communication, the ability to provide stigmergic teleautonomy is a useful advantage to a stigmergic navigation system.

2.5.2 Summary

Despite its attractiveness, using stigmergy to modify agent behaviour favorably is not without challenges. Unless, the agents are performing stigmergy in an internal simulation (as in Vaughan et al. [2002]) they will require a mechanism to mark their environment directly [Werger and Mataric, 1999]. In the physical world, this might involve dropping items of a certain type (beacons, shapes), which will be in finite sup-

ply and most likely encumbering in large quantity. On the other hand, marking the environment directly using chalk or similar mechanism requires precise motor control and is potentially damaging to the environment. Perception is also an issue. Physical agents need to have the capacity to accurately recognize stigmergic markings. Consequently, the stigmergic communication should be easily recognizable. One possibility is to use a trail of pheromone like ants by integrating biological sensors capable of sensing chemical trails into mechanical robots [Kuwana et al., 1999].

Even so, stigmergic communication can begin to clutter the environment if not pruned regularly or made indiscriminately. Consequently, it is imperative that alterations to the environment be made judiciously. If the information being imparted is time sensitive, the indirect communication will eventually be obsolete. In these situations, a mechanism may be required to automatically clean up out-dated or obsolete communications (as with ant pheromone trails), lest it negatively affect agent performance. Furthermore, agents need to be able to decide between various attractors reliably, or they might oscillate between the multiple stimuli. If so, they may need to be able to differentiate markers to allow them to follow a gradient towards a goal or at least recognize markings that have already been visited.

2.6 Using Games for AI Research

As mentioned in Chapter 1, the test-bed for the experiments conducted in this thesis is a simulation-based computer game. While the nature of the specific environment will be discussed in Section 4.2, it is worthwhile mentioning the contributions such environments have made to AI and why such environments are employed by AI

researchers.

First, there are a number of reasons that researchers opt to use simulation in general as opposed to physical experimentation. Constructing physical robots and experimenting in the real world is expensive and presents challenges unrelated to many core AI issues that are of interest [Etzioni, 1993]. This includes, for example, sensory issues and the mechanical construction and maintenance of robots. Using computer simulations allows tangential issues of vision processing and mechanical complications to be avoided, allowing more time to be spent focusing on the issues of interest. Furthermore, physical agents are typically expensive and not readily expendable.

By contrast, although computer simulated environments can also require extensive investment of time and effort to construct [Lewis and Jacobson, 2002], they can be used by countless researchers and be reconfigured as necessary to explore a myriad of issues once completed [Etzioni, 1993]. Moreover, software simulation affords a higher degree of experimental control. Experiments can be run repeatedly with identical settings, under more stringent conditions, or in domains that would be dangerous to equipment. Of course, simulated research does not eliminate the necessity for experimentation in the real world [Brooks, 1991a]. The desirability of both is shown in the RoboCup robotic soccer challenges, which occur in both simulated and real world environments [Tambe et al., 1999; Kitano, 2000].

To avoid the time and expense of developing computer simulators, a growing number of researchers are now leveraging computer games to study AI issues and as an alternative to the more expensive computer generated forces simulators [Lewis and

Jacobson, 2002; Laird, 2000, 2001; Laird and Lent, 2001; Laird, 2000]. Computer game engines are attractive in that they do not require expensive hardware and specialized graphics software, yet handle I/O, 2D/3D rendering, network connectivity, sound and reasonably approximate real world physics. This makes good simulation environments accessible to a wider number of researchers that are unable to afford the expensive hardware and software of a high-end simulation environment. This is especially true in recent years, as games have become more elaborate and better approximate the real world. The practicality of using computer games has increased as many game makers have begun providing better software interfaces to their products [Lewis and Jacobson, 2002]. The efficacy and legitimacy of using computer games for AI research is evident by the increasing number of researchers using games such as *Unreal Tournament*[1], *Quake*[2] and *Half-Life*[3] as simulators [Lewis and Jacobson, 2002; Laird, 2001; Bylund and Espinoza, 2002; Jacobson and Hwang, 2002; Kaminka et al., 2002; Piekarski and Thomas, 2002].

Laird [2001, 2000]; Laird and Lent [2001] cite several additional advantages to using computer games as an AI test-bed.

1. Computer game manufacturers and game players are placing increased emphasis on more human-like behaviour and intelligence from computer-controlled players.

2. Advanced AI is generally considered to become the new distinguishing technology of computer games.

---

[1]Unreal Tournament is a Trademark of Epic Games Inc.
[2]QUAKE is a registered trademark of Id Software, Inc.
[3]Half-Life is a registered trademark of Sierra Entertainment, Inc.

3. Continual advancement in available game hardware is likely to make it feasible for more processing power to be devoted to AI processing in game play.

A simultaneous advantage to game developers and players is that research advances contribute to making computer games better as well. In many games today, computer-controlled players (or *bots*) provide ample challenge due to their accuracy and reaction time as opposed to complex tactics or strategy. For example, agents might be allowed to see through walls or be given a complete map of the environment *a priori*. While game players tolerate this, anecdotal evidence suggests that many would prefer to play against opponents that are subject to the same limitations as the player [Laird, 2001], as evidenced by the enormous popularity of online game play.

The quality of the AI used to control the characters that populate these worlds is an important aspect of the enjoyment the user derives from the experience. As game development companies strive to make their games standout, the amount of research funding available for computer game AI will undoubtedly grow [Laird, 2002].

By using computer games as a research test-bed, researchers are freed to focus on AI issues rather than expending effort creating these simulated environments [Laird, 2002]. At the same time, computer games can benefit from advances in AI research applied directly in a computer game environment, allowing the entities that populate them to operate more effectively, without "cheating".

2.7   Summary

This chapter has provided the reader with an overview of the issues and challenges of embodied multi-agent navigation, and the need for communication in this

task. Further, it has demonstrated the previous use of stigmergy as a useful form of communication. The next chapter describes an approach to navigation that uses stigmergy to solve some of these issues.

Chapter 3

Stigmergic Navigation

> The reasonable man adapts himself to the world: the unreasonable one
> persists in trying to adapt the world to himself. Therefore all progress
> depends on the unreasonable man.
>
> – George Bernard Shaw - 1856-1950

Chapter 2 has outlined the advantages (and drawbacks) of local navigation and

has described the promise shown by stigmergic techniques in assisting agents in their

navigational tasks. Accordingly, this chapter outlines methods to improve local nav-

igation performance in a complex environment using simple *stigmergic markers* that

agents deploy as they travel. Stigmergic markers allow agents to encode information

gained by their explorations into the environment and share that information with

their teammates (and even themselves). The approaches described herein are ap-

plied and tested in an environment that is more complex than many used in previous

work (e.g. [Balch and Arkin, 1994, 1993; Werger and Mataric, 1999, 1996]). More-

over, my techniques implement stigmergy directly, rather than simulating stigmergy

in the internal representations of agents as has been done in previous approaches

(e.g. [Vaughan et al., 2002, 2000b,a; Parunak et al., 2001; Sauter et al., 2002]). I also present an approach to stigmergic trail-making that minimizes clutter.

3.1 Overview of Approach

My methodology for achieving the research goals involves designing agents to identify locations deemed heuristically useful, marking these locations using stigmergic markers, and exploiting that information appropriately. Hereafter, I refer to the process of using these markers to make more purposeful reactive navigation decisions in an unknown environment as *stigmergic navigation.*

As part of this work, I have employed stigmergic navigation techniques to solve a number of the problems faced by agents using local navigation including:

1. allowing agents to handle local maxima situations or box-canyons (described in Section 2.3) more quickly.

2. allowing agents to avoid and escape local minima.

3. after a goal is first discovered, allowing agents to share knowledge about the goal's location, so that they and their teammates locate it more quickly, reliably and with greater frequency without constructing or maintaining a world map.

The experiments illustrating these situations appear in Chapter 5. The remainder of this chapter describes the methodology employed and agents used to solve these problems.

3.2 Stigmergic Markers

For stigmergy to be possible, agents need to be capable of marking their environment in a manner recognizable to others. A convenient mechanism for accomplishing this (and the one used here) is to simply place recognizable markers (i.e. physical objects) on the ground at particular locations. Dropping a marker on the ground is relatively fast and simple, and does not permanently damage or alter the environment. Such a simple marking is also more readily recognized and parsed than a more complex one. Although there is no limit to the volume of information that can be shared via stigmergy, it is generally better if these objects are uncomplicated, small (physically), inexpensive and expendable. Consequently, the methods described herein strive to keep markers as rudimentary as possible.

There are two categories of stigmergic markers used in my thesis research. The first of these are *homogeneous* markers - very basic markers that can serve either as a source of attraction or repulsion to any agents that regard them. In this form, the presence and location of a marker alone is significant. These markers can be employed to identify areas of the environment that agents should avoid, such as local maxima. They can also be used to signify actions that an agent should take at particular locations, such as leaping over a local minima. Alternatively, they can be used to draw agents towards areas that are more beneficial to the agents' explorations. While they have several applications, the problem with markers this basic is that agents have no ready mechanism for mediating between markers of this type when more than one is visible, other than by choosing the closest. This can potentially cause agents to oscillate between attractive and repellent markers and hinder their performance as a

result, in much the same way multiple goals would.

To supplement these simple markers, I introduce a second type - *heterogeneous* markers. Heterogeneous markers, in addition to imparting information by their location, also encode an attractive *value* that is perceptible by an observing agent. These markers can then be dropped in value order to encode a sequence of markers, for example. The use of multiple encoding values allows a perceiving agent to decide between markers to follow when it encounters more than one. Even if there is no easy method of deciding upon the relative importance of markers, heterogeneity can be used to allow agents to record markers they have visited, since agents can identify them uniquely. This allows agents to disregard visited markers in the future to avoid cyclic behaviour. Using heterogeneous markers allows agents to make more purposeful navigation decisions when caught in an area of the environment where no other stimulus is present. Indeed, Parunak and Brueckner [2000] point out that when mechanics of the environment are in place to take advantage of the increased vocabulary available with multiple marker types, additional marker types can increase performance in coordinating agents.

3.3   Agent and Environment Structure

Behaviour-based agents using the schema-based approach to agent control were chosen as the most appropriate for exploring the effect of stigmergy on local navigation performance because the intelligence of behaviour-based agents emerges from the complex interactions between their behaviours and world [Arkin and Balch, 1997]. A hybrid agent design was not chosen in part to illustrate what was possible with this

much more parsimonious agent design.

In addition, a complex environment was required to reproduce the local navigation problems that stigmergic navigation is intended to surmount. In this regard, a 3D indoor environment was chosen as most building interiors contain numerous open areas (e.g. rooms), doorways, hallways, blind alleys and obstacles.

Stigmergic navigation involves selecting appropriate marker types, conditions for dropping markers, and conditions for following markers. The following sections deal with describing the combinations of these elements used to solve the particular navigation problems outlined in Section 3.1. Each section begins with a brief problem summary before outlining a proposed solution.

3.4   Box-canyons

A complex environment such as an office building or similar structure consists of a number of open areas connected by relatively narrow doorways or corridors that form an interconnected network of box-canyons. As described in Section 2.3, box-canyons can easily confound local navigation methods, since they limit an agent's perceptual range and provide the agent no clues about what direction to take to best achieve its goals. In these situations, the faster the agent moves on to another area to explore, the better.

Two types of markers can be used to allow agents to extract themselves more readily from these perceptually uninteresting areas. The first type are heterogeneous markers that agents are periodically attracted to and that agents can use to mark out narrow regions of the environment or *bottlenecks*. The second type are homogeneous

markers that repel agents that observe them and that agents can use to identify and mark out and avoid local maxima. The use of each of these is described in the following subsections.

3.4.1 Marking Bottlenecks

By placing heterogeneous markers in constricted areas of their environment, agents can supply each other with an attractive stimulus in the absence of pre-existing "naturally-occurring" stimuli. Accordingly, I employ the heuristic of placing attractive markers at narrow (or enclosed) positions in the environment such as doorways and within hallways, since they represent bottlenecks. Within the open area of an empty room, any one coordinate position in the enclosure is no more critical than another in terms of choosing a path from the agent's current location to a location in another room. While locations lying on a straight-line path in the direction of the doorway from the current room to the target room can be considered "better", any other path through any other series of points in the room is still adequate to ultimately get the agent to the goal. The exception to this is the position corresponding to the doorway itself, which must ultimately be part of the path. Any path from a given room to another must pass through some number of doorways and/or hallways.

This technique first depends on the use of heterogeneous markers with unique IDs, and designing agents to be able to perceive these IDs so that they can record markers that they have visited in a short-term memory. As will be seen, once markers are uniquely distinguishable and agents thereby able to recognize and remember markers they have visited recently, improved exploration becomes possible. Precisely how the

marker IDs are signalled is not important – in the real world, the markers might be transponders that emit a unique numeric ID, or might be numbers written on the ground in chalk. Perceiving the latter would be a vision-processing challenge outside the scope of this thesis.

To achieve a balance between exploring an area and moving to a new area, agents must be compelled to follow markers only periodically, rather than continuously. In my approach, an agent is only attracted to markers when it has not followed a marker trail for a given period of time. In this state, the agent is considered to have *marker-affinity* and will move toward any marker that it sees. As soon as it is successful in reaching a marker, it enters into a *marker-following* state. In this state the agent continues to be attracted to markers that it has not visited earlier (as long as it detects it is in a doorway or hallway), but is not attracted to markers it has already visited. It is a given that the first marker it reaches will be in a doorway or hallway, as markers are only dropped in these locations. If the agent has entered a hallway rather than doorway, it will be most likely faced with a trail of markers leading down the corridor.

By recording and ignoring markers that have been visited while in marker following mode, the agent is drawn along by the trail of unvisited markers until it enters another open area. As soon as the agent detects that it is no longer in a narrow space (doorway or hallway) it transitions to a *marker-neutral* state during which it is no longer attracted to markers. Each computational cycle, any markers that have not been visited within a certain period of time are purged from the agent's list of visited markers. This is necessary to allow the agent to potentially traverse a previously

travelled marker trail at some future time. This purging also serves to keep the agent's memory requirements very minimal. The marker-neutral state persists for a predetermined period of time, before the agent once again becomes marker-affinitive.

In essence, these bottleneck markers act to draw agents through tight spaces that are difficult for agents using local navigation to find and move through. If the room has only one entrance/exit, this allows the agent to escape it more quickly and continue its explorations via the trail it followed to get in or another trail laid by other agents. Conversely, if the room has several unexplored portals leading to new regions of the map, the agent has the opportunity to discover these alternate exits during the period in which it is not attracted to markers, causing it to mark them for other agents as well as itself.

In this case, I am essentially providing agents with the features that distinguish hallways and doorways, and similar knowledge could be provided for any other type of bottleneck. A useful future extension, however, would be to mark such structures by looking at the information gain that would be provided by a marker in a given area. This is similar to the work of Burgard et al. [2000] described in Section 2.3, in which agents move to locations from which the most unexplored area can be viewed.

3.4.2 Marking Local Maxima

In addition to using markers to draw agents through constricted spaces in the environment, agents can be assisted in identifying and avoiding local maxima by placing attractive/repellent homogeneous markers at these locations as a warning to others. By marking local maxima with these markers, agents will be pushed through

uninteresting areas and on their way, rather than remaining in a local maxima. Unlike paths to exits and down hallways, local maxima are not as stable in the long term: the locations of goals can change, making local maxima out of other areas in the environment that were previously benign. So a type of homogeneous markers known as *local maxima* markers is defined, with the additional property of disappearing after a period of time, to allow for these local maxima areas to be re-opened to exploration.

In order to minimize clutter and maximize their effectiveness, agents mark out a local maxima under fairly constrained circumstances. For a local maxima marker to be dropped, the following must hold:

1. no goal can be visible from the agent's current position;

2. no local maxima markers can be visible from the agent's current position within a certain range;

3. the agent must be sufficiently distant from walls and other obstructions in all directions (i.e. in an open area);

The first constraint captures the essence of what these local maxima markers are intended to signify. The second constraint is necessary to minimize unnecessary clutter and likewise allow the agent sufficient freedom of movement. The third constraint is intended to ensure local maxima markers are placed in locations where they will be most visible. It is also intended to force agents to stay near the edges of areas that are local maxima (when they are being repelled by these markers). By keeping agents near the edges of a local maximum, they are more likely to happen upon its exits. When they are first attracted to these markers, this also has the advantage of

drawing them into the open, away from obstacles, and where they are able to see a large portion of their environment.

This method of marking can be likened to the use of spatial grid maps by Balch and Arkin [1994]. Unlike their method, however, the agents in this thesis are storing information about visited locations in the environment, rather than in an internal data structure. In addition, this information is less fine-grained, making it less verbose, and can be generated and used by multiple agents simultaneously.

3.5   Stigmergic trail-making

While easing the passage of agents through narrow areas and local maxima is helpful to speed up navigation, neither of these assist directly in finding a goal. In order to assist others in findings a discovered goal, agents must be given the ability to mark and interpret a purposeful trail. Stigmergic trail-making is the process of constructing such a trail via a chain of markers. These marker trails are *cooperatively* constructed by any number of agents over time as they explore. This allows the agents to repeatedly locate a discovered goal without internal world modelling, internal maps, or internal path planning.

The process of stigmergic trail-making is best illustrated by way of an example. Consider the situation in Figure 3.1. A primitive stigmergic trail is created when an agent perceives a goal and drops a marker on the ground at its current location (Figure 3.1a). The marker dropped by the agent is assigned a perceptible numeric value that represents its position in the trail (in this case, 1). The dropped marker identifies a vantage point from which the goal should be visible to an agent standing

at the marker's location. By itself, a trail this rudimentary is of limited utility: to make the trail more sophisticated, it is necessary to extend the trail. As Figure 3.1 illustrates, this occurs when another agent (or even the same agent at a later point in time) sees an end-point of the trail and drops a second marker (see Figure 3.1b). The second marker dropped is assigned a value one higher than the marker that the agent has observed (in this case, 2). As the process repeats, the trail gradually lengthens (see Figure 3.1c, d).

To minimize clutter, agents drop markers only under fairly constrained conditions. When the agent perceives the goal (or a marker already on the trail), it checks for the presence of any stigmergic markers already serving as attractors to this target. If no markers are visible, the agent drops a marker as described above. Note that this does not preclude the construction of multiple trails during this process, only that when such trails are made they will not be in obvious sight of the one another. This is illustrated in Figure 3.1e, where an agent sees the end of a trail, by perceiving a 3 marker but no 4, and so extends the trail in another direction. Note that under conditions such as this, extending the trail in different directions is very helpful - agents coming from different directions or from different sides of obstacles can find a useful path to a goal.

In general, an agent only extends the trail when it perceives what appears to be its end. If the agent perceives other markers from its location, it only drops a marker if the goal or a higher valued marker is not also visible. If such a marker is visible, the agent knows it is not at the end of a trial and does not drop a marker. In keeping with attempting to limit marker clutter, an agent does not extend a trail when it perceives

A) Agent sees its goal but no markers that identify it and drops a level-1 marker at its current position

B) Agent sees a level-1 marker and drops a level-2 marker at its current position.

C) Agent sees the level-2 marker and drops a level-3 marker at its current position.

D) Agent sees a level-3 marker and drops a level-4 marker at its current position.

E) Agent sees a level-3 marker and cannot see the level-4 marker and so drops a level-4 marker at its current position.

F) Agent sees 3 markers from its current position, and moves toward the lowest numbered marker without dropping any markers.

Figure 3.1: Stigmergic trail making

that it is not at the trail's end. Such a situation is depicted in Figure 3.1f: the agent perceives marker 3 and two marker 4's; it knows that 3 is not the end of the trail (because it perceives a 4 marker), and that its vantage point is not far enough away to warrant dropping another marker (since existing markers already supply pertinent navigation information from the agent's current location, or the 4 marker would not be present).

The conditions above allow an agent to simply follow a trail without dropping new markers when there is enough navigation information available to do so. Thus, a branching situation such as that shown in Figure 3.1e will only occur when an agent's perceptions indicate it is at the end of the trail when it in fact is not. In such a setting, there is not enough navigation information to perceive otherwise given the agent's current location, and so branching the trail at that point is a logical action, even though the agent knows nothing about the other branch.

Agents utilize the trail to locate the goal at the other end by following the trail of markers, always moving toward the lowest numbered marker visible. Since the markers are dropped based on visibility, the agents are able to consistently follow one marker to the next without interruption to locate the goal. This also helps minimize the length of a path followed by an agent - if an agent perceives a 3 and a 4 marker, for example, the agent will not bother moving to the 4, since the 3 is already closer to the goal.

The two important features to emphasize in this approach are a) that this trail-building occurs collaboratively; and b) that it occurs while agents follow existing trails and otherwise navigate through the environment. In the case of several goals

in reasonably close proximity, the trails may connect. This may cause an agent to perceive more than one lowest valued marker: in this case an agent will be attracted to the marker in closest proximity at any time, so may start following toward one goal and end up at another. Either way, however, the agent reaches a goal, which is the point of the process.

3.5.1 Marking Local Minima

Another application of stigmergic markers is to allow agents to deal with local minima, such as low lying obstacles that can prevent an agent from moving in a straight line to a visible attractor. In order to deal with this particular situation, I introduce another homogeneous marker type - *local minima* markers. Rather than attracting or repelling agents, these markers are unique in that they prescribe an action to undertake in response to a specific situation at a particular place in the environment.

In the experiments described in Chapter 5, these markers compel agents to jump when sufficiently close to them. In so doing, an agent moving toward an attractor is able to leap over a low lying obstacle and continue on its way, rather than being stuck at the low-lying barrier. In order for this prescribed action to be a useful one, agents only drop these local minima markers when they perceive an attractor and a low-lying barrier in their path to it and when they are sufficiently close to the barrier itself. If the barrier is too high to be jumped over, the agent does not drop a marker of this type. Using markers in this manner can allow agents to handle certain types of local minima more readily, rather than relying strictly on noise added to their movements

(as discussed in Section 2.3).

As with local maxima markers, these markers are designed to disappear after a certain interval. This was deemed necessary since the location of local minima can change.

These markers are somewhat more specialized than the other markers types presented in this chapter. This is a natural consequence of their encoding a particular action to take (since it is relevant to only a fairly constrained situation). Though not investigated here, it is also possible make these markers repellent instead, and place them at locations identified by other agents as local minima. However, as will be discussed in Section 5.3.5, jumping is a somewhat more profitable option in the experiments executed in this thesis.

3.6 Stigmergic Navigation

Section 3.4 and 3.5 above each describe two methodologies for dropping and responding to markers. In each section, the two methodologies are complementary halves of stigmergic navigation. Bottleneck and local maxima markers serve to promote exploration and allow agents to more quickly locate an unknown goal somewhere in the agents' environment. Stigmergic trail and local minima markers on the other hand, facilitate subsequent trips to a goal once it has been discovered. A number of experiments, detailed in Chapter 5 were executed to measure the efficacy of the various marker types and marker dropping and following rules discussed in this section, together and in combination. This includes the combined application of the four basic marker types and uses reviewed here.

3.7 Comparison with other work

Before describing implementation details, it is worthwhile to compare the approach used here to others in the literature. The work of Vaughan et al. [2002, 2000b] with Localization-Space Trails (LOST) described in Section 2.5 bears some similarity to the approach outlined in this thesis. Like the approach described here, their work was shown to allow robots using odometry-based localization to reliably move between a starting position and goal location. Their method was observed to converge to the best route discovered by the team and handle the failure of individual robots (a common benefit of multi-agent systems and stigmergy). As with the routes found using the method described in Section 3.5, the routes determined by LOST were also not necessarily optimal [Vaughan et al., 2002].

Unlike the methods described in this chapter, rather than marking the environment directly via physical markings or modifications external to the agent, their work only simulates stigmergy. Agents maintain crumb trails in individual memory and communicate these explicitly by describing them in terms of common reference features. Simulating stigmergy simplifies marking the environment and cleaning up obsolete markers. This means that agents do not need to carry a supply of objects to mark the environment or employ another means of laying a physical crumb trail. Similarly, removing outdated or unhelpful markers is simply a matter of deleting them from the appropriate memory location. This allows them to clean up old crumbs at a physical distance from the location it references.

Though simulating stigmergy sidesteps the difficulties associated with marking the environment, it sacrifices some of the advantages as well. Most significantly, the

method requires explicit communication to work. Agents therefore require communication hardware and a reliable network to share crumb trails. The method is therefore vulnerable to interruptions in communication (in addition to cumulative odometry errors), such as distance or interposing barriers or hostile signal jamming (in some potential domains). This makes the approach unsuitable for many domains in which stigmergy may be attractive, but where direct communication might be intermittent or blocked. Furthermore, the crumb trails are maintained in internal memory, increasing agent memory requirements. Finally, the processing demands of this technique are increased as well. Agents must refer to crumbs via reference features and construct messages in this format.

In my approach, by contrast, agents mark the environment directly, communicate with one another indirectly and are influenced or react to markers as they observe them. By storing information externally and close to the locations where they are pertinent, the information's relevance is improved. This is analogous to the concepts of temporal and spatial locality of reference related to high level programming language compiler optimization. Temporal locality of reference refers to the idea that if data is referenced or used at a certain point in a program's execution, there is a high probability that it will be used again in the near future. Spatial locality, recognizes that if data is used at some point in a program, there is a high likelihood that data stored physically nearby will also be needed or used by the executing program. Recognizing this, program caching can be improved (via an increased number of cache hits) by caching memory with these two ideas in mind. Similarly, information about an agent's world is most usefully stored physically close to the locations in which it is

likely to be relevant. The result is that the agents do not require explicit communication hardware, nor the more extensive processing necessary to maintain and process internal memory structures. Though the experiments in this thesis are done in a software environment, the methods described are equally applicable to real world domain (difficulties related to perceptual and environmental modification notwithstanding).

Though the methods of Vaughan et al. allow robotic agents to more quickly and reliably navigate between start and goal locations than robots that do not use it, it does nothing to reduce the time it takes for the goal to be initially discovered. In contrast, my method to promote exploration, described in Section 3.4.1, seeks to allow the team to discover an unknown goal for the first time faster than otherwise. This is especially advantageous in applications where quickly locating a particular unknown goal is critical (such as rescue scenarios).

The work of Sauter et al. [2002] described in Section 2.5 also uses a simulated form of stigmergy to aid agents in locating and navigating to targets on a battle field. As with LOST, their technique is dependent on a wireless network and requires advance preparation of the environment to function. It is another example of simulated stigmergy, where environmental changes are stored in nodes in the network, rather than by physically marking the environment. Their technique essentially allows agents to perceive mobile targets at a distance through the wireless connections that are distributed throughout the environment allowing them to perceive more of the environment than lies in their immediate perceptions. This is similar to the marker trails as outlined here, which allow agents to detect a goal at a distance by indirection via the markers.

Beyond the infrastructure necessary for communication, the work of Sauter et al. [2002] also requires special place agents that must be evenly distributed throughout the environment ahead of any navigating agents. This additional factor makes this approach unworkable for most real world environments, as the problem of even distribution is not much less significant than that of navigation itself.

The simulated stigmergy employed in Vaughan et al. [2002], Parunak et al. [2001] and Sauter et al. [2002] have an advantage in that they can more easily control the evaporation and natural clean up of pheromone trails that occur with real world ant trails. Although outside the scope of this thesis, it is worth noting that others are working on physical stigmergic mechanisms that allow for physical fading [Kuwana et al., 1999].

The next chapter (Chapter 4) describes and outlines the design and implementation of both the agents and environment that were used to study the efficacy of the methods of stigmergic navigation covered in this chapter.

Chapter 4

Implementation

This chapter describes the methods I have chosen to implement the strategies described in Chapter 3, including agent structure, environment structure, perception, action selection, action execution and stigmergic marking mechanisms.

4.1   Agent Structure

Agents in this thesis were implemented using a schema-based control approach described previously in Section 2.1.2. This approach combines motor schemas with associated gain values into groups that are tailored to deal with a variety of different agent states. The appropriate collection of motor schemas is activated as the current situation demands in order to allow the agent to respond suitably.

Perceptual schemas are the mechanism that agents use to sense their internal state and external environment in a particular context. For example, the *percept-marker* perceptual schema (as described in appendix A) is responsible for using an agent's

sensory apparatus to detect when a marker is visible and provide interested motor schemas with information about the marker, such as its location.

Motor schemas are responsible for manipulating the agent's effectors to accomplish certain well-defined tasks, such as dropping a marker, moving toward a visible attractor, and similar tasks. For motor schemas responsible for moving the agent toward a particular location, this involves re-orienting the agent so that it faces the motor schema's target and then moving the agent toward it. In my agent implementation, this consists of outputting a vector that indicates the desired direction that the motor schema would like the agent to move along with a desired velocity. The simulation environment will handle dealing with the physical changes in the environment that this requests.

As several motor schemas can be active at any given moment, the vectors that are output by each active motor schema are combined into a single merged vector and velocity that the agent follows. The influence of each motor schema on the final movement vector and velocity depends on its gain value within the currently active assemblage.

As mentioned in Section 2.1.2, motor schemas are grouped into collections known as *assemblages* that are tailored to handle the most important aspects of an agent's current situation. For example, a *drop-marker* assemblage could consist of motor schemas most relevant to placing a marker of some type on the ground. Motor schemas that are inappropriate in a particular state are either not present or have a lower gain value reflecting their reduced importance.

Assemblages also have associated perceptual schemas that trigger their activa-

tion or deactivation. For example, the drop-marker assemblage mentioned above is activated when the *percept-drop-marker* perceptual schema returns a value of *true*. Conversely, the drop-marker assemblage is deactivated when the percept-drop-marker perceptual schema returns a value of *false*. Each agent also has a default assemblage that is invoked when no other assemblage is active, based on the current world state.

Appendix A provides a listing and descriptions of the assemblages and motor and perceptual schemas used in various experimental scenarios.

4.2   Half-Life Environment

The simulated world of the computer game Half-Life was chosen as the test-bed for this research. Half-Life is a 3-dimensional first-person computer game with rich graphics and a game engine that approximates the physics of the real world sufficiently well. Half-Life games are played out on a multitude of maps all with different layouts of rooms, hallways and goals items, making it possible to experiment with a number of different map configurations easily.

The size of the world defined by a Half-Life map is 8191 units in height, width, and depth. Space in this environment is measured in terms of an abstract *unit* that is 1/8191th of the length of a dimension. There is thus no direct relationship to any measurement in the real world. However, an agent in Half-Life is intended to be roughly human-sized, and is defined by a bounding rectangle 72 units high, 32 units wide, and 32 units deep. Common-sense approximation to the real-world equivalent of a unit would thus be about an inch, assuming a human is six feet tall. In referring to the environment, however, I will be using the same unit term used in Half-Life in

order to be consistent with other research using this domain. A particular location in the environment is described by a real-numbered vector (x,y,z).

The makers of the game have previously released source code and instructions for compiling a module that can be linked into the game to create modifications to the original game. This software development kit made it possible to design and implement schema-based agents to participate in the experiments as described in Chapter 5. An added benefit of using this software environment was that it was possible for a human player to enter the game alongside the agents and interact directly with them as another game player. This was useful in observing agent actions and performance and in experimenting with teleautonomous control using stigmergy.

4.3 Agent Implementation

Agents were implemented using Visual C++ 6.0, Service Pack 4, using version 2.2 of the Half-Life standard software development kit (SDK) provided by Sierra. Half-Life development/modification requires a copy of the original game to provide core game functionality. Half-Life runs on Windows 9x, NT, 2000, XP and requires a Pentium 133 or better with 24MB of RAM. The SDK consists of a collection of C++ source code files that can be modified or extended as desired to create new versions of the game in a compiled DLL that is linked in as desired. This SDK is intended to allow programmers to modify the game and create entirely new scenarios. It is possible to develop entirely different games or slight modifications while making use of the same underlying game engine. A number of agents, colloquially known as *bots*, of varying levels of sophistication have been developed to allow human opponents to

play against automated competitors.

In order to implement the agents used in this thesis, I designed and implemented an agent architectural framework in C++ using an object-oriented design to match that of the game engine itself. The resulting framework allows for dynamic reconfiguration of agent assemblages, schemas and associated gain values. The supporting class framework allows for new motor and perceptual schemas to be developed quickly and easily and leveraging existing code as much as possible. The currently implemented motor and perceptual schemas are able to instantiate and make use of each other to build up more complex behaviours. As such, this will serve as a future test-bed for working with behaviour-based agents in complex domains.

Each agent is comprised of a class that represents the agent's internal state and a controller class that initializes its assemblages and executes them. Motor and perceptual schema base classes capture and abstract functionality common to all motor and perceptual schemas. Specialized classes are instantiated for motor or perceptual schemas of a particular type and encapsulate the processing associated with their particular areas. Schema and assemblage classes are generated by factory classes that control their lifetime and clean up memory when they are no longer required. These factory classes ensure that only one instance of a perceptual schema is instantiated and used by any number of motor schemas for a particular agent.

4.4   Action Selection Process

Agents are physically stored as a file containing a list of assemblages, each of which has an associated file of motor schemas and activation or deactivation conditions. In

Half-Life, an entity's reactions to its current situation is carried out via a game engine method called (somewhat grandly) its *think* function. The Half-Life game engine calls this method for each entity (computer controlled players, devices, etc.) many times per second, though the precise count can vary depending on system load. Each update to the state of the environment is referred to as a *frame*, during which the game engine performs entity maintenance prior to rendering the next graphical representation of the environment. This includes calling each agent's think method, where logic is executed that determines the entity's reactions to its current situation. Each agent's *think* method is invoked once during each frame update.

To integrate agent control into the Half-Life environment, it was necessary to create a C++ class that inherits the *think* method from the appropriate base class and then insert the appropriate agent control logic. It is during execution of this method that the agent senses its environment, determines its internal state and decides on and/or updates its actions accordingly. The agent control logic is run server-side in either dedicated server mode, where the game engine runs without the overhead of graphics, or application mode where a human player can host and also join the game itself. In this way, each agent can perceive and act in turn based on what it detects about itself and its immediate surroundings.

Each time frame, the agent's *think* function is activated and each behaviour regards the environment from its own unique perspective. For example, the *avoid-static-obstacle* motor schema (through usage of the appropriate perceptual schema) views any stationary object (such as a wall) in the agent's path as an obstacle and reacts accordingly. Similarly, the *move-to-marker-bottleneck* behaviour is concerned

only with moving to bottleneck markers. Focusing only on particular aspects of the agent's environment makes behaviours more modular, and therefore simpler to implement and manage.

In my implementation, agents first perceive the world through activation and deactivation perceptual schemas associated with the currently active assemblage to determine if the assemblage is still appropriate. If it is not, the assemblage is deactivated and an assemblage appropriate to the agent's current situation is activated in its place from the collection of assemblages. If conditions do not activate a suitable assemblage, a default assemblage takes over instead.

Once an appropriate assemblage has been activated, a think method is called for each motor schema that makes up the assemblage in turn. These motor schemas themselves query appropriate perceptual schemas in deciding what action or movement vector the agent should take. Once each motor schema's think method has been called, their recommended vectors are combined into a final movement vector and velocity. The influence of each schema varies based on the gain value associated with the motor schema within the active assemblage.

For purposes of efficiency, only one instance of a perceptual schema is instantiated for the agent as a whole. The single instance is shared by any motor schemas or assemblages that use it. Consequently, even though several motor schemas might call a particular perceptual schema, the body of its perception routine is only executed once each iteration of an agent's think method.

4.5 Perception Mechanisms

To be as realistic as possible, agents were only allowed to react to and use information that was visible to them (within their field of view). When developing agents for Half-Life, two common perception methods are usually employed. Entities (agents, markers, goals) are perceived by iterating over a global entity list maintained by the game engine, looking for entities of a desired type. For each entity of the relevant type, checks are made to ensure that the entity is visible to the agent from its current location in the environment. This is done by using game engine primitives to draw vectors from the agent's eye position in the 3D environment to the origin of the target entity. The value returned by the trace function is then examined to determine if the line encountered any obstructions before reaching the target. If no obstructions were hit, the agent's line of sight to the entity is clear. A check can also be made to determine whether the entity lies in the agent's current field of view using a game engine primitive designed for this purpose. If the agent has a clear line of sight and the entity lies in the agent's field of view the entity is considered visible.

Obstacle detection, on the other hand, is sonar-like and works by using game engine functions to draw lines forward from the agent's hips, shoulders and eyes, 40 units ahead and checking for impacts. If any trace forward of the agent hits an obstruction, then an obstacle is detected. It is also possible to determine whether the obstruction struck is another agent that can be moved by impact or a solid object such as a wall. The agent can also detect if the object can be jumped or ducked under by examining the various trace results. If, for example, the shoulder and head traces do not hit anything, but traces at hip or knee level hit something, then the object

can potentially be jumped over or jumped upon. If, however, the head or shoulder traces impact an obstacle, but the hip traces do not, then the agent can attempt to duck under the obstruction instead. Similarly, if the left side traces strike something, but the right side traces do not, then the agent is near the edge of an obstruction. In this case, the agent's best option is to move right to avoid it.

While I have adopted these two common methods used by many other game agent developers, I supplement these with an additional method of my own. In order to allow agents to perceive bottlenecks, I developed a technique that operates by sending out lines 20 units distant, 180 degrees around the agent. For any line that hits an object, a check is made in the opposite direction for a mirror image hit. If a hit in that direction also occurs, vectors perpendicular to the two walls so impacted are examined to determine if they are parallel to each other. This last check avoids misidentifying a corner as a doorway or hallway. Doorway/hallway detection is the most processor intensive perceptual schema due to the sheer volume of checks that must be made to reliably perceive a doorway. In a real world environment, it is likely that the agent would need to move fairly slow to allow itself time to properly recognize doorways/hallways in this manner.

4.6   Movement and Action

Agents' movements and actions are carried out by each agent's controller object. Each time the agent's think method is called, the controller obtains a new desired yaw, pitch and roll as output by the currently active assemblage. The controller then compares the desired yaw, pitch and roll with the agent's current position and moves

the agent towards the ideal orientation. To be more physically realistic, I limit the agent's yaw change to 20 degrees each execution of an agent's think method.

Agent actions such as jumping, ducking, and readying a marker for deployment are also executed by the controller as dictated by the active assemblage. Certain actions require *locking* the agent's appropriate effectors, such as dropping a marker. Agent actions are divided into 3 basic types, including *arm*, *body* and *leg* actions. This approach allows agents to perform certain actions such as dropping a marker and jumping in parallel. It also handles the possibility of two different motor schemas requesting conflicting actions. When two or more motor schemas request actions that conflict, the action specified by the motor schema with the highest gain value is executed.

4.7 Marking the Environment

To allow agents to mark their environment, a pre-existing game weapon type known as *satchels* were adapted to serve as the marker types described in Section 3.2 for stigmergic navigation. Satchels are normally explosive packages (in a regular Half-Life game) that are dropped on the ground and detonated by a player via a second operation at some later time. To make them suitable for stigmergic navigation, I modified these satchels so that they never explode and so that multiple satchels can be deployed simultaneously. Agents were also given an unlimited supply of these items. Thus, agents mark their environment by dropping satchels on the ground at a desired location. In order to support multiple marker types simultaneously, I added additional satchel types that agents are able to selectively deploy. To make these

varied marker types visually distinguishable to a human observer, I also adapted each marker type to use a different graphical representation (or model).

Several motor schemas were created to support stigmergy. These include motor schemas tailored to drop markers of particular types, including *drop-marker-goal, drop-marker-bottleneck, drop-marker-local-maxima* and *drop-marker-local-minima*. Each of these motor schemas cause agents to mark the environment by dropping a satchel of appropriate type under specific conditions. Each motor schema works in conjunction with a perceptual schema including *percept-drop-marker-goal, percept-drop-marker-bottleneck, percept-drop-marker-local-maxima* and *percept-drop-marker-local-minima* each of which is responsible for detecting when the agent should drop a marker of a certain type. The conditions under which these perceptual schemas return a true value vary depending on how they are used in the experiments to be described in the next chapter.

Similarly, several motor schemas were also implemented to cause agents to react appropriately to the presence of the various marker types employed. These include *move-to-marker-goal* and *move-to-marker-bottleneck*, each of which causes agents to move toward a selected visible marker (if any) by outputting a vector towards the marker that is combined with the vectors output by any other active motor schemas. The *avoid-marker-local-maxima*, on the other hand, alternates between moving the agent toward and away from a local maxima marker (as outlined in Section 3.4.2). Lastly, the *avoid-marker-local-minima* cause agents to jump in the air when coming within a certain range of these markers to allow agents to leap over low-lying obstacles. As well, each utilizes a corresponding perceptual schema (e.g. *percept-marker-goal*),

which detects and provides information about the location of markers (of a particular type) that are visible to the agent. Markers can also be made heterogeneous by encoding a numeric value detectable by a perceptual schema and used by interested motor schemas. In contrast, homogeneous markers do not encode information beyond their presence and corresponding effect on the agents.

For a list of motor and perceptual schemas used in this thesis, please see Appendix A.

The next chapter describes the results obtained in a number of experiments that were carried out using the agents, environment and stigmergic markers just described to study the impact of stigmergy on common local navigation difficulties.

Chapter 5

Experimentation

To demonstrate the efficacy of the stigmergic navigation methods described in Chapter 3, and explore combinations of these methods, a series of experiments were conducted that used stigmergic markers to improve multi-agent local navigation. Each of the techniques described in Chapter 3 was explored alone and in selected combinations.

The details of individual experimental conditions and results are presented in separate sections below. Prior to the descriptions of individual experiments, I present the elements that are common to all experiments: computing facilities employed, the environment inhabited by agents, and details of experimental controls and result formats.

Figure 5.1: Experimental Environment

4

5.1 Experimental Set-up

Experiments were run on a Pentium III - 933 MHZ computer, running Windows

2000 Pro Service Pack 3 with 512 MB RAM.

[4]Being a first-person game, the views here are all from the point of view of a human player observing the environment and the agents that inhabit it. The weapon in the foreground is a game artifact associated with the human player.

5.2   Scenario

The objective in all experiments was for a team of six agents to find a specific goal at an unknown location in their environment as quickly and as many times as possible in a 30 minute period. The 3D environment selected for experimentation was chosen because its layout and structural features presented elements common to indoor domains including open areas, doorways and corridors that can trap agents in local maxima and minima. Figure 5.1 shows a view from within the environment, which is a standard Half-Life map file entitled crossfire (packaged with the game as the map file crossfire.bsp).

Figure 5.2 provides a 2-dimensional overhead schematic view of the environment. As can be seen, this presents a more than sufficient challenge for a reactive navigation system. Beyond being of a significant size and having many opportunities for agents to head away from a goal, or in circles, the layout of the environment includes open areas, box-canyons, stairwells, ramps and elevation changes. Certain portions, such as other levels of the environment, have been omitted for clarity, but this diagram provides a sufficiently accurate representation for discussion purposes.

In order to accurately measure the impact of each marking method, all experiments and trials were executed under the same conditions: the environment, goal location and agent starting positions were consistent throughout all experimental trials.

Agents began each trial in either Room 6 or Room 7 (as depicted in Figure 5.2). Two starting rooms were used in all experimental trials, so that agents would not interfere with each other excessively due to overcrowding at the start of a trial. Since the two starting positions are positioned symmetrically, opposite each other, navigat-
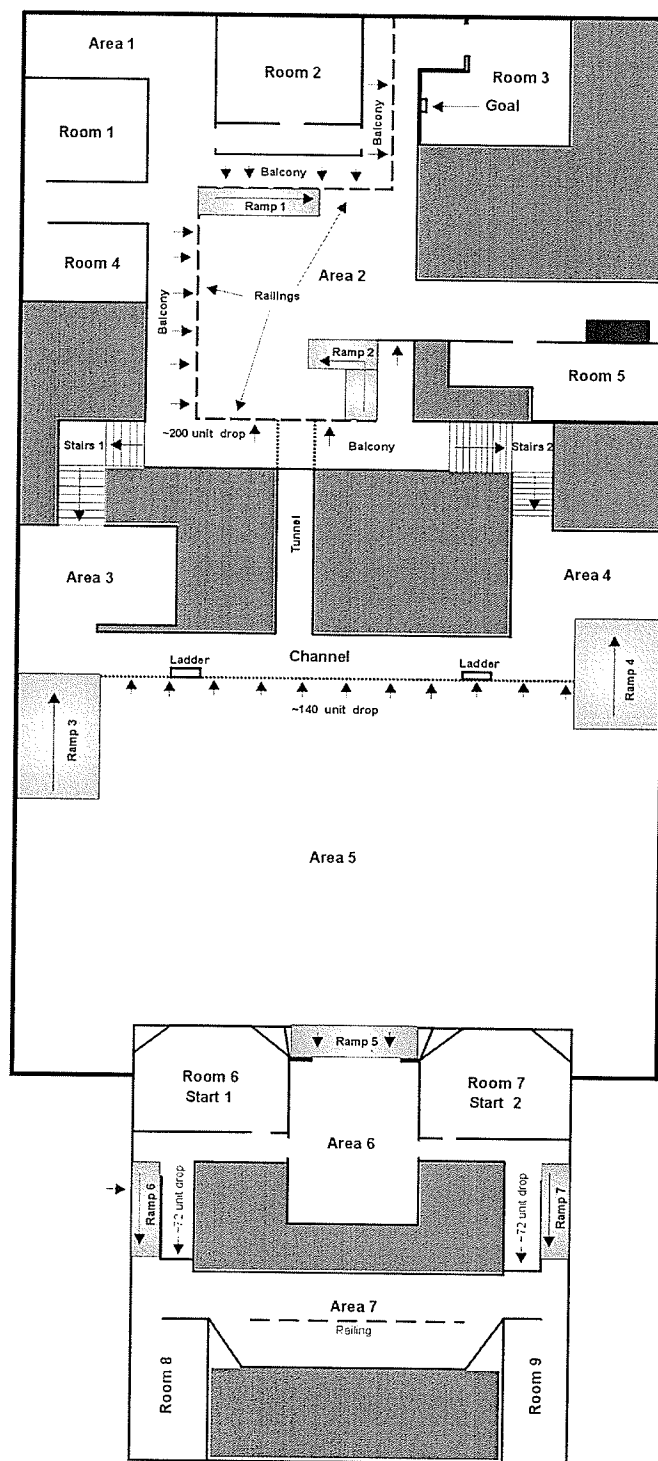
Figure 5.2: Map of Experimental Environment

ing to the goal from either is presumably no more or less difficult and should have no effect on results.

The objective for the agents in all trials was to navigate to a device located in a room on the far side of the environment (Room 3). As can be seen from Figure 5.2, agents had to navigate over ramps and stairs, and negotiate a number of doorways, open areas and box-canyons in order to find the goal. After locating and moving to within 40 units of the goal, a point was added to the running total of goals discovered in the trial. The discovering agent was then transported back randomly to either Room 6 or Room 7 to attempt to find the goal again.

Experimental results were gathered over the course of 40 contiguous trials, representing 20 hours running time for each experiment, in order to capture sufficient data for later analysis and comparison. In all experiments, the intent was to maximize the number of times the goal was discovered and minimize the time taken to initially locate the goal.

5.2.1   Result Recording

Global data was recorded for each trial, including the trial number and trial start date and time. Individual statistics were also tracked and recorded for each agent, including the time that the agent first entered into the environment, the agent's name, the number of times the agent reached the goal, the number of seconds an agent took to discover the goal for the first time (goal discovery time minus the starting time of the trial). In conjunction with this, the number of times this same agent's think method was invoked prior to initial goal discovery.

Additional processing overhead associated with some marker types can slow execution of the entire environment. As a result, agents using computationally more intensive behaviours are effectively operating with a reduced amount of time per trial, because the game time limit expires according to the system clock, regardless of the speed of game execution. In a physical implementation, if each agent was operating truly concurrently, this would not be a factor.

In order to compensate for this factor, I also tracked the total number of frames executed by the game engine in each set of 40 trials. The total frames executed is roughly comparable to the total number of times an agent's think method is invoked (as indicated in Section 4.4). This number was then divided by the total goals reached in the set of trials to arrive at a *frame count index*, that represented the average number of frames executed per goal. This is thus a measure of simulation time between goals. Consequently, it serves as a more accurate measure than real world time passed for the purposes of performance comparison between marking methods.

5.3   Experimental Results

Experiments were performed using the following combinations of stigmergic markers:

1. No Markers, in order to establish a control group with which to compare other results.

2. Bottleneck Markers, in order to illustrate the potential of this approach for dealing with box-canyon problems, as described in Section 3.4.1.

3. Local Maxima Markers, as described in Section 3.4.2.

4. Bottleneck and Local Maxima Markers, in order to assess their combined effect.

5. Bottleneck, Local Maxima and Local Minima Markers (as described in Section 3.5.1), in order to assess their combined effect.

6. Stigmergic Trail Markers, as described in Section 3.5.

7. Stigmergic Trail and Bottleneck Markers.

8. Stigmergic Trail, Bottleneck and Local Maxima Markers.

9. Stigmergic Navigation (i.e. the collection of all defined stigmergic mechanisms), in order to demonstrate the benefit gained using all of the above in combination.

10. Stigmergic Navigation and Teleautonomous Control, in order to illustrate the potential for a human to teleautonomously control a set of agents using my navigational mechanisms (as outlined in Section 2.5.1).

As can be seen from Figure 5.2, the location of the goal in room 3 did not present the possibility of a local minima situation occurring at the goal itself. As a result, local minima markers were only used in conjunction with the other marking methods rather than separately. This was because the other marker types form multiple intermediate goals, potentially creating additional local minima situations and so local minima markers were thought to be more suited to augmenting the other marker types (at least under these experimental conditions.).

Each set of experiments is described in separate subsections below, including the conditions under which agents dropped markers and manner in which agents reacted

to markers, along with an analysis of the observed results. Complete result listings are also provided in Appendix B.

5.3.1   No Markers

To establish a baseline for comparison with the various marking schemes, a set of experiments was executed with a team of agents that did not employ stigmergic markers. Agents instead relied solely on random search to locate their goal.

Results

The team of agents using no markers discovered the goal using random search 161 times in 40 trials for an average of 4.03 goals per trial (with a standard deviation for goals reached in a trial of 1.67). It took them on average 592.35 seconds (with a standard deviation of 362.25 seconds) and an average of 57982.88 think method invocations to initially discover the goal. The agents' best performance in locating the goal repeatedly in a single trial was 8. The best time to initial discovery was 89.98 seconds over 8820 think method invocations. The total frame count execution after 40 trials was 7045454, which averages to 176136.35 frames per trial. The frame count index was 43760.58, indicating that was the average number of think method invocations to reach the goal.

5.3.2   Bottleneck Markers

A second set of experiments was executed with agents that employed bottleneck markers (described in Section 3.4.1), to lead agents out of box-canyons (described in

Figure 5.3: Agent moves toward a bottleneck marker

Section 2.3). Figure 5.3 shows an agent navigating toward a bottleneck marker.

The *percept-drop-marker-bottleneck* perceptual schema was configured in this experiment to return a true value whenever the agent was standing in a doorway or a hallway, unless a marker could already be seen within 60 units (as defined in Section 4.2) of the agent's current position. The *percept-drop-marker-bottleneck* perceptual schema was used to activate the *motor-drop-marker-bottleneck* motor schema, which caused the agent to drop a marker on the ground at its current position.

Another perceptual schema, *percept-marker-bottleneck*, was also configured to maintain a dynamic list of markers visited by the agent. These visited markers were

purged from the short-term list after 30 seconds had elapsed between the agent's visit to the marker. A marker was considered to be visited when the agent was standing within 40 units of the marker. This was intended to discourage cyclic behaviour and cause the agent to proceed along a trail of markers more deliberately (as described in Section 3.4.1). Otherwise, an agent might move back and forth between markers if it happened to turn around and see a previously visited marker again.

The *motor-move-to-marker-bottleneck* motor schema was also configured to cause the agent to move toward a visible bottleneck marker until it was within 40 units of the marker. At the point that the first marker was reached, the motor schema continued to move the agent toward any subsequent markers as long as the agent detected that it was still in a doorway or hallway. As soon as the agent detected that it was no longer in a narrow passageway (after reaching a marker), the *motor-move-to-marker-bottleneck* motor schema was suspended for a period of 30 seconds (during which any markers in its list of visited markers are gradually purged).

Results

The agents in this set of experiments located the goal 278 times in 40 test runs for an average of 6.95 goals per trial, with a standard deviation of 2.59. On average these agents discovered the goal for the very first time within 463.67 seconds (with a standard deviation of 241.87 seconds), taking 43917.58 think method invocations to do so. The minimum time to initially locate the goal in any trial was 211.86 seconds and 20074 think method invocations. The most goals reached in a single trial was 12.

As can be seen, marking bottlenecks resulted in improvements in the total goal
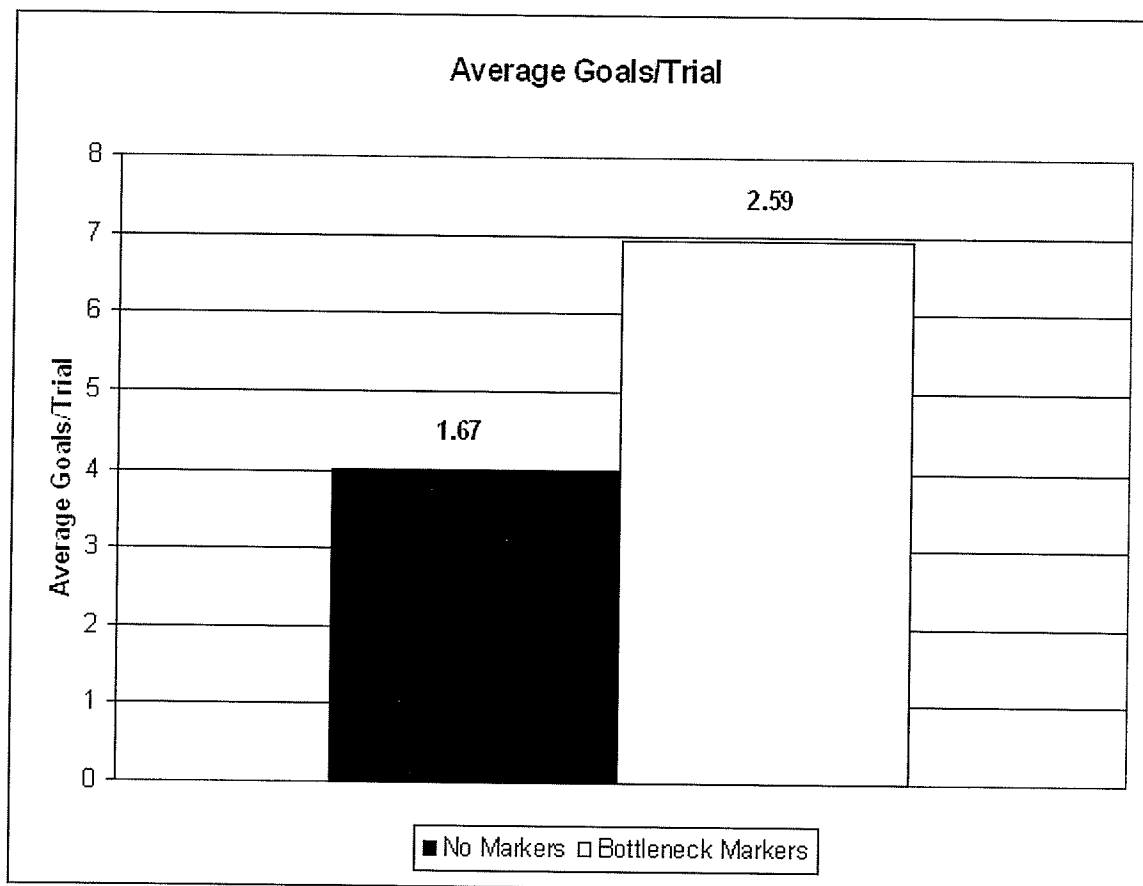
Figure 5.4: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

count, the average goals/trial, and even the average time to initially locate the goal. The latter is desirable, of course, when time is a factor, such as searching a building for survivors after a disaster to get them medical attention quickly. The average time to locate the goal for the first time by any agent in the group was reduced by 21.72 percent. The average number of think method invocations to discover the goal was reduced by 24.26 percent.

The average goals discovered per trial was increased by 72.67 percent. Accordingly, the total goals for all trials was increased by the same percentage. Figure 5.4

illustrates these findings in comparison to agents that did not employ markers. In addition, the team of agents using bottleneck markers located the goal more often in the majority of trials (as evidenced by the results listed in Section B.2).

On the other hand, the *best* time to goal discovery across *all* trials was somewhat more than that of agents using no markers. This was not unexpected, as the agents using markers were stabilizing their environment. In so doing, their performance was made more consistent rather than ranging from good to bad based on chance. This is indicated in that the standard deviation for initial goal discovery was 241.87, indicating that the time to initial discovery of a goal using markers was somewhat more consistent than without markers.

The total frame count execution after 40 trials was 6829138, which averages to 170728.45 frames per trial. This reduction in overall frames per trial (compared to agents that did not use markers) occurs because the agents using markers are doing more work each frame. This is also evidenced by the lower frame count index (the average number of think method invocations to reach the goal) of 24565.24. This is a 43.86 percent reduction in the average number of think method invocations per goal compared to agents using no markers.

The standard deviation for goals per trial, at 2.59, was wider than that of agents not employing markers. The increased variability in goals is perhaps due in part to the fact that these markers are dropped without the agents knowing where the goal is located in the environment. As a result, the possibility exists that many (or all) of the agents might be initially led in a completely incorrect direction, slowing their time to find the goal. However, since a marker no longer affects an agent once the

agent has visited it (that is, once the agent comes within a few units), the effect of being misled would be temporary. Regardless, it is equally probable that the marker would in fact lead the agent in a better direction than a worse one (accounting for the variability in goals found).

Regardless, with an average of 6.95 goals per trial, a standard deviation of 2.59 indicates that finding less than 4.36 goals (0.33 higher than the average using no markers) in a trial would be unusual. So, while their effect on finding a single goal location in the environment still depends somewhat on chance, they can be profitably used in situations where maximal map coverage in minimal time is desired.

5.3.3 Local Maxima Markers

The next set of trials was executed using a marker type that is used to identify local maxima as described Section 3.4.2. Agents were compelled to drop a local maxima marker at their current position when the following conditions were met:

1. No goal was visible

2. No walls were nearby (within 160 units)

3. No local maxima markers were already present within a certain range (in this case 500 units).

Agents alternated between moving toward local maxima markers and avoiding them. When first perceiving a local maxima marker, while in a local maxima marker-affinitive state, an agent moved up to the marker's position. Once the agent physically reached the marker (moved within 40 units), the agent transitioned to avoiding local

maxima markers. The local maxima marker *avoidance* state persisted for 30 seconds, during which the agent had an opportunity to find an exit from its current area (since it had nothing interesting in it). To allow for a certain freedom of movement, agents are only repelled by local maxima markers when they are in within 120 units. This has the effect of keeping them to the edge of the area they are in and closer to any exits from it, without trapping them against one particular side. If the agent does not find an exit, it is eventually drawn back into the open to try again upon becoming local maxima marker-affinitive again.

As indicated in Section 3.4.2, local maxima markers are designed to disappear after a set time period. In this case, local maxima markers disappeared after 150 seconds.

Results

The agents in this experiment discovered the goal a total of 224 times in 40 trials, for an average of 5.6 goals per trial (with a standard deviation of 1.97). These findings in comparison with previously presented methods are shown in Figure 5.5. The most goals found in a single trial was 11. The average time taken to discover the goal for the first time was 533.48 seconds (with a standard deviation of 242.33) over 52135 think method invocations. The best time to reach the goal over all trials was 173.77 seconds over 17029 think method invocations. As an examination of the results listed in Section B.3 shows, local maxima markers improved performance over using no markers in the majority of trials.

In addition, the time to initially locate the goal was improved on average by 9.94
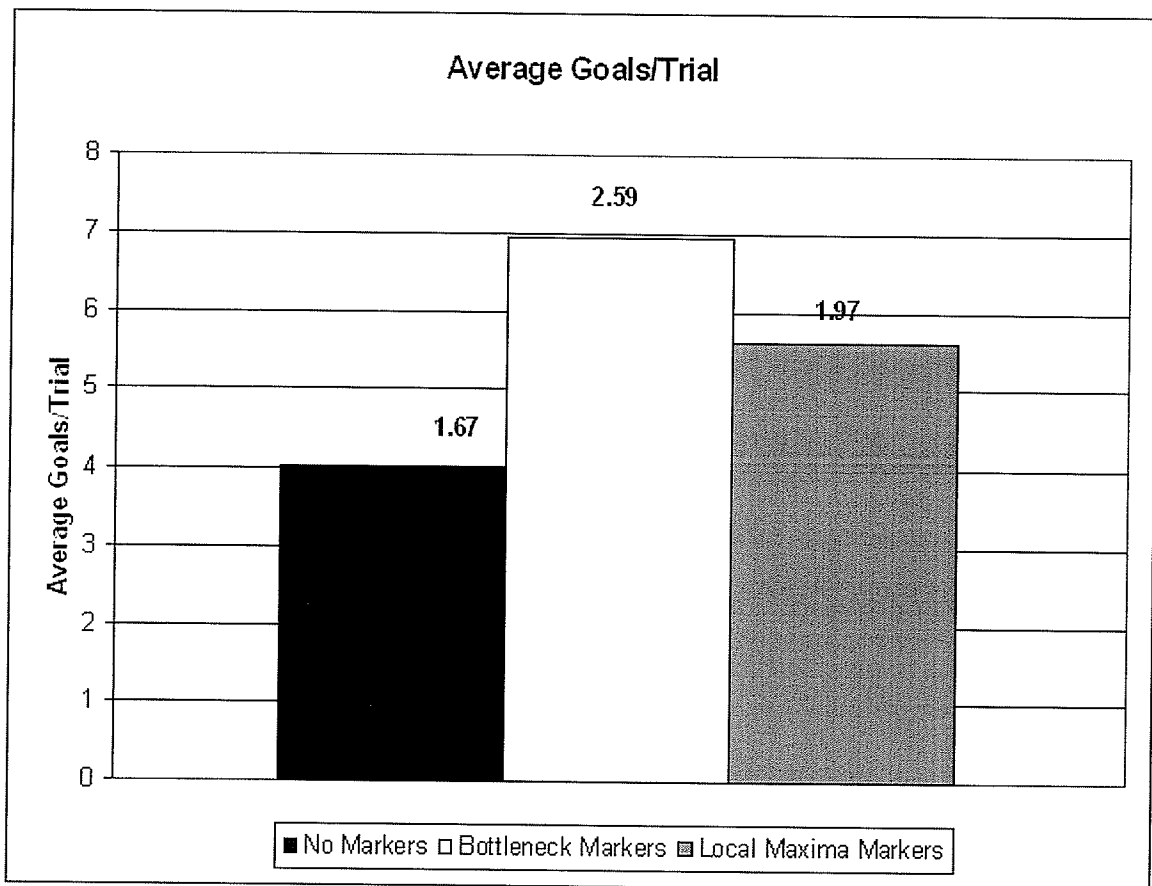
Figure 5.5: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

percent. The average number of think method invocations to discover the goal was reduced by 10.09 percent. The number of goals discovered in total was improved by 39.13 percent over no markers. This was not as strong an improvement as gained by marking bottlenecks, which posted a 21.72 and 72.67 percent improvement in time and goals respectively. However, this is not surprising, since local maxima markers work more indirectly, by pushing agents away from uninteresting places rather than actively drawing agents to new regions.

As well, the total frame count for all 40 trials was 7026189 (an average of 175654.73 per trial). This was only slightly less than that of agents using no markers, which is not unexpected as the computational overhead of detecting whether to drop a local maxima marker is fairly low, particularly with respect to bottleneck detection. A frame count index of 31366.92 was improved over agents using no markers, but not quite as strong as that gained using bottleneck markers.

5.3.4 Bottleneck and Local Maxima Markers

The next set of experiments employed the bottleneck and local maxima markers described above in combination. Agents dropped and reacted to bottleneck markers and local maxima markers as described in Section 5.3.2 and Section 5.3.3 respectively. Agents followed the additional restriction of not dropping local maxima markers within 150 units of bottleneck markers. Agents were configured to value bottleneck markers more strongly than local maxima markers via a higher gain value, so that the presence of a local maxima marker would not prevent them from moving to a simultaneously visible bottleneck marker.

Average Goals/Trial

Figure 5.6: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

Results

A team of agents employing both bottleneck and local maxima markers in combination located the goal 305 times in 40 trials for an average of 7.63 goals per trial (with a standard deviation of 3.04). The performance of this combination of markers is illustrated in Figure 5.6. The highest number of goals in a single trial was 13. The best time to initial discovery was 244.12 seconds with an average time taken to discover the goal for the first time at 567.02 seconds (with a standard deviation of 222.61 seconds) or 51697.6 think method invocations.

As can be seen, using both marker types in combination yielded only a 4.28 percent improvement in initial discovery time over all. However, this is partially attributable to the additional processor overhead incurred by using both marker types. This is evidenced by the fact that the average number of think method invocations to discovery was improved by 10.84 percent. This is slightly improved over that of local maxima markers alone, but significantly less than that of bottleneck markers used in isolation. Yet, using both marker types resulted in a higher number of goals being discovered than using just bottleneck or local maxima markers alone. The average goals per trial was increased over no markers by 89.44 percent using both marker types. This is compared to 72.67 percent for bottleneck markers and 39.13 percent for local maxima markers.

Interestingly, the results listed in Section B.4 show that using both these marker types resulted in fewer goals in 7 trials. In contrast, using bottleneck markers exclusively resulted in a lower number of goals than using no markers in only 3 trials. That using both marker types resulted in a greater number of goals being discovered is perhaps because it resulted in a number of significantly higher results in a number of trials. While this method had results in excess of 10 goals in 10 trials, the bottleneck method posted only 6 results higher than 10. Consequently, using both methods resulted in the better performances outweighing the poorer (relative to bottleneck markers alone).

This combination of markers resulted in a lower frame count than previous methods. In 40 trials, the environment state was updated 6598645 for an average of 164966.13 think method invocations per trial. This is approximately 11000 fewer

think method invocations per trial, which indicates that this method still performed better with a shorter operating period. The frame count index using this marking method was 21634.9 and better than all marking methods covered above.

5.3.5 Bottleneck, Local Maxima and Local Minima Markers

The next set of trials included bottleneck, local maxima and local minima markers in combination. Unlike the other types of markers, local minima markers did not attract or repel agents, but instead compelled them to jump in the air when sufficiently close to them. This approach can be useful in the experimental environment in situations where the agent can leap over any low-lying obstacles in its path. In this environment, such obstacles were the most obvious type of local minima. For example, the balconies overlooking Area 2 of Figure 5.2, are lined by railings that can block agents attempting to move toward markers lying in area 2. In particular, agents coming up the stairs from Area 3 can sometimes see an attractive marker placed by another agent in Area 2 below them. However, when the agent attempts to move toward the attractive marker, the railing blocks the agent's path. In order to handle this situation, it is possible for an agent to jump over the railing rather than remaining stuck or even proceeding the long way around the railing.

As indicated in Section 3.5.1, local minima markers are designed to disappear after a set time period. As with local maxima markers, this time period was set to 150 seconds.

Average Goals/Trial

Average Goals/Trial (y-axis, scale 0 to 9)

Bars with values atop: 1.67, 2.59, 1.97, 3.04, 3.03

Legend:
- ■ No Markers
- □ Bottleneck Markers
- ▣ Local Maxima Markers
- □ Bottleneck and Local Maxima Markers
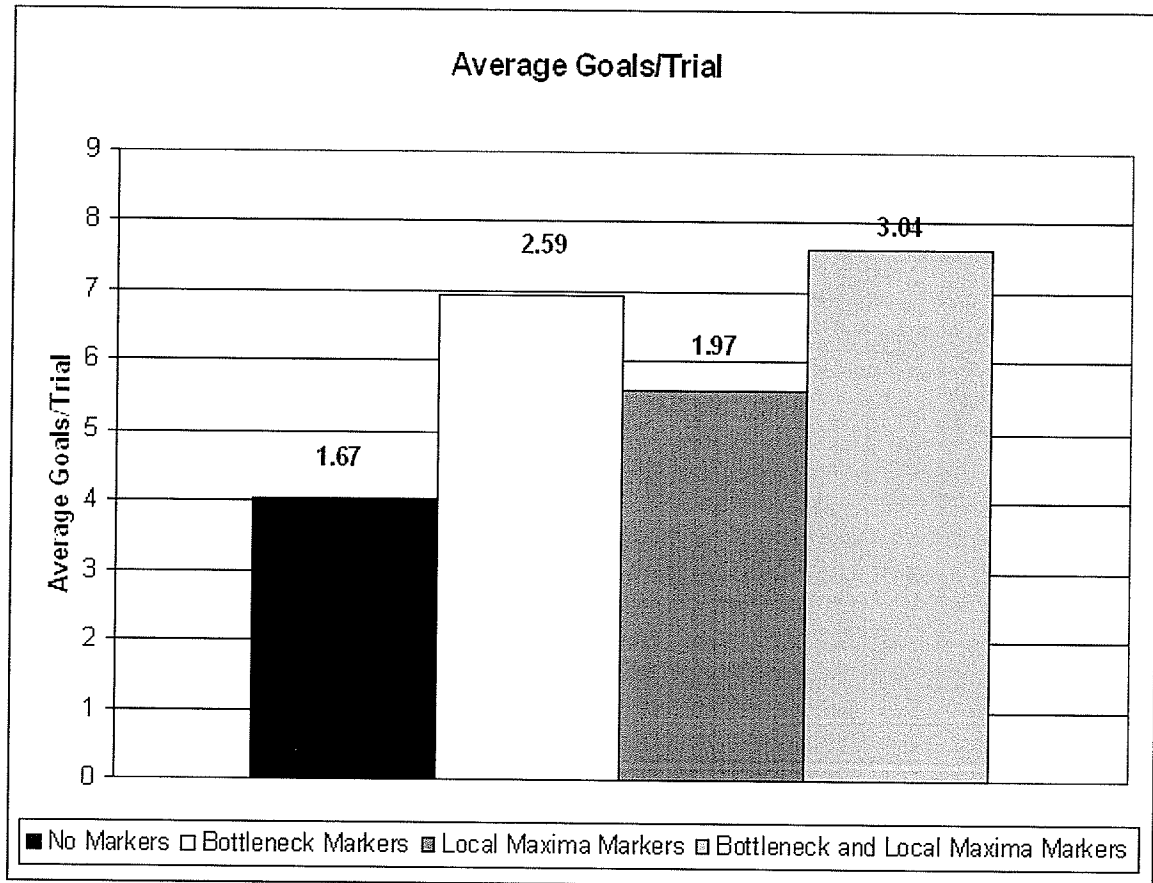- ▦ Bottleneck, Local Maxima and Local Minima Markers
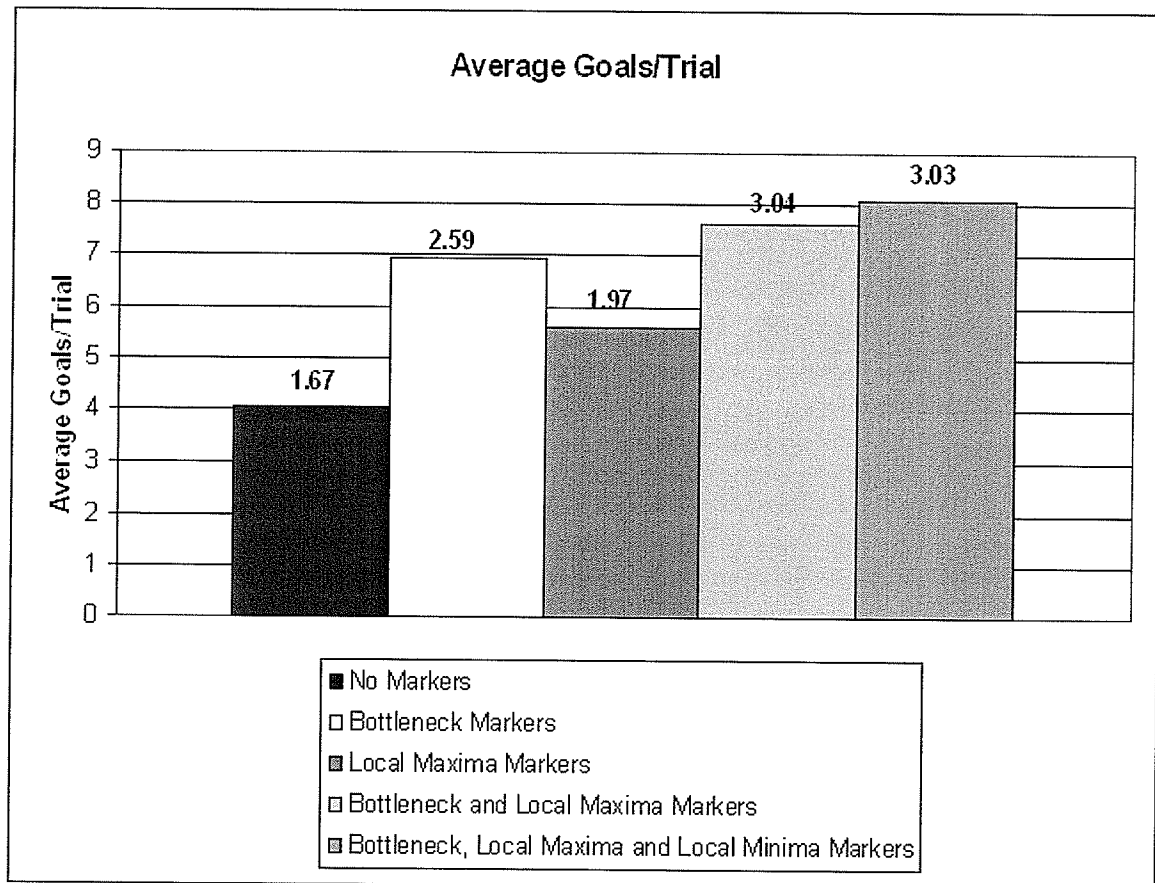
Figure 5.7: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

Results

Using bottleneck, local maxima, and local minima markers in combination resulted in 323 goals in 40 trials for an average of 8.08 goals per trial (with a standard deviation of 3.03. As Figure 5.7 shows, this was a significant improvement over agents using no markers. It was also somewhat improved over that of the other marking techniques described above. The average time taken to initially discover the goal in a trial was 467.67 seconds (with a standard deviation of 242.47 seconds) in 24814.38 think method invocations.

The total frame count for these trials was 3774499 or an average of 94362.48 frames per trial. This is substantially less than that of previous marking methods, indicating that agents are accomplishing more work in the same time as compared to other methods, which underscores its effectiveness. This is also evidenced in that the frame count index in these trials was 11685.76, the lowest thus far by a significant margin.

5.3.6 Stigmergic Trail Markers

In this set of experiments, agents dropped and used goal markers according to the rules described in Section 3.5, always moving toward the lowest numbered marker perceptible.

Results

As illustrated by Figure 5.8, stigmergic trail markers increased performance substantially over agents using no markers and that of all marking techniques described above. The average goals per trial for a team of agents using these markers resulted in more than 10 times the number of goals than a team of agents using no markers at all.

A team of agents using stigmergic trail markers located the goal 1828 times in 40 trials for an average of 45.7 goals per trial, with a standard deviation of 48.14 goals. On average they located the goal for the first time in 462.19 seconds of first appearing in the environment, with a standard deviation of 229.9 seconds. The most goals reached in a single trial was 182, and the fastest time to discover the goal was

Average Goals/Trial

Figure 5.8: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

103.62 seconds or 10213 think method invocations for the agent that discovered the goal.

The total frame count for these trials was 6984317 for an average of 174607.93 frames per trial. This results in a frame count index of 3820.74, substantially lower than any of the marking methods presented so far.

The result was that a trail of markers was gradually built up, working backward from the goal to the initial starting point after the goal was first discovered. Consequently, subsequent agents were eventually able to follow a trail of markers directly to

the goal almost from the moment they appeared at the starting point. The eventual result was a more or less direct parade of agents heading to the goal. Indeed, this method exceeded the performance of agents using no markers in *all* trials and that of the combination of bottleneck and local maxima markers in 36 out of 40 trials.

As one would expect, the greatest factor affecting performance in these trials was how quickly the marker trail gets created. Once the marker trail was in place, the agents were able to follow it repeatedly and consistently until time expired. The poorer performances only resulted when this trail was not constructed early enough for the agents to benefit. When the agents failed to locate the goal early enough or frequently enough, their performance was comparable to using no markers at all. Yet, once the trail extended far enough back from the goal, the agents rapidly outpaced all others methods by a wide margin. Undoubtedly, the agents' poor performances would have been greatly improved either by increasing the number of searching agents or the time period (relative to another method operating over the same amount of time).

Interestingly, the average time to initial goal discovery was also improved by 21.97 percent over agents using no markers. One possible reason for this is that the first stigmergic trail marker dropped ensured that the agent discovering the goal did not wander inadvertently back out of the room in which the goal is situated.

I frequently observed the situation where an agent entered the goal room, perceived the goal and headed back out. This was usually due to the influence of the *motor-noise* motor schema, which adds some randomness to an agent's movements (for reasons described in Section 2.3). It also occurred when the angle of the agent's approach

caused the *motor-avoid-static-obstacle* motor schema to activate and steer the agent away from the goal and lose sight of it.

However, with stigmergic trails, the agent drops a stigmergic trail marker upon first seeing the goal, allowing it to recover more quickly. This is because even after wandering out of the goal room, the agent can still see the trail marker it just dropped and is drawn back into the room directly. Without the benefit of the trail marker, the agent is forced to rely on its chance wanderings to bring it back into the room.

5.3.7 Stigmergic Trail and Bottleneck Markers

The next set of trials used both stigmergic trail markers and bottleneck markers to observe their combined effect.

Results

Combining stigmergic trail markers with bottleneck markers improved performance dramatically over using no markers and even over that of stigmergic trail markers used alone. This can be seen in Figure 5.9, which compares the average goals per trial using stigmergic trail and bottleneck marker to all previously described marking techniques. In addition, an examination of the results listed in Section B.7 shows that this combined approach yielded better results than no markers in 39 of 40 trials. At the same time, it resulted in a higher number of goals than stigmergic trails alone in 25 of 40 trials. This is also shown in that a total of 2253 goals were reached across all trials for an average of 56.33 goals per trial. The standard deviation for goals reached per trial was 46.96.
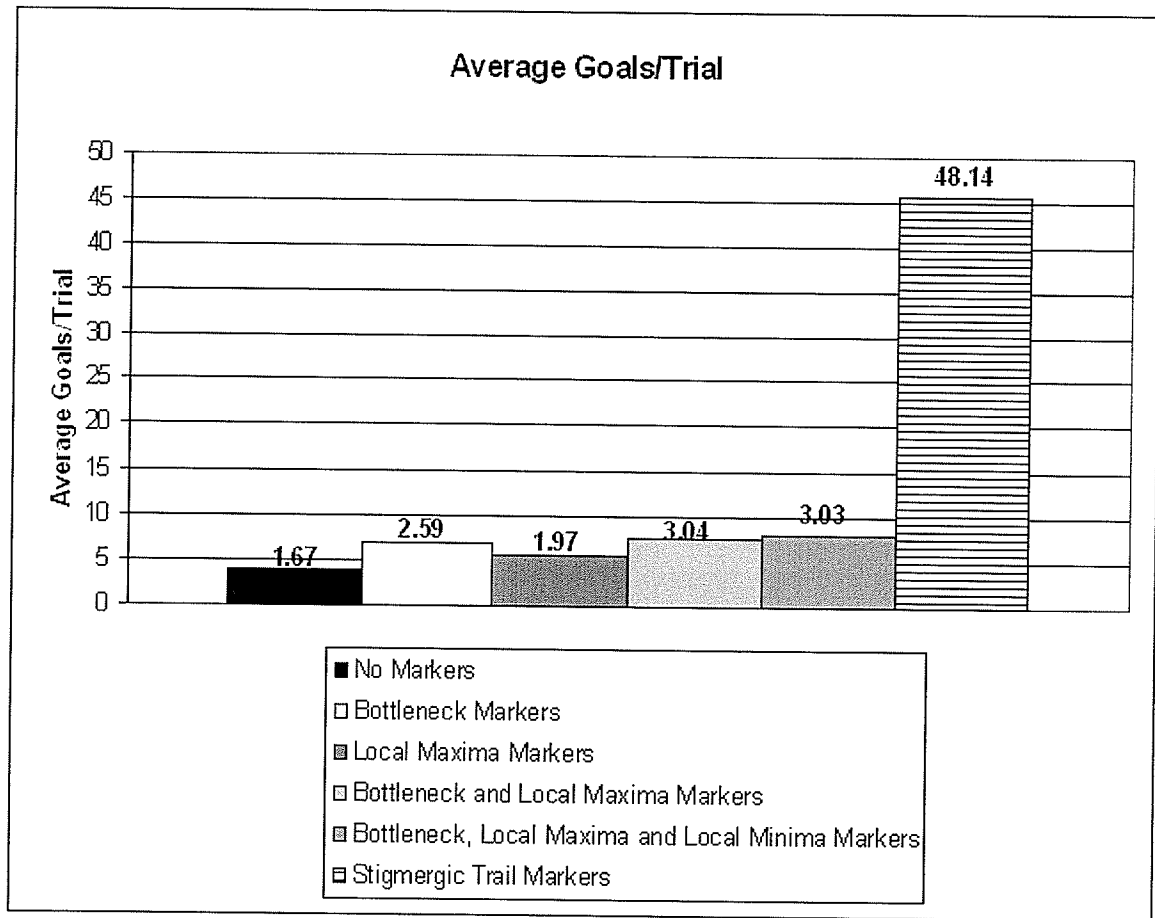
Figure 5.9: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

The average time to discover the goal for the first time was 508.15 (with a standard deviation of 259.44) over an average of 46896 think method invocations. This is a 14.21 percent decrease in time and 19.12 percent decrease in think method invocations to discover the goal compared to using no markers. Though stigmergic trails alone posted a higher improvement in time to initially locate the goal, this is partially due to the increased computational load incurred using both marking methods in combination. This is shown in that average number of think method invocations for this method is much closer percentage-wise to that of stigmergic trails than the average time. It is also evident in that the total frame count for these trials was 6724598 for an average of 168114.95 frames per trial and a frame count index of 2984.73, lower than that of stigmergic trails alone.

Regardless, using both markers in combination resulted in 13 times (1299 percent) more goals being located on average per game versus agents not using markers (compared to 10 times more on average using stigmergic trail markers by themselves). The improvements gained here over using stigmergic trail markers alone are due to the bottleneck markers promoting greater coverage of the environment by drawing agents to different areas periodically. Once the stigmergic trail has lengthened a certain distance back from the goal, the faster other agents notice the trail the better. Bottleneck markers help in this regard by drawing agents back and forth from room to room more regularly than they do when wandering randomly. Thus, once the trail is sufficiently long, other agents see its end sooner and the complete trail gets built earlier as a result. Since the trail gets built earlier, the agents have a longer period of time in which to benefit from it, resulting in a higher performance overall.

5.3.8 Stigmergic Trail, Bottleneck and Local Maxima Markers

The next set of trials included stigmergic trails, bottleneck and local maxima markers in combination.

Results

The combination of these 3 marker types resulted in 3817 goals being reached in total over all 40 trials. This is 22.7 times more goals than a team of agents using no markers and an average of 95.43 goals per game (with a standard deviation of 66.64 goals). This substantially exceeds the 56.33 average of stigmergic trail and bottleneck markers in combination, as can be seen in Figure 5.10.

The average time to initially discover the goal in a trial was 525.3 seconds (with a standard deviation of 291.27 seconds) over an average of 30869.03 think method invocations. This is an 11.32 percent reduction in average time to locate the goal in a trial. More significantly, it is a 46.76 percent reduction on average in the number of think method invocations required to locate the goal. The large difference in these percentages is primarily due to the additional processor load incurred by using all 3 marking methods in combination.

As shown in Figure 5.10, using these 3 marker types together resulted in a higher number of goals in 37 out of 40 trials. At the same time, it resulted in a higher number of goals than using just stigmergic trail and bottleneck markers in combination in 27 out of 40 trials. The frame count index for this marking method was somewhat improved over that of stigmergic trail and bottleneck markers by themselves at 1219.89.

Average Goals/Trial

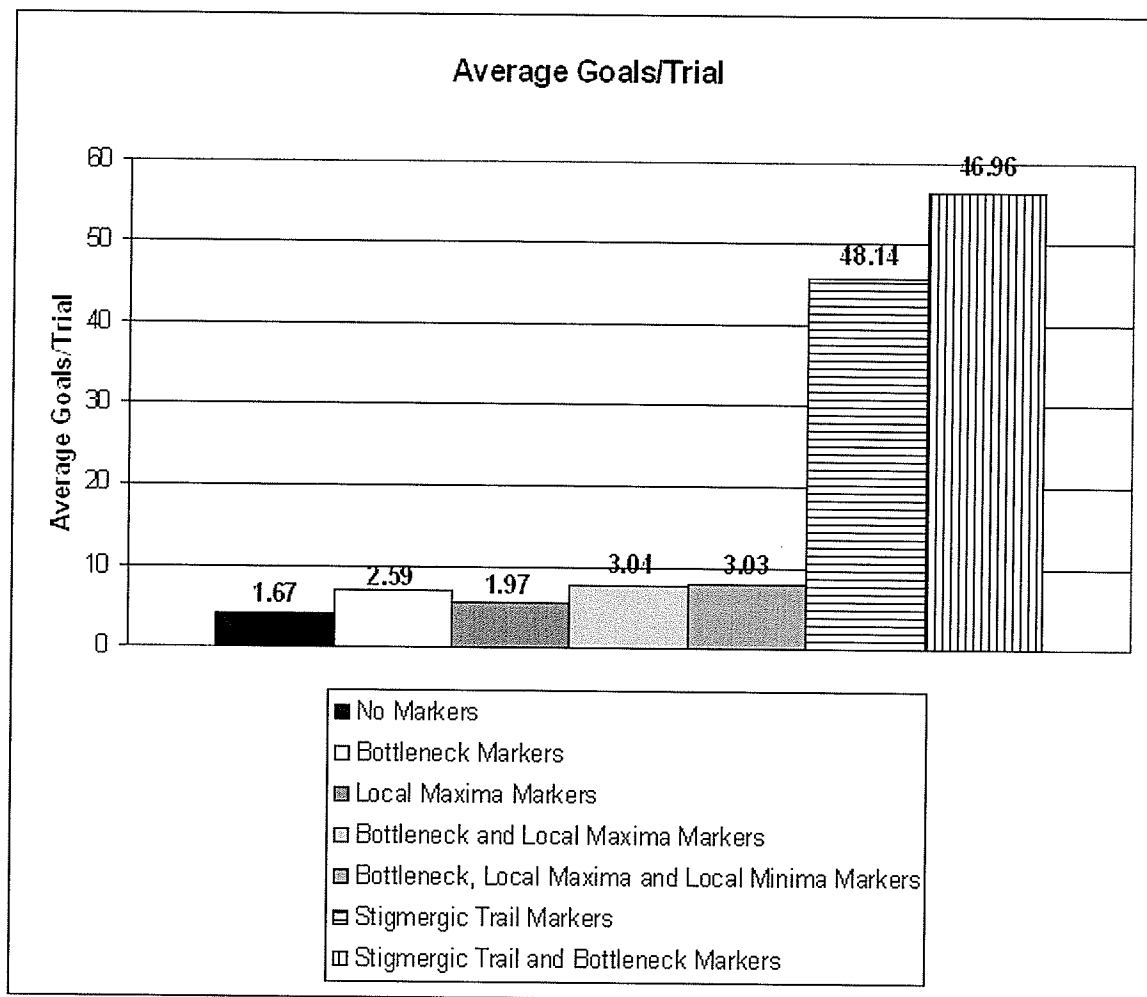Figure 5.10: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

This was based on a total frame count of 4656317 and average frame count per trial of 116407.93

As with the previous experiment, the improved performance using the three marker types in combination is due to the stigmergic trail being constructed and extended faster than otherwise. The addition of the local maxima markers promotes even wider and more frequent coverage of the environment than bottleneck markers by themselves (as evidenced by the experiment described in Section 5.3.4). Local maxima markers have the additional effect of drawing agents into the open initially where they can see a wider portion of their surroundings. This also increases the likelihood that one or more agents will perceive a trail end sooner.

These effects in combination contribute to the stigmergic trail being established earlier in the trial, allowing the team to benefit from its use for a longer period of time. This is also evidenced in that this combination of marker types resulted in a best performance in a single trial: reaching the goal 247 times. This is significantly higher than any previous marker combinations.

5.3.9   Stigmergic Navigation

This experiment used all four marker types introduced in Section 3.2 in combination, including bottleneck, local maxima, stigmergic trail markers and local minima markers.

Results

Using all four marker types in combination resulted in 3856 goals being reached in 40 trials for an average of 96.4 goals per trial (with a standard deviation of 60.62 goals). This is almost 23 times more goals than agents using no markers. The average time to locate the goal for the first time in a particular trial was 490.11 (with a standard deviation of 262.24) in 24814.38 think method invocations. The former is a 17.26 percent decrease (compared to agents using no markers) in the average time to initially locate the goal. Even more strongly, the latter is a 57.2 percent decrease in the average number of think method invocations required to initially locate the goal (compared to agents using no markers).

While this was a substantial improvement over agents that did not use markers, it is quite similar to the results obtained using bottleneck, local maxima and stigmergic trail markers without local minima markers. This is shown in Figure 5.11, where the bars identifying the average goals per trial between the two latter two methods are nearly the same.

There are several likely reasons that local minima markers did not result in a substantial improvement. To begin with, the situation that the local minima markers are intended to handle appears in only a small portion of the environment. Consequently, the opportunities for them to have a positive affect on the agents' performance is limited. Moreover, the agents themselves are fairly well-equipped to deal with local minima via introduction of noise into their movements as part of their behaviour-based design. Finally, the other marker types can themselves reduce the frequency of local minima in this experiment.

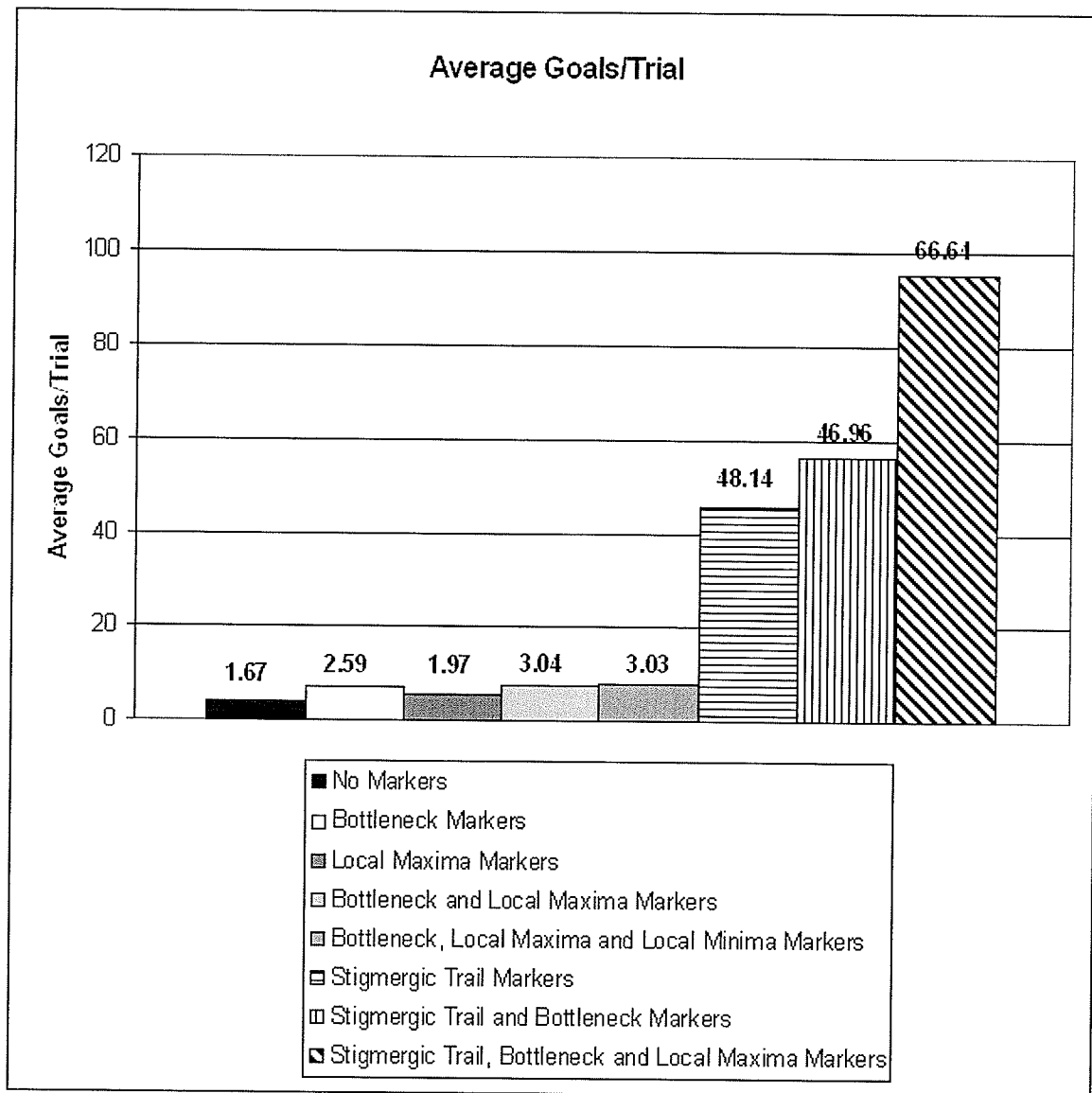Figure 5.11: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

Observations of the agents in a number of experiments showed that the stigmergic trail leading to the goal was typically built along a path that bypassed the upper balcony where local minima are known to appear. Instead, the trail typically takes the agents across Area 5 (see Figure 5.2) to either Ramp 3 or 4, down the channel near the center of the map, down the tunnel connecting it to Area 2 before proceeding down the last hallway before the goal. As a result, the occasional positive impact of the local minima markers did not have a substantial effect on overall performance.

Interestingly, the total frame count using all four marker types, at 4285980 (an average of 107149.5 per trial), was not the lowest frame count overall. Rather the lowest frame count of all marking methods was that using bottleneck, local maxima and local minima markers in combination. The reason for this seeming contradiction is that once stigmergic trails are in place and visible to the agents, they forgo the other marker types in favor of the stigmergic trail markers. As a result, agents perform less computational processing during this time. The frame count index bears out the superior performance of stigmergic navigation in that it had the lowest frame count index of all methods at 1111.51.

5.3.10 Stigmergic Navigation and Teleautonomous Control

This last experiment involved having a human assist the team of agents in locating the goal by dropping local maxima and bottleneck markers. The agents themselves employed all four markers types, but the human-controlled player was limited to merely guiding the agents via local maxima and bottleneck markers.

The expectation was that a human would be able to place these markers more

quickly and intelligently, to lead the agents more quickly to the goal. This in turn was expected to allow the agents to construct a stigmergic trail more quickly and perform somewhat better as a result. The finding that a human could do better at controlling the agents than the agents themselves would hardly be surprising. However, this is intended as a demonstration in principle that stigmergic markers can be successfully used as a means for implementing teleautonomous control.

Unlike previous experiments, only a small number of trials were executed to explore the potential impact that human level intelligence might have in laying these basic marker types. Also, requiring a human to endure 20 hours of trials, in accordance with previous experiments was deemed to be impractical and unnecessary to achieve its purpose.

Results

In four trials, a team of agents assisted by a human-controlled agent located the goal 735 times for an average of 183.75 goals per trial (with a standard deviation of 9.77 goals). The average time to initially locate the goal in a particular trial was 118.05 seconds (with a standard deviation of 17.54 seconds) over 5125.75 think method invocations.

Though executed over a smaller number of trials, these results indicate that the agents benefited significantly from teleautonomous guidance (as shown in Figure 5.12). The number of goals reached on average was approximately double that of the best performing marking techniques described herein. The time required to initially locate the goal was even more improved. What was interesting about these

Average Goals/Trial

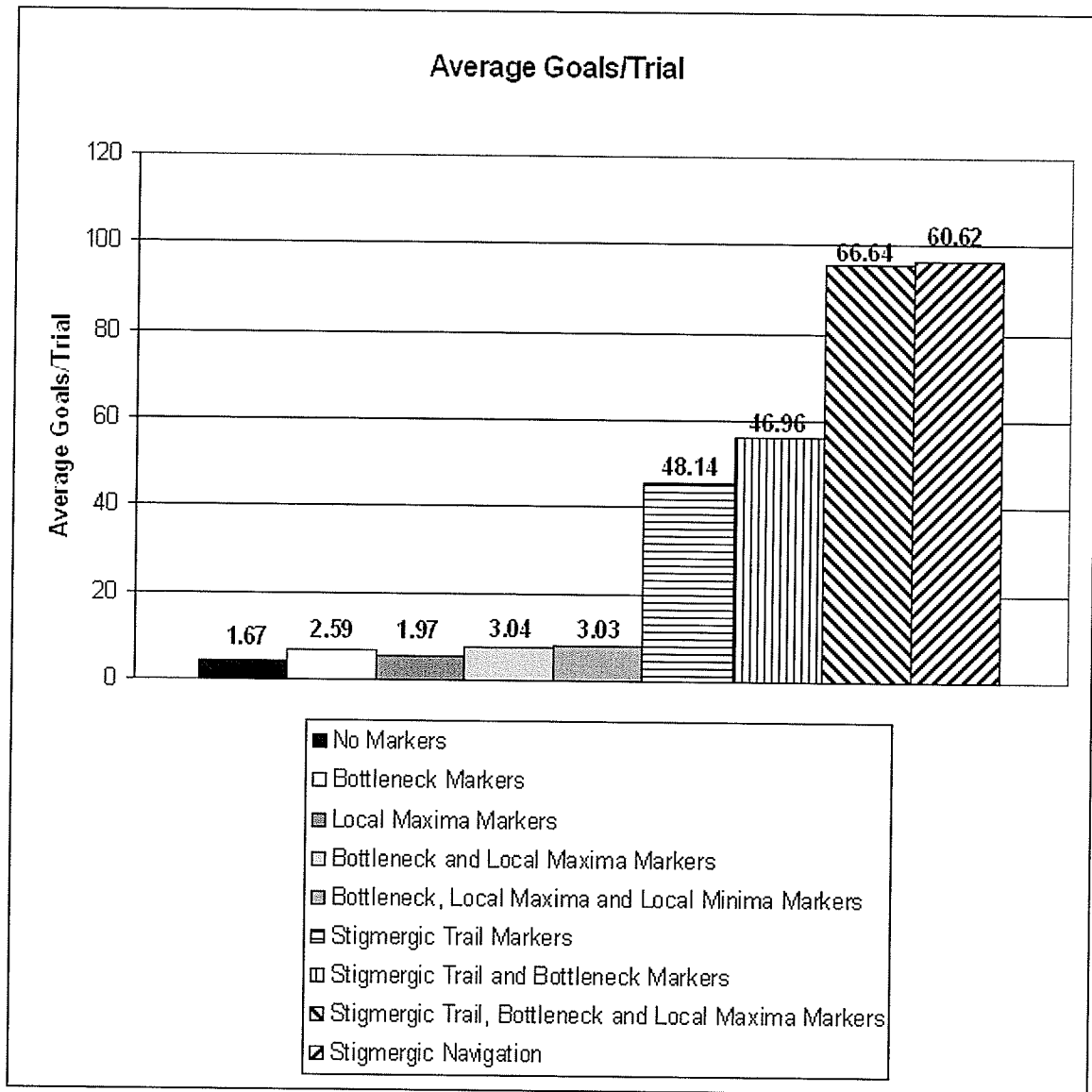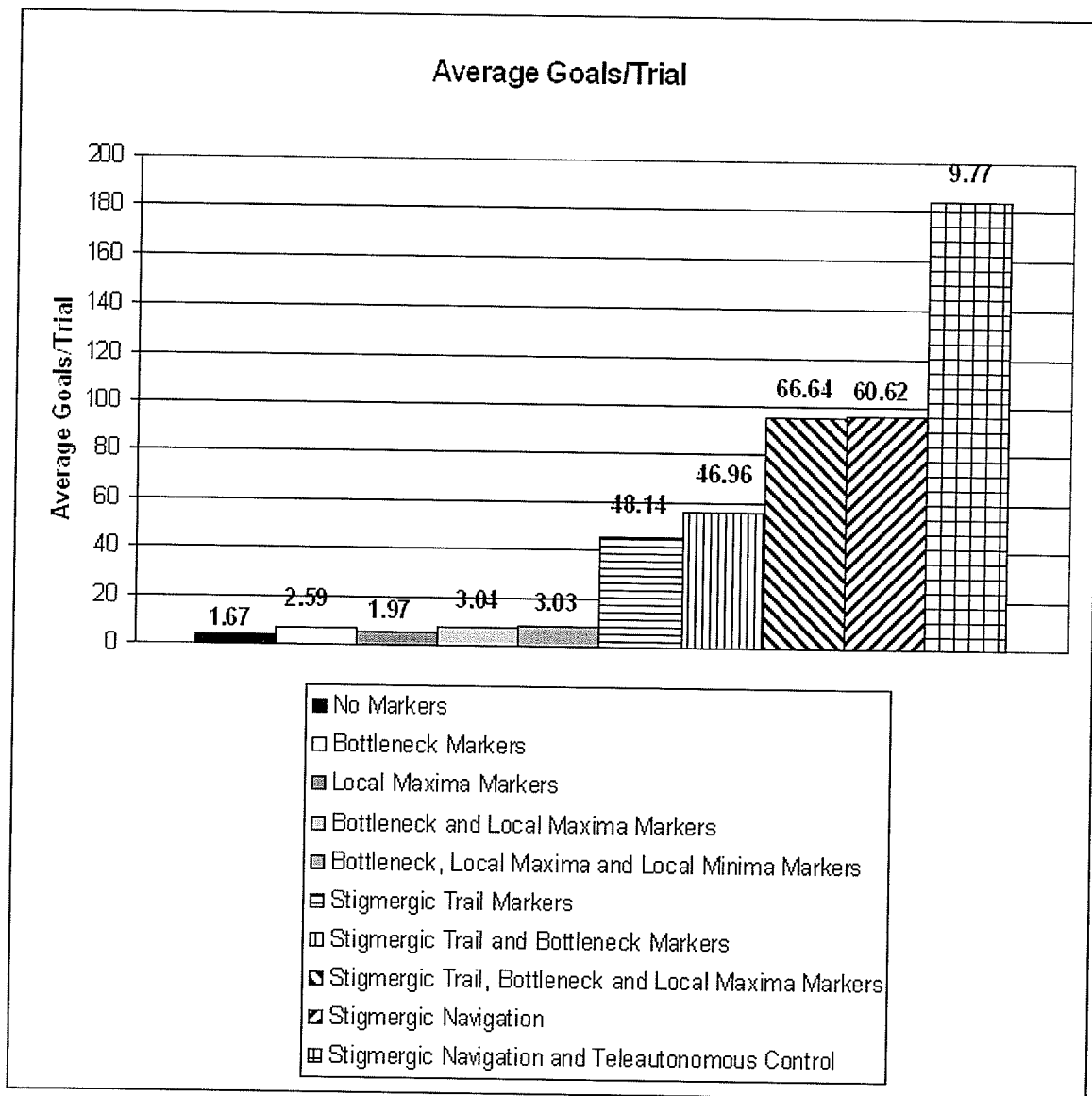Figure 5.12: Comparison of average goals/trial (the standard deviation for the number of goals/trial is indicated atop each bar)

results was that performance was not improved even further. Presumably, a human would help much more, which would seem to indicate that these techniques are quite powerful even when employed by reactive agents.

This demonstrates that stigmergy has strong potential as a means to influence and guide a team of agents so that they achieve their goals quicker and with greater ease.

5.4   Discussion

Table 5.1 summarizes some of the performance results presented in the previous sections, including the average time and number of think method invocations to initially locate the goal in a trial and the total goals across all 40 trials. As can be seen, the various marker types that I have used have resulted in substantial improvements in performance in most cases.

Bottleneck and local maxima markers increased the average number of goals found per trial significantly. The combination of bottleneck and local maxima markers even more so. Stigmergic trail markers resulted in a *very* substantial improvement over agents using no markers, or even the improved performance gained using the first two marker types. The best performances, however, occurred when bottleneck and local maxima markers were combined with stigmergic trail markers. This was because the bottleneck and local maxima markers encouraged agents to travel from area to area more frequently, causing agents to perceive and therefore extend the stigmergic trails more quickly. This allowed the team of agents to benefit from the trail for longer periods of time in each trial, leading to improved performance.

Table 5.1: Summary of Results

Marking Method	Average Time	Average Think	Total Goals
None	592.35	579822.88	161
BTl	463.67	43917.58	278
LocMax	533.48	52135.00	224
BTl/LocMax	567.02	51697.60	305
BTl/LocMax/LocMin	467.67	24068.98	323
StgTrl	462.19	45043.30	1828
StigTrl/BTl	508.15	46896.00	2253
StigTrl/BTl/LocMax	525.3	30869.03	3817
StigNav	490.11	24814.38	3856
StigTele	118.06	5125.75	183.75

None=No Markers, BTl=Bottleneck Markers, LocMax=Local Maxima Markers, LocMin=Local Minima Markers, StigTrl=Stigmergic Trails, StigNav=Stigmergic Navigation, StigTele=Stigmergic Navigation and Teleautonomous Control

The application of local minima markers, while nowhere near as significant, was notable in that rather than attracting or repelling agents, it specified an action to take at a certain location. This represents a step in the direction of encoding programs into the environment and merits further study [Werger and Mataric, 1999, 1996]. As well, its low impact on performance was deemed to be partially a factor of the agents' ability to deal with local minima and that agents tended to follow a path that bypassed them. In future work, it will be worthwhile to apply the same marker types in an environment that presents more difficult and frequent local minima, as the impact of local minima is largely domain-dependent.

Chapter 6

Conclusion

In this thesis I have presented techniques that use stigmergy to improve the reactive (or local) navigation performance of a team of agents navigating an unknown complex environment. In a series of experiments, I have demonstrated the efficacy of these methods. By marking bottlenecks and local maxima, agents were able to more frequently discover an unknown goal. At the same time, through stigmergic trail-making, agents were able to greatly increase the frequency and ease with which they could subsequently locate a previously discovered goal. All these methods are noteworthy in that the team of agents cooperatively marks the environment and shares knowledge of their explorations with each other to the betterment of the entire group, resulting in significantly better global and individual performance.

6.1 Findings and Analysis

The results of the experiments in Chapter 5 have answered the research questions outlined in Section 1.4. Agents improved their performance in locating an unknown goal by marking bottlenecks and local maxima to promote exploration. Agents were also able to cooperatively construct a minimal trail of markers leading to a discovered goal. They were likewise able to avoid a certain type of local minima using markers that instruct agents to jump when approaching them. All methods outlined in Chapter 5 were successful to varying degrees in getting agents to their goal faster and with increasing ease.

The heuristic of marking bottlenecks was shown to be useful in solving the box-canyon problem (as evidenced by my experimental results), yielding significant gains in terms of both time to initial goal discovery and the total number of times the team was able to rediscover their goal. Since it is difficult (if not impossible) for an agent to detect the existence of a doorway at a distance, the markers allow the agents to place a more easily detectable object at the location of interest. This simplifies the sensory load on other agents. In a real world scenario, it is likely that a robot would have to actually be at the doorway already in order to detect it (without a map and accurate localization) and so it would have to discover it itself. Using markers, the agent can "see" the doorway at a distance and utilize that information more readily.

Agents must balance exploration with marker following, otherwise they will simply follow the trail blazed by a teammate doing nothing new. Thus, choosing when to follow markers is important. I employed the simple mechanism of having agents ignore markers for a period of time after reaching one. This allowed them to wander around

the area that they presumably entered upon reaching the doorway for a while and perhaps find another different doorway.

As future work, it seems feasible to use this technique to systematically explore an environment. By giving markers unique identifiers, agents could potentially pass marker trails to each other when encountering each other. In essence, after an agent finds a goal and encounters another agent, it could give another agent instructions on how to reach the goal by following a particular sequence of markers

Experimental results also indicated that agents can benefit from marking local maxima and using them to more quickly navigate through perceptually uninteresting areas. While the results were not as good as those gained marking bottlenecks, this still yielded moderate improvements over random search by promoting wider coverage of the environment in shorter periods of time.

Of course, randomly placed stimuli are certainly no better than no stimuli at all. In fact, randomly placed stimuli can worsen performance by inhibiting exploration. For example, an agent might remain in a room where an attractor is present even though the room is otherwise unimportant. Consequently, markers must be deployed intelligently and minimally to avoid unnecessary clutter.

The use of local minima markers allowed agents to avoid certain local minima by jumping over them. The results yielded using this marking method, were not as dramatic and not as general as the other marking methods. Though somewhat more specialized than the other marking methods I have presented, this method is notable in that it encodes an instruction rather than attracting or repelling agents.

The most dramatic improvements were gained using stigmergic trail markers. This

method allowed agents to cooperatively construct a minimal trail of markers that allowed the entire team to share their explorations and proceed to their goal more quickly on subsequent trips. This was accomplished without agents using world modelling or path planning or needing to localize themselves in their environment.

My method of stigmergic trail-making encoded a simple numeric value indicative of the marker's place in the trail. This has the advantage of not requiring the agents to estimate distances. Instead, agents simply add one to the ID of an observed marker or set the marker value to 1 when observing the goal directly. Another method would be to number markers with a value indicating the agent's current distance from an observed goal. In the case of observing and identifying a marker, this would be a combination of the distance the agent is from the observed marker and the marker's value. Of course, this would require that agents have the means to estimate or measure their distance from an observed marker or goal.

As a result of combining the various marker types, agents were able to outstrip the performance of agents that did not use stigmergy by a broad margin. Bottleneck and local maxima markers allowed agents to locate the goal faster. This allowed agents to construct a stigmergic trail sooner. As a result, the agents were able to repeatedly locate the goal almost 23 times more frequently than without markers. Moreover, this rises to 39.39 times that of no markers, when factoring in the reduced number of frames executed per trial as a result of the extra computational overhead of the stigmergic methods.

6.2   Challenges

Deciding when to drop and when to follow markers is critical to the effectiveness of stigmergic navigation. If agents drop markers indiscriminately, they introduce noise and distraction that can in fact hinder rather than help agent performance. Similarly, *when* to follow markers is important too. Agents need to balance exploitation of markers with performance of other tasks or cyclic behaviour may result. Even more important is the ability to distinguish between markers so that agents can decide which markers to follow and which to ignore. This includes avoiding or ignoring markers already visited and preferring markers never visited over those that have been.

While my stigmergic techniques have yielded substantial gains, implementing these techniques in the real world, presents a number of challenges easily avoided in simulation. To apply these techniques in the real world agents will need to carry a physical supply of marker types or employ another method of marking the environment directly. The former method would mean agents would have a finite supply of markers and need to be especially discriminating in deploying them. The latter method requires that agents possess the necessary motor control to mark the environment in recognizable ways.

Marker fading has also not been dealt with in this thesis. I have attempted to show the advantages to be gained by being discriminate in choosing when to mark the environment in order to avoid marker clutter. However, several of the marker types fade in simulation to achieve the results here (these are noted where the marking technique is explained). This fading would have to be made physically possible in the

markers chosen in the real world. As noted in Section 2.5, research is ongoing into the use of chemical pheromones for these purposes [Kuwana et al., 1999]. Nonetheless, the techniques and principles that I have outlined should be equally applicable to physical robots as they are to simulated agents.

6.3   Future Work

In the future, it will be useful to explore how changing the environment or the number of agents affects performance. With respect to bottleneck markers, it would be interesting to observe how performance of these markers varies with the number of bottlenecks present in the environment or on the path to the goal. Similarly, it would be advantageous to study the performance of local minima markers in an environment that *forces* agents to negotiate them to get to the goal (rather than bypassing them, as frequently occurred in these experiments). It would be especially interesting to observe the effect on the average goals obtained per trial using stigmergic trails by varying the number of agents participating. Adding additional agents should increase the likelihood of the goal being discovered sooner. This in turn is likely to cause the stigmergic trail to be constructed sooner, allowing the agents to benefit from it for a longer period of time (similar to the effect gained using bottleneck and local maxima markers to augment stigmergic trail markers).

Another factor worth examining is varying the length of time agents spend ignoring bottleneck markers after traversing them to a new area. It seems probable that the time agents should spend ignoring these markers is a factor of the size of the area that they enter into after traversing a trail of bottleneck markers. If they enter a

small room, they will need little time to explore it, before exiting. The 30 second interval that I used in my experiments was arrived at through preliminary trials, but can undoubtedly be improved on and is situation dependent. Similarly, varying the length of time that agents retain visited markers in memory may prove interesting as well. Among other effects, varying this length affects the amount of items in the list and by extension an agent's memory requirements.

In regards to local maxima and local minima markers, it is also worth investigating the impact of varying the interval after which these markers disappear. Doing so will likely provide insights into what factors contribute to a marker's ongoing relevance. Indeed, this could also be combined with a reinforcement mechanism (similar to ant pheromone trails), to cause markers that are continuing to be useful to remain in place longer. As well, it would be useful to explore ways in which the local minima markers that I implemented can be extended or generalized to handle local minima situations more generally, rather than being limited to leaping over low-lying local minima.

In addition to investigating extensions to the marker types that I have presented, there is much potential for future work using far more advanced techniques. Getting agents to *learn* when to place and follow stigmergic markers is one such example. Rather than limiting agents to dropping and reacting to markers according to preset rules, it would be interesting to get agents to learn when and where to place markers themselves and to be more discriminating in exploiting them. One approach to accomplish this could include categorizing certain environmental features into exemplar states to allow agents to generalize and learn the characteristics of desirable locations

and update them dynamically. In so doing, agents can then use stigmergic markers to identify valuable locations more generally as they travel and indicate this via a marker along with a value rating (to allow observing agents to prefer one location over another).

Another application of stigmergic markers that warrants exploration is using markers to assist agents in localizing themselves cooperatively in conjunction with path planning navigation methods. This could be accomplished by having agents dynamically create landmarks in environments that lack easily identifiable signposts by dropping markers. In this approach, each marker dropped would encode the agent's beliefs about the coordinates of the location that the agent is marking along with a confidence factor. This would allow other navigating agents to use the marker as a potential landmark and as a means to relocalize themselves.

As well, it will be worthwhile experimenting with a variety of marker types that signify actions like the local minima markers that I have presented here. It seems likely that these additional marker types can be used to sequence the movements and actions of agents as they traverse their environment. For example, other agents (or a human being) can place different symbols that specify an order in which locations should be traversed to get somewhere in the world. By doing so, it seems feasible that the agents themselves can be kept relatively simple and still perform complex actions. This may be particularly useful when applied in conjunction with teleautonomy. The superior perceptual ability and intelligence of a human can be used to assist agents in performing complicated tasks or in recognizing objects that agents have trouble recognizing themselves.

Most notably indirect communication markers can be used to inform agents of positions where they can attempt to perform certain actions that require that they be in particular positions within an environment. For example, going through a doorway requires that the agent be in front of the doorway. Similarly, climbing a ladder requires that an agent first stand before the ladder, just as using an elevator requires that the agent be inside the elevator. At its logical extreme, it becomes possible to encode entire programs of actions into the world itself [Werger and Mataric, 1999, 1996]. The interesting challenge becomes deciding what types of markers to leave and where to leave them so that they are maximally useful and minimally obtrusive.

There are also future applications of the use of stigmergic markers, such as their application to assist in the teleautonomous control of agents. Finally, there are a large number of potential studies in multi-agent systems that could proceed from this work: I have fixed the number of agents, but were that to be made variable, a number of interesting things could be explored. Density of marking, and the effect of clutter as we increase the number of individuals, for example, or the interference between agents as they attempt to mark the environment from their own potentially unique perspectives.

6.4   Summary

Stigmergy shows great promise in solving many of the navigation challenges faced by reactively navigating agents, leveraging their innate sensory abilities. By modifying their surroundings, agents have been shown to be better able to navigate unknown and complex environments without the need to construct, store and maintain world

models. Just one of the additional possibilities to be leveraged from storing discovered knowledge of the world in the environment through stigmergy was shown, through the use of teleautonomous robotic control using my techniques. This application allowed a human to control a team of agents without having to specify specific agents nor engage in explicit communication.

The techniques that I have outlined in this thesis only begin to scratch the surface in terms of the potential applications of stigmergy in robotic agents. All of the techniques used in this thesis, save for the local minima technique, were reasonably general and fairly easily transferable to other domains, and I have outlined a number of future directions for this work. While stigmergic techniques, in nature and otherwise, are not new, I believe that both they and the central concept they embody - storing knowledge in a distributed fashion throughout the world as opposed to inside an agent - are both currently very under-appreciated in artificial intelligence. In the future, however, I believe that the concept of storing knowledge in the world, and techniques such as stigmergic navigation which embody this concept, will be seen much more extensively and viewed as powerful tools.

Appendix A

Schemas and Assemblages

This appendix describes some commonly used motor schemas, perceptual schemas and assemblages that are formed by combining them in order to handle particular states. The gain values and motor schemas used by each assemblage vary somewhat depending on the experimental scenario, but some common examples are provided. In addition, the gain values associated with motor schemas are not necessarily ideal, but were found to work reasonably well. Little time was spent trying to fine tuning gain values, as my focus was on improving performance via stigmergy.

A.1 Motor Schemas

motor-avoid-marker-local-maxima causes the agent to move away from the nearest perceptible local maxima marker.

motor-avoid-marker-local-minima causes the agent to move toward the nearest perceptible local minima marker.

126

motor-avoid-mobile-obstacle causes the agent to turn away from a mobile obstacle such as another agent.

motor-avoid-static-obstacle causes the agent to turn 180 degrees in the opposite direction of a perceived static obstacle such as a wall or other barrier.

motor-drop-marker-bottleneck causes the agent to drop a bottleneck marker on the ground at its current location.

motor-drop-marker-goal causes the agent to drop a stigmergic trail marker on the ground at its current location.

motor-drop-marker-local-maxima causes the agent to drop a local maxima marker on the ground at its current location.

motor-drop-marker-local-minima causes the agent to drop a local minima marker on the ground at its current location.

motor-move-to-hev-charger causes the agent to move towards a visible HEV armour charging unit (if the agent's armour level is not at maximum).

motor-move-to-ladder causes the agent to move towards a perceptible ladder.

motor-move-to-marker-bottleneck causes the agent to move towards the nearest perceptible bottleneck marker.

motor-move-to-marker-goal causes the agent to move toward the lowest valued perceptible stigmergic trail marker.

motor-move-parallel-wall causes the agent to move in a direction parallel to a nearby wall.

motor-noise causes the agent to move in a random direction.

motor-on-ladder causes the agent to move up or down on a ladder.

motor-ready-marker causes the agent to select and ready a marker for deployment.

motor-stuck causes the agent to jump, duck and move randomly in order to extract itself when it gets stuck (unchanged position for extended period).

motor-swim causes the agent to move upward to the surface of the water when the agent is in water.

A.2 Perceptual Schemas

percept-bottleneck returns a true value when the agent is standing in a doorway or hallway.

percept-dock-hev returns a true value when conditions are right for the agent to switch to a set of motor schemas appropriate to docking with a HEV (Hazardous Environment) suit wall charger. This is a suit that is used to protect the agent from hazardous chemicals in some areas of half-life environments, and requires an energy recharge after any chemical exposure. Returns true when the agent has less than full health and is within (configurable) range of a visible HEV wall charger.

percept-drop-marker-bottleneck returns a true value when conditions are such that the agent should drop a bottleneck marker at its current location.

percept-drop-marker-goal returns a true value when conditions are such that the agent should drop a stigmergic trail marker at its current location.

percept-drop-marker-local-maxima returns a true value when conditions are such that the agent should drop a marker at its current location.

percept-drop-marker-local-minima returns a true value when conditions are such that the agent should drop a local minima marker at its current location.

percept-hev-charger returns a true value when a HEV suit wall charger is within sensory range.

percept-in-water returns true when the agent is immersed in water (water coverage is also a part of many Half-Life environments, and the agent can remain in water only for a limited length of time without damage). Can be queried for information about how deep in the water the agent is (i.e. knee high, submerged)

percept-ladder returns true when a ladder is visible to the agent. Can be queried for distance and relative vectors to ladder

percept-marker-bottleneck returns a true value when one or more bottleneck markers lies in the agent's perceptual range and provides a list of distances and vectors to them.

percept-marker-goal returns a true value when one or more stigmergic trail markers lie in the agent's perceptual range and provides a list of distances and vectors

to them.

percept-marker-local-maxima returns true when one or more stigmergic marker(s) lies in the agent's perceptual range and a list of distances and vectors to them.

percept-marker-local-minima returns a true value when one or more local minima markers lies in the agent's field of view and provides a list of distances and vectors to them.

percept-mobile-obstacle returns true when a mobile obstacle, such as another agent, lies in the agent's path.

percept-on-ladder returns true when the agent is currently climbing a ladder.

percept-static-obstacle returns true when an obstacle lies in the agent's path. Likewise, provides additional information about the outline of the obstruction. This allows the agent to determine whether ducking or jumping or turning away is the best action to take to avoid it.

percept-stuck returns true when the agent has not moved from it's last recorded position in a number of time units.

A.3 Assemblages

wander is the default assemblage that is used when no other assemblage's are appropriate and is deactivated when any other assemblage's activation conditions are met. The behaviours making up this assemblage, and their respective gains, are:

1. motor-avoid-static-obstacle with a gain value of 8.8

2. motor-avoid-mobile-obstacle with a gain value of 6.23

3. motor-avoid-marker-local-maxima with a gain value of 40

4. motor-avoid-marker-local-minima with a gain value of 40

5. motor-move-to-hev-charger with a gain value of 28

6. motor-move-to-ladder with a gain value of 10.5

7. motor-move-to-marker-bottleneck with a gain value of 80

8. motor-move-parallel-wall with a gain value of 28

9. motor-noise with a gain value of 6.35

10. motor-stuck with a gain value of 40

11. motor-ready-marker with a gain value of 1

dock-with-hev-charger is used to allow the agent to move right up to a wall charger for the HEV suit (explained above). To this end, motor schemas such as avoid static obstacle are not included as they would prevent the agent from closing the distance to the wall unit.

This assemblage is activated when the agent's armour level is less than full power and it moves within 40 units of a HEV charger. The behaviours making up this assemblage, and their respective gains, are:

1. motor-move-to-hev-charger with a gain value of 10

2. motor-use-hev-charger with a gain value of 10

drop-marker-local-maxima is activated to allow the agent to drop a local maxima marker on the ground at it's current location. It is activated when percept-drop-marker-local-maxima returns a true value. It is deactivated when percept-drop-marker-local-maxima returns a false value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-drop-marker-local-maxima with a gain value of 1

drop-marker-local-minima is activated to allow the agent to drop a local minima marker on the ground at it's current location. It is activated when percept-drop-marker-local-minima returns a true value. It is deactivated when percept-drop-marker-local-minima returns a false value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-drop-marker-local-minima with a gain value of 1

drop-marker-bottleneck is activated to allow the agent to drop a local maxima marker on the ground at it's current location. It is activated when percept-drop-marker-bottleneck returns a true value. It is deactivated when percept-drop-marker-bottleneck returns a false value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-drop-marker-bottleneck with a gain value of 1

drop-marker-goal is activated to allow the agent to drop a stigmergic trail marker on the ground at it's current location. It is activated when percept-drop-marker-goal returns a true value. It is deactivated when percept-drop-marker-goal

returns a false value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-drop-marker-goal with a gain value of 1

move-to-hev-charger causes the agent to move toward the nearest visible wall armour charging unit. This is a device in the Half-Life game that allows a player to increase the protection offered by a protective Hazardous Environment (HEV) suit. It is activated when percept-hev-charger returns a true value and deactivated when it returns a false value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-avoid-mobile-obstacle with a gain value of 5.1

2. motor-avoid-static-obstacle with a gain value of 5.8

3. motor-move-to-hev-charger with a gain value of 40

4. motor-move-parallel-wall with a gain value of 28

5. motor-noise with a gain value of 2.5

6. motor-ready-marker with a gain value of 1

7. motor-stuck with a gain value of 40

move-to-marker-bottleneck once the agent has entered a bottleneck such as a doorway or hallway this assemblage is activated. It causes the agent to move toward the nearest visible bottleneck marker visible that it has not visited in the past 30 seconds. The noise motor schema in this assemblage is lowered significantly, so that it does not unduly interfere with the agent's marker following. It is

activated when percept-bottleneck returns a true value. It is deactivated when percept-bottleneck returns a false value or percept-marker-goal returns a true value. The behaviours making up this assemblage, and their respective gains, are:

1. motor-avoid-static-obstacle with a gain value of 0.2

2. motor-move-to-marker-bottleneck with a gain value of 30

3. motor-move-parallel-wall with a gain value of 28

4. motor-noise with a gain value of 0.1

5. motor-ready-marker with a gain value of 1

move-to-marker-goal causes the agent to move toward the lowest numbered stigmergic trail marker visible. It is activated when percept-move-to-marker-goal returns a true value. It is deactivated when percept-move-to-marker-goal returns a false value, percept-hev-charger is false, percept-drop-marker-goal is true or percept-stuck is true. The behaviours making up this assemblage, and their respective gains, are:

1. motor-avoid-static-obstacle with a gain value of 0.2

2. motor-move-parallel-wall with a gain value of 28

3. motor-move-to-marker-goal with a gain value of 40

4. motor-noise with a gain value of 5.0

5. motor-ready-marker with a gain value of 1

on-ladder activates a set of motor schemas suited to ladder climbing. It is activated when percept-on-ladder returns true. It is deactivated when percept-on-ladder returns false or percept-enemy returns true. The behaviours making up this assemblage, and their respective gains, are:

1. motor-on-ladder with a gain value of 11

2. motor-move-to-ladder with a gain value of 11 (to cause the agent to face the ladder).

Appendix B

Result listings

This appendix provides complete result listings for the various experiments detailed in Chapter 5. The results shown are for a team of 6 agents across 40 games. Each table lists the game number, average initial discovery time, average initial discovery think count and total goals for the team.

B.1 No Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	991.563	96243	4
2	586.75	57444	6
3	333.241	32066	4
4	433.809	42499	7
5	1124.45	110100	4
6	280.008	27441	8
7	337.906	33115	4
8	280.25	27463	2
9	769.896	75430	6
10	1462.81	143375	3
11	184.595	18109	5
12	1506.4	146478	1
13	431.532	42323	4
14	359.744	35276	5
15	388.314	38095	7
16	522.459	51244	2
17	595.972	58430	4
18	453.628	44485	2
19	349.166	34221	4
20	479.968	47049	3
21	340.629	33396	6
22	161.092	15789	2
23	973.19	95380	1
24	398.366	39025	6
25	598.527	58609	5
26	701.211	68875	3
27	502.553	49237	5
28	257.023	25189	4
29	89.9881	8820	5
30	185.905	18203	4
31	1262.74	123758	4
32	449.84	44084	4

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	540.621	53008	5
34	549.287	53832	4
35	809.507	79335	3
36	715.278	70053	5
37	789.132	77314	2
38	417.749	40925	3
39	634.86	62192	4
40	1443.99	141405	1
Average	592.35	57982.88	4.03

B.2 Bottleneck Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	252.141	23512	4
2	485.531	45830	7
3	428.773	40165	8
4	357.973	33941	12
5	776.731	73541	8
6	281.066	26760	11
7	430.796	40944	10
8	352.702	33439	8
9	218.235	20512	9
10	661.046	62994	5
11	238.459	22627	11
12	264.394	25106	5
13	1504.37	142995	1
14	226.658	21382	10
15	462.33	43520	7
16	337.729	32008	9
17	211.96	20074	9
18	577.529	54712	4
19	434.583	41310	9
20	926.219	87985	6
21	507.603	48120	3
22	752	71440	4
23	239.444	22610	8
24	322.896	30408	5
25	630.169	59802	5
26	729.311	69236	4
27	454.236	43017	6
28	548.53	51968	5
29	428.901	40583	11
30	535.02	50655	7
31	404.147	38408	5
32	354.812	33640	6

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	259.974	24480	4
34	531.228	50172	9
35	343.135	32616	6
36	641.15	60585	5
37	386.691	36575	8
38	408.606	38935	11
39	223.552	21065	7
40	416.172	39031	6
Average	463.67	43917.58	6.95

B.3 Local Maxima Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	578.52	56630	6
2	643.574	63023	7
3	356.849	34855	8
4	173.77	17029	5
5	672.988	65924	5
6	488.203	47836	3
7	385.248	37788	4
8	599.497	58800	7
9	1347.43	132098	2
10	666.292	65319	6
11	465.756	45610	5
12	219.1	21478	5
13	725.213	71101	5
14	454.213	44482	6
15	712.997	69909	3
16	722.707	70880	4
17	312.553	30640	9
18	909.886	89204	4
19	517.305	50685	7
20	645.893	63236	7
21	368.118	36057	5
22	320.929	31465	11
23	424.08	41524	5
24	472.605	46301	7
25	379.555	37159	4
26	470.17	43947	6
27	1056.63	102687	3
28	822.14	79935	7
29	582.036	56687	8
30	251.944	24497	8
31	367.367	35703	3
32	337.261	32883	7

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	454.978	44248	9
34	319.508	31116	5
35	671.454	65726	5
36	661.962	64760	4
37	182.661	17876	3
38	279.578	27337	4
39	804.777	78760	7
40	513.618	50205	5
Average	533.48	52135	5.6

B.4 Bottleneck and Local Maxima Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	720.835	63630	8
2	665.43	61179	8
3	491.406	43934	9
4	703.671	63003	4
5	887.163	80624	4
6	545.852	49801	5
7	777.533	72185	11
8	697.132	61399	10
9	747.33	66626	5
10	865.641	78422	4
11	688.074	62474	6
12	395.732	35386	9
13	437.298	40682	8
14	1113.75	102671	3
15	688.129	63634	3
16	200.713	18510	13
17	867.346	80240	8
18	295.113	26937	13
19	372.25	34275	11
20	329.988	29777	6
21	590.833	55143	8
22	244.124	22167	10
23	683.236	62584	5
24	621.11	57038	6
25	435.615	39519	12
26	762.396	70378	7
27	425.223	38018	11
28	542.289	49769	6
29	258.061	23190	9
30	552.199	51293	7
31	294.935	26942	7
32	842.54	77575	3

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	337.559	30678	11
34	248.504	22137	11
35	629.249	57485	4
36	807.117	73480	4
37	507.022	45994	7
38	290.913	26425	11
39	763.275	70346	5
40	354.256	32354	13
Average	567.02	51697.60	7.63

B.5 Bottleneck, Local Maxima and Local Minima

Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	187.931	9505	10
2	467.342	24704	10
3	335.334	17300	9
4	352.056	18441	12
5	597.99	30374	9
6	225.294	11539	8
7	624.138	32310	8
8	927.765	48333	6
9	990.101	50033	6
10	371.374	18877	9
11	249.698	13224	11
12	746.368	37921	8
13	231.09	11825	12
14	1183.84	60224	3
15	875.051	45432	5
16	404.241	20731	9
17	344.696	17476	8
18	434.355	21415	10
19	390.02	20040	6
20	341.621	18025	5
21	629.317	32037	4
22	294.774	15477	14
23	824.205	43116	4
24	705.645	36975	8
25	675.952	34878	3
26	255.259	13322	6
27	388.949	20093	9
28	382.781	19863	6
29	309.956	15915	13
30	177.005	9052	10
31	233.778	11635	5
32	628.235	31898	8

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	440.063	22246	5
34	215.53	10998	10
35	355.021	19033	14
36	414.716	21805	11
37	386.776	19534	13
38	204.805	10628	4
39	455.003	24173	6
40	448.739	22352	6
Average	467.67	24068.98	8.08

B.6 Stigmergic Trail Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	335.324	32385	106
2	654.179	64097	14
3	999.759	98396	5
4	422.866	41608	15
5	470.282	46435	7
6	491.609	48416	30
7	710.185	70048	13
8	380.794	37545	17
9	651.008	64172	20
10	312.768	30872	17
11	168.434	16600	157
12	529.195	52336	31
13	367.235	35883	8
14	485.66	47468	58
15	312.106	30528	15
16	191.854	18033	182
17	928.17	91536	9
18	529.321	49551	117
19	618.697	52711	59
20	734.083	70840	10
21	643.355	63445	70
22	260.158	25629	101
23	382.96	37768	12
24	776.634	76510	23
25	434.205	42846	92
26	250.281	24679	150
27	829.705	81734	14
28	630.568	62138	6
29	562.766	55473	134
30	263.584	25989	38
31	793.263	78235	4
32	280.249	27638	29

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	509.771	50200	37
34	103.642	10213	7
35	163.183	16047	10
36	152.12	14998	30
37	280.304	27657	55
38	489.886	48342	78
39	183.73	17206	40
40	203.603	15525	8
Average	462.19	45043.30	45.70

B.7 Bottleneck and Stigmergic Trail Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	320.083	29402	142
2	193.157	17783	95
3	818.246	75924	28
4	567.292	52519	85
5	198.762	18535	88
6	574.408	54200	14
7	971.863	91717	7
8	257.479	24111	89
9	294.024	27314	128
10	240.653	21994	22
11	853.443	79260	13
12	1224.31	114380	2
13	728.335	67840	90
14	757.589	70246	21
15	857.744	79014	2
16	293.563	27171	74
17	490.089	44938	50
18	738.954	63628	8
19	547.559	46988	9
20	350.21	32035	7
21	246.345	22511	81
22	249.077	22757	116
23	901.377	83349	55
24	325.358	30286	56
25	215.88	20084	115
26	367.874	34154	21
27	587.285	54583	9
28	580.16	53929	76
29	448.181	41529	60
30	388.204	36232	81
31	163.611	14996	154
32	374.499	35114	113

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	399.361	36706	151
34	784.355	74128	5
35	583.6	54427	83
36	326.341	29436	8
37	876.717	81297	14
38	359.007	33613	59
39	352.299	31196	11
40	518.877	46514	11
Average	508.15	46896.00	56.33

B.8 Bottleneck, Local Maxima and Stigmergic Trail

Markers

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	529.241	30005	78
2	637.177	38908	109
3	322.534	18690	16
4	873.366	50176	10
5	670.233	38469	87
6	1275.38	75181	5
7	371.875	21191	57
8	255.447	14784	227
9	247.075	15021	247
10	439.562	25622	90
11	671.233	35507	124
12	596.151	35312	119
13	718.543	41069	62
14	503.554	29711	153
15	249.449	14000	201
16	155.975	9080	240
17	546.458	29160	63
18	795.105	45663	11
19	475.623	28024	43
20	486.242	28413	106
21	1308.86	75247	3
22	129.957	6949	124
23	428.747	25149	102
24	439.874	27089	86
25	377.243	24807	172
26	394.333	24774	133
27	706.276	42726	74
28	299	17166	67
29	362.31	21155	64
30	423.539	25670	44
31	217.554	13049	74
32	593.133	36847	137

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	592.072	35092	10
34	1196.36	72049	3
35	588.54	33869	43
36	331.597	21697	204
37	283.037	16694	127
38	407.025	24367	126
39	1002.31	60090	44
40	109.899	6289	132
Average	525.30	30869.03	95.43

B.9 Stigmergic Navigation

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	327.56	16685	10
2	778.84	39508	169
3	559.765	28189	113
4	579.417	28553	23
5	418.394	21199	7
6	379.185	19390	190
7	558.487	27975	72
8	234.027	11633	134
9	312.51	15332	90
10	936.121	43503	71
11	331.667	16931	223
12	936.657	47345	15
13	800.732	40948	117
14	882.88	45056	3
15	478.128	24051	51
16	550.241	27979	107
17	548.239	28638	77
18	491.877	25036	171
19	561.107	29146	55
20	325.248	16511	118
21	1073.44	54356	64
22	153.931	7778	32
23	365.526	18471	115
24	305.89	15496	193
25	270.89	13676	37
26	159.5	8100	132
27	235.197	12011	204
28	693.66	34918	33
29	778.87	39960	20
30	367.819	19034	112
31	1062.11	55104	54
32	362.28	17868	127

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
33	785.981	39842	60
34	294.994	14868	133
35	281.976	14601	161
36	604.419	30610	35
37	173.319	9607	135
38	189.132	9654	99
39	286.19	14319	179
40	168.112	8694	115
Average	490.11	24814.38	96.4

B.10 Stigmergic Navigation and Teleautonomous Control

Game No.	Average Discovery Time	Average Discovery Think Count	Total Goals
1	100.077	4317	93
2	170.85	6614	199
3	99.5019	4728	240
4	101.792	4844	203
Average	118.06	5125.75	183.75

Bibliography

Philip E. Agre. *The Dynamic Structure of Everyday Life*. PhD thesis, MIT, Cambridge, Mass, 1988.

John Anderson and Alfred Wurr. Dimensions of teleautonomy in mobile agents. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, pages 1–6, July 2002. Banff, AB.

Ronald C. Arkin. *Behavior-based Robotics*. MIT Press, Cambridge, 1998.

Ronald C. Arkin and K. Ali. Integration of reactive and telerobotic control in multiagent robotic soccer systems. In *Proceedings of the 3rd International Conference on the Simulation of Adaptive Behaviour*, 1994. Brighton.

Ronald C. Arkin and Tucker Balch. Aura: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):175–189, 1997.

Ronald C. Arkin and Tucker Balch. Cooperative multiagent robotic systems. In R.P. Bonasso and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, 1998. Cambridge, MA.

Tucker Balch and Ronald C. Arkin. Avoiding the past: A simple but effective strategy

for reactive navigation. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, volume 1, pages 678–685, May 1993. Atlanta, Georgia.

Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994.

Tucker Balch and Ronald C. Arkin. Motor schema-based formation control for multi-agent robot teams. In *Proceedings of 1995 International Conference on Multiagent Systems*, pages 10–16, 1995.

Tucker Balch and Ronald C. Arkin. Clay: Integrating motor schemas and reinforcement learning. GIT 11, College of Computing, University of Southern California, Los Angeles, March 1997.

Tucker Balch and Ronald C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Trans. On Robotics and Automation*, 14(6), December 1998.

Jacky Baltes and John Anderson. Flexible binary space partitioning for robotic rescue. Submitted to IEEE IROS - Intelligent Robots and Systems, 2003.

R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Marc. G. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):237–256, 1997.

Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA2(1):14–23, April 1986.

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

Rodney A. Brooks. Intelligence without reason. In *Proc. of the 1991 Int. Joint Conf. on Artificial Intelligence*, pages 569–595, 1991a.

Rodney A. Brooks. The role of learning in autonomous robots. In *Fourth Annual Workshop on Computational Learning Theory*, pages 5–10, 1991b.

S. Buck, T. Schmitt, and M. Beetz. Reliable multi robot coordination using minimal communication and neural prediction. In M. Beetz, J. Hertzberg, M. Ghallab, and M. Pollack, editors, *Advances in Plan-based Control of Autonomous Robots. Selected Contributions of the Dagstuhl Seminar Plan-based Control of Robotic Agents*. Springer Verlag, 2002.

W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

Markus Bylund and Fredrik Espinoza. Testing and demonstrating context-aware services with quake iii arena. *Communications of the ACM*, 45(1):46–48, January 2002.

M. DesJardins, E. Durfee, C. Ortiz, and M. Wolverton. Survey of research in distributed continual planning. *AI Magazine*, 1999.

E. Durfee. Distributed continual planning for unmanned ground vehicle teams. *AI Magazine*, pages 55–61, 1999.

Oren Etzioni. Intelligence without robots: a reply to Brooks. *AI Magazine*, 14(4): 7–13, December 1993.

J. Fredslund and M. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

Daniel D. Fu, Kristian J. Hammond, and Michael J. Swain. Navigation for everyday life. Technical Report TR-96-03, University of Chicago, Computer Science Department, 1100 East 58th Street, Chicago, Illinois 60637, February 1996.

E. Gat. On three-layer architectures. In R. Murphy D. Kortenkamp, R.P. Bonnasso, editor, *Artificial Intelligence and Mobile Robots*. AAAI Press, 1997.

Timothy Groner and John Anderson. Efficient multi-robot localization and navigation through passive cooperation. In *Proceedings of the 2001 International Conference on Artificial Intelligence*, pages 84–89, June 2001. Las Vegas, Nevada.

K.J. Hammond, T.M. Converse, and J.W. Grass. The stabilization of environments. *Artificial Intelligence*, 72(1-2):305–327, 1995.

O. Holland and C. Melhuish. Stigmergy, self-organisation, and sorting in collective robotics. *Artificial Life*, 5(2), 2000.

B. Holldobler and E.O. Wilson. *The Ants*. Harvard University Press, 1990.

Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 434–459, 2002.

Jeffrey Jacobson and Zimmy Hwang. Unreal tournament for immersive interactive theater. *Communications of the ACM*, 45(1):39–42, January 2002.

Gal A. Kaminka, Manuela M. Veloso, Steve Schaffer, Chris Sollito, Rogelio Adobatti, Andrew N. Marshall, Andrew Scholer, and Sheila Tejada. Game bots: A flexible test bed for multiagent team research. *Communications of the ACM*, 45(1):43–45, January 2002.

H. Kitano. Robocup rescue: A grand challenge for multiagent systems. In *Proceedings of the 4th International Conference on Multi-Agent Systems*, pages 5–12, 2000.

C. Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Preprint submitted to Robotics and Autonomous Systems*, October 30 1998.

C.R. Kube and H. Zhang. Stagnation recovery behaviors for collective robots. In *Proceedings of 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 1883–1890. IEEE Computer Society Press, 1995. Los Alamitos, CA.

Nicholas Kushmerick. Ask not what's inside your head but what your head's inside of, 1994. URL `citeseer.nj.nec.com/113548.html`.

Nicholas Kushmerick. Cognitivism and situated action: Two views on intelligent agency. *Computer and Artificial Intelligence*, 15(5), 1996.

Y. Kuwana, S. Nagasawa, I. Shimoyama, and R. Kanzaki. Synthesis of silkworm moth's pheromone-oriented behavior by a mobile robot with moth's antennae as pheromone sensors. *Biosensors and Bioelectronics*, 14:195–202, 1999.

John Laird. An exploration into computer games and computer generated forces.

In *9th Conference on Computer Generated Forces and Behavioral Representation*, 2000.

John Laird. Using a computer game to develop advanced ai. *Computer*, 34(7):70–75, July 2001.

John Laird. Research in human-level ai using computer games. *Communications of the ACM*, 45(1):32–35, January 2002.

John Laird and M. Van Lent. Human-level ai's killer application: Interactive computer games. *AI Magazine*, pages 15–25, Summer 2001.

Michael Lewis and Jeffrey Jacobson. Games engines in scientific research. *Communications of the ACM*, 45(1):27–31, January 2002.

M. J. Mataric. Behavior-based control: Examples from navigation, learning and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3): 323–336, 1997.

K. Moorman and A. Ram. A case-based approach to reactive control for autonomous robots. In *AAAI Fall Symposium on AI for Real-World Autonomous Mobile Robots*, 1992. Cambridge, MA.

U. Nehmzow, D. Gelder, and T. Duckett. Automatic selection of landmarks for mobile robot navigation. Technical Report UMCS-00-7-1, Department of Computer Science, University of Manchester, 2000.

H. S. Nwana, L. C. Lee, and N. R. Jennings. Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88, 1996.

H. V. D. Parunak and S. Brueckner. Ant-like missionaries and cannibals: Synthetic pheromones for distributed motion control. In *Fourth International Conference on Autonomous Agents*, pages 467–474, June 2000. Barcelona, Spain.

H. V. D. Parunak, S. Brueckner, J.A. Sauter, and J. Posdamer. Mechanisms and military applications for synthetic pheromones. In *Workshop on Autonomy Oriented Computation*, May 29 2001. Montreal, Canada.

Andres Perez-Uribe and Beat Hirsbrunner. Learning and foraging in robot-bees. In *SAB2000 Proceedings Supplement Book*, pages pp. 185–194, Honolulu, 2000. International Society for Adaptive Behavior.

Wayne Piekarski and Bruce Thomas. Arquake: The outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, January 2002.

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

D. Reece and M. Kraus. Tactical movement planning for individual combatants. In *Proceedings of 9th Conference on Computer Generated Forces and Behavioral Representation*, 2000.

M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, MA, 1998.

S. Rumeliotis, P. Pirjanian, and M.J. Mataric. Ant-inspired navigation in unknown

environments. In *Proceedings of the 4th International Conference on Autonomous Agents*, pages 25–26, June 7 2000. Barcelona, Spain.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

John A. Sauter, Robert Matthews, H. Van Dyke Parunak, and Sven Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 434–440. ACM Press, July 2002. ISBN 1-58113-480-0. Bologna, Italy.

Antonio Sgorbissa and Ronald C. Arkin. Local navigation strategies for a team of robots. Technical report, Georgia Tech Robotics Laboratory, 2001.

P. Stone and M. Veloso. Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

M. Tambe, J. Adibi, Y. Al-Onaizan, G.A. Kaminka, and I. Muslea. Building agent teams using an explicit teamwork model and learning. *Artificial Intelligence*, 110 (2):215–239, June 1999.

John K. Tsotsos. Behaviorist intelligence and the scaling problem. *Artificial Intelligence*, pages 135–160, 1995.

R. Vaughan, K. Stoy, G. Sukhatme, and M. Mataric. Blazing a trail: insect-inspired resource transportation by a robot team. In *Proceedings of the International Symposium on Distributed Autonomous Robot Systems*, 2000a. Knoxville, TN, USA.

R. Vaughan, K. Stoy, G. Sukhatme, and M. Mataric. Whistling in the dark: Cooperative trail following in uncertain localization space. In *Proc. of the Fourth Int. Conf. on Autonomous Agents*, 2000b.

R. Vaughan, K. Stoy, G. Sukhatme, and M. Mataric. Lost: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation*, 18(5):796–812, 2002.

M. Veloso and P. Stone. Individual and collaborative behaviors in a team of homogeneous robotic soccer agents. In *Proceedings of the 3rd International Conference on Multi-agent Systems*, pages 309–316, 1998.

Jim Van Verth, Victor Brueggeman, Jon Owen, and Peter McMurry. Formation-based pathfinding with real-world vehicles. In *Game Developers Conference Proceedings*, 2000.

G. Weiss. *Multiagent Systems*. MIT Press, 1999.

Barry Brian Werger. Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110:293–320, 1999.

Barry Brian Werger and M. Mataric. Exploiting embodiment in multi-robot teams. Technical Report IRIS-99-378, University of Southern California, Institute for Robotics and Intelligent Systems, 1999.

Barry Brian Werger and Maja Mataric. Robotic food chains: Externalization of state and program for minimal-agent foraging. In *Proceedings of the 4th International*

Conference on Simulation of Adaptive Behavior: From Animals to Animats 4, pages 625–634. MIT Press, 1996.