Nadia Nedjah
Luiza de Macedo Mourelle

Swarm Intelligent Systems

With 65 Figures and 34 Tables

Springer

Dr. Nadia Nedjah
Department of Electronics Engineering and Telecommunications - DETEL
Faculty of Engineering - FEN
State University of Rio de Janeiro - UERJ
Rua São Francisco Xavier, 524, 5o. andar
Maracanã, CEP 20559-900
Rio de Janeiro, RJ
Brazil
E-mail: nadia@eng.uerj.br

Dr. Luiza de Macedo Mourelle
Department of System Engineering and Computation - DESC
Faculty of Engineering - FEN
State University of Rio de Janeiro - UERJ
Rua São Francisco Xavier, 524, 5o. andar
Maracanã, CEP 20559-900
Rio de Janeiro, RJ
Brazil
E-mail: ldmm@eng.uerj.br

Library of Congress Control Number: 2006925434

ISSN print edition: 1860-949X
ISSN electronic edition: 1860-9503
ISBN-10 3-540-33868-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33868-0 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Printed in The Netherlands

Cover design: deblik, Berlin
Typesetting by the authors and SPI Publisher Services
Printed on acid-free paper SPIN: 11612803 89/SPI 5 4 3 2 1 0

Preface

Swarm intelligence is an innovative computational way to solving hard problems. This discipline is inspired by the behavior of social insects such as fish schools and bird flocks and colonies of ants, termites, bees and wasps. In general, this is done by mimicking the behavior of the biological creatures within their swarms and colonies.

Particle swarm optimization, also commonly known as PSO, mimics the behavior of a swarm of insects or a school of fish. If one of the particle discovers a good path to food the rest of the swarm will be able to follow instantly even if they are far away in the swarm. Swarm behavior is modeled by particles in multidimensional space that have two characteristics: a position and a velocity. These particles wander around the hyperspace and remember the best position that they have discovered. They communicate good positions to each other and adjust their own position and velocity based on these good positions.

The ant colony optimization, commonly known as ACO, is a probabilistic technique for solving computational hard problems which can be reduced to finding optimal paths. ACO is inspired by the behavior of ants in finding short paths from the colony nest to the food place. Ants have small brains and bad vision yet they use great search strategy. Initially, real ants wander randomly to find food. They return to their colony while laying down pheromone trails. If other ants find such a path, they are likely to follow the trail with some pheromone and deposit more pheromone if they eventually find food.

Instead of designing complex and centralized systems, nowadays designers rather prefer to work with many small and autonomous agents. Each agent may prescribe to a global strategy. An agent acts on the simplest of rules. The many agents co-operating within the system can solve very complex problems with a minimal design effort. In General, multi-agent systems that use some swarm intelligence are said to be swarm intelligent systems. They are mostly used as search engines and optimization tools.

The goal of this volume has been to offer a wide spectrum of sample works developed in leading research throughout the world about innovative methodologies of swarm intelligence and foundations of engineering swarm intelligent systems as well as application and interesting experiences using the particle swarm optimisation, which is at the heart of computational intelligence. The book should be useful both for beginners and experienced researchers in the field of computational intelligence.

Part I: Methodologies Based on Particle Swarm Intelligence

In Chapter 1, which is entitled *Swarm Intelligence: Foundations, Perspectives and Applications*, the authors introduce some of the theoretical foundations of swarm intelligence. They focus on the design and implementation of the Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) algorithms for various types of function optimization problems, real world applications and data mining.

In Chapter 2, which is entitled *Waves of Swarm Particles (WoSP)*, the author introduce an adaption of the conventional particle swarm algorithm that converts the behaviour from the conventional search and converge to an endless cycle of search, converge and then diverge to carry on searching. After introducing this new waves of swarm particles (WoSP) algorithm, The author present its behaviour on a number of problem spaces is presented. The simpler of these problem spaces have been chosen to explore the parameters of the new algorithm, but the last problem spaces have been chosen to show the remarkable performance of the algorithm on highly deceptive multi dimensional problem spaces with extreme numbers of local optima.

In Chapter 3, which is entitled *Grammatical Swarm: A variable-length Particle Swarm Algorithm*, the authors examine a variable-length Particle Swarm Algorithm for Social Programming. The Grammatical Swarm algorithm is a form of Social Programming as it uses Particle Swarm Optimisation, a social swarm algorithm, for the automatic generation of programs. The authors extend earlier work on a fixed-length incarnation of Grammatical Swarm, where each individual particle represents choices of program construction rules, where these rules are specified using a Backus-Naur Form grammar. The authors select benchmark problems from the field of Genetic Programming and compare their performance to that of fixed-length Grammatical Swarm and of Grammatical Evolution. They claim that it is possible to successfully generate programs using a variable-length Particle Swarm Algorithm, however, based on the problems analysed they recommend to exploit the simpler bounded Grammatical Swarm.

In Chapter 4, which is entitled *SWARMs of Self-Organizing Polymorphic Agents*, the authors describe a SWARM simulation of a distributed approach to fault mitigation within a large-scale data acquisition system for BTeV,

a particle accelerator-based High Energy Physics experiment currently under development at Fermi National Accelerator Laboratory. Incoming data is expected to arrive at a rate of over 1 terabyte every second, distributed across 2500 digital signal processors. Through simulation results, the authors show how lightweight polymorphic agents embedded within the individual processors use game theory to adapt roles based on the changing needs of the environment. They also provide details about SWARM architecture and implementation methodologies.

Part II: Experiences Using Particle Swarm Intelligence

In Chapter 5, which is entitled *Swarm Intelligence — Searchers, Cleaners and Hunters*, the authors examine the concept of swarm intelligence through three examples of complex problems which are solved by a decentralized swarm of simple agents. The protocols employed by these agents are presented, as well as various analytic results for their performance and for the problems in general. The problems examined are the problem of finding patterns within physical graphs (e.g. *k-cliques*), the *dynamic cooperative cleaners* problem, and a problem concerning a swarm of UAVs (unmanned air vehicles), hunting an evading target (or targets).

In Chapter 6, which is entitled *Ant Colony Optimisation for Fast Modular Exponentiation using the Sliding Window Method*, the authors exploit the ant colony strategy to finding optimal addition sequences that allow one to perform the pre-computations in window-based methods with a minimal number of modular multiplications. The authors claim that this improves the efficiency of modular exponentiation. The author compare the addition sequences obtained by the ant colony optimisation to those obtained using Brun's algorithm.

In Chapter 7, which is entitled *Particle Swarm for Fuzzy Models Identification*, the authors present the use of Particle Swarm Optimization (PSO) algorithm for building optimal fuzzy models from the available data. The authors also present the results based on selection based PSO variant with lifetime parameter that has been used for identification of fuzzy models. The fuzzy model identification procedure using PSO as an optimization engine has been implemented as a Matlab toolbox and is presented in the next chapter. The simulation results presented in this chapter have been obtained through this toolbox. The toolbox has been hosted on `SourceForge.net`, which is the world's largest development and download repository of open-source code and applications.

In Chapter 8, which is entitled *A Matlab Implementation of Swarm Intelligence based Methodology for Identification of Optimized Fuzzy Models*, the authors describe the implementation of the fuzzy model identification procedure (see Chapter 7) using PSO as an optimization engine. This toolbox provides the features to generate Mamdani and Singleton fuzzy models from

the available data. The authors claim that this toolbox can serve as a valuable reference to the swarm intelligence community and others and help them in designing fuzzy models for their respective applications quickly.

We are very much grateful to the authors of this volume and to the reviewers for their tremendous service by critically reviewing the chapters. The editors would also like to thank Prof. Janusz Kacprzyk, the editor-in-chief of the Studies in Computational Intelligence Book Series and Dr. Thomas Ditzinger from Springer-Verlag, Germany for their editorial assistance and excellent collaboration to produce this scientific work. We hope that the reader will share our excitement on this volume and will find it useful.

March 2006

Nadia Nedjah
Luiza M. Mourelle

State University of Rio de Janeiro
Brazil

Contents

List of Figures

List of Tables

5

Swarm Intelligence — Searchers, Cleaners and Hunters ⋆ ⋆⋆

Yaniv Altshuler[1], Vladimir Yanovsky[1], Israel A. Wagner[1,2], and Alfred M. Bruckstein[1]

[1] Computer Science Department, Technion, Haifa 32000 Israel.
{yanival, volodyan, wagner, freddy}@cs.technion.ac.il
[2] IBM Haifa Labs, MATAM, Haifa 31905 Israel.
wagner@il.ibm.com

This chapter examines the concept of *swarm intelligence* through three examples of complex problems which are solved by a decentralized swarm of simple agents. The protocols employed by these agents are presented, as well as various analytic results for their performance and for the problems in general. The problems examined are the problem of finding patterns within physical graphs (e.g. *k-cliques*), the *dynamic cooperative cleaners* problem, and a problem concerning a swarm of UAVs (unmanned air vehicles), hunting an evading target (or targets). In addition, the work contains a discussion regarding open questions and ongoing and future research in this field.

5.1 Introduction

Significant research effort has been invested during the last few years in design and simulation of intelligent swarm systems. Such systems can generally be defined as decentralized systems, comprising relatively simple agents which are equipped with limited communication, computations and sensing abilities, designed to accomplish a given task ([8, 9, 10, 11]).

However, much work is yet to be done for such systems to achieve sufficient efficiency. This is caused, among others, by the fact that the geometrical theory of such multi agent systems (which is used to tie geometric features of the environments to the systems' behaviors) is far from being satisfactory, as pointed out in [12] and many others. Furthermore, while strong analysis

⋆ This research was partly supported by the Ministry of Science Infrastructural Grant No. 3-942
⋆⋆ This research was partly supported by the Devorah fund

Y. Altshuler et al.: *Swarm Intelligence — Searchers, Cleaners and Hunters*, Studies in Computational Intelligence (SCI) **26**, 93–132 (2006)
www.springerlink.com

of a swarm protocol and its performance is crucial for the development of stronger protocols and for overcoming the *resource allocation problem* users of multi agent system often face, most of the works in the field present merely a superficial analysis of the protocols, often in the sole form of experimental results.

This work introduces three complex problems which are to be solved by such intelligent swarms, utilizing specially designed protocols. The protocols and their performance are analyzed, and experimental results concerning the actual performance of the protocols are presented.

A more elaborated overview of previous work which concerns swarm intelligence is presented in Section 5.1.1 while details regarding the motivation behind this research appears in Section 5.1.2. A key element in design and analysis of swarm based systems and of swarm algorithms is the simplicity of the agents (in means on sensing and computation capabilities, communication, etc.). A discussion concerning this issue appears in Section 5.1.3.

Several works considered multi agents robotics in static environments. Such works can be found in [1], [2], [7] and elsewhere. These works present, among other results, protocols that assume the only changes taking place in the environment to be caused through the activity of the agents. The first problem presented in this work is a problem in which the agents must work in dynamic environments — an environment in which changes may take place regardless of the agents' activity. This problem is a dynamic variant of the known *Cooperative Cleaners* problem (described and analyzed in [1]). This problem assumes a grid, part of which is 'dirty', where the 'dirty' part is a connected region of the grid. On this dirty grid region several agents move, each having the ability to 'clean' the place ('tile', 'pixel' or 'square') it is located in, while the goal of the agents is to clean all the dirty tiles (in the shortest time possible). The dynamic variant of the problem (described in Section 5.2.1) involves a deterministic evolution of the environment, simulating a spreading *contamination* (or spreading *fire*).

The Dynamic Cooperative Cleaners problem was first introduced in [3], which also included a cleaning protocol for the problem, as well as several analytic bounds for the time it takes agents which use this protocol to clean the entire grid. A summary of those results appears in Section 5.2.4, while Section 5.2.5 describes a method of using a prior knowledge concerning the initial shape in order to improve its cleaning time.

The second problem presented is the *Cooperative Hunters* problem. This problem examines a scenario in which one or more *smart targets* (i.e. a platoon of T-72 tanks, a squad of soldiers, etc') are moving in a predefined area, trying to avoid detection by a swarm of UAVs (unmanned air vehicles). The UAV swarm's goal is to find the target (or targets) in the shortest time possible, meaning, we must guarantee that there exists time t in which all the escaping targets are detected, and that this t is minimal. While the swarm comprises relatively simple UAVs, which lack prior knowledge of the initial positions

of the targets, the targets possess full knowledge of the whereabout of the swarm's agents, and are capable of intelligent evasive behavior.

A basic protocol for this problem and its analysis appears in [15]. However, this protocol assumes that the area in which the targets can move is known to the UAVs. Furthermore, although trying to minimize the communication between the swarm's agents, the work in [15] still requires a relatively high amount of explicit cooperation between the agents, which can be obtained through the use of a relatively high amount of communication.

This work contains a protocol for the requested task, which assumes no previous knowledge considering the area to be searched, and uses only a limited communication between the agents. The problem, the protocol and an analysis of its performance are presented in Section 5.3.

The third problem presented in this work is the *Physical k-Clique* problem, where a swarm comprising n mobile agents travels along the vertices of a physical graph G, searching for a *clique* of size k. A physical graph refers to a graph whose edges require a certain amount of time (or resources) for traveling along. Furthermore, information regarding the graph is available to the mobile agents only locally (meaning that the agents gather information only regarding the vertices they travel through). Thus, the complexity of the problem is measured in the number of edges traveled along, and not in the computational resources used by the agents. The work presents a search protocol for the agents, as well as experimental results for its performance (Section 5.4).

Section 5.5 contains a discussion regarding the problems, and details about extended research, which is currently being performed by the authors.

5.1.1 Swarm Intelligence — Overview

The area of multi agents and multi robotics distributed systems has become increasingly popular during the last two decades. Many applications, mainly in the contexts of computer networks, distributed computing and robotics, are nowadays being designed using techniques and schemes which are based on concepts derived from multi agents, or *swarms*, research.

The basic paradigm behind multi agents based system is that many tasks can be more efficiently completed by using multiple simple autonomous agents (robot, software agents, etc.) instead of a single sophisticated one. Regardless of the improvement in performance, such systems are usually much more adaptive, scalable and robust than those based on a single, highly capable, agent.

A multi agent system, or a swarm, can generally be defined as a decentralized group of multiple autonomous agents, either homogenous or heterogenous, such that those agents are simple and possess limited capabilities. Section 5.1.3 discusses the various limitations expected from such agents, while a commonly used taxonomy for multi agent robotics can be found in [47].

The inherent complexity of distributed multi agent systems, which is derived from the multitude of free variables involved in the operation and the

decision process of such systems, makes their analysis extremely difficult. This may explain the relatively small number of theoretical results in this field, and the fact that most works in this field are justified through experimental results, or by analysis of simple cases only. Further hardness is derived from the fact that distributed multi agent systems are complex systems by nature, with all the classical features of such systems. Thus, the field of multi agent systems becomes an exciting and largely unexplored field for research and development.

Furthermore, while many works have been done considering multi agents in static environments (such as [1, 2, 7]), only a few works examined multi agent systems that operate in environments that change not only through the activity of the agents. As a results, the field of multi agent systems in dynamic environments is an exceptionally fascinating aspect of multi agents research.

Many research efforts have examined distributed systems models inspired by biology (see [55] or for behavior based control model — [64, 59], flocking and dispersing models — [78, 67, 69] and predator-prey approach — [61, 73]), physics [56], and economics [48, 49, 50, 51, 52, 53, 54].

Capabilities that have been of particular emphasis include task planning [57], fault tolerance [82], swarm control [79], human design of mission plans [77], role assignment [88, 65, 80], multi-robot path planning [89, 76, 70, 93], traffic control [84], formation generation [58, 96, 97], formation keeping [60, 91], target tracking [83] and target search [75].

Another interesting field of research is that of the biology inspired *ACO* metaheuristic for solving problems in dynamic environments. In [112], pheromones are treated as a *desirability feature* and are placed by the agents on the graph's edges. The information stored in these pheromones is essentially an implicit measurement of the probability of the edges to belong to the optimal solution for the problem (*TSP*, in that case). Other applications are -

- Sequential ordering problem [113].
- Quadratic assignment problem [114].
- Vehicle routing problem [115].
- Scheduling problem [116].
- Graph colouring problem [117].
- Partitioning problems [118].
- Problems in telecommunications networks [119], [120].

5.1.2 Swarm Intelligence — Motivation

The experience gained due to the growing demand for robotics solutions to increasingly complex and varied challenges has dictated that a single robot is no longer the best solution for many application domains. Instead, teams of robots must coordinate intelligently for successful task execution.

[99] presents a detailed description of multi robots application domains, and demonstrates how multi robots systems are more effective than a single

robot in many of these domains. However, when designing such systems it should be notice that simply increasing the number of robots assigned to a task does not necessarily improve the system's performance — multiple robots must cooperate intelligently to achieve efficiency.

Following are the main inherent advantages of a multi agent robotics (note that much of them hold for other multi agent systems, such as a distributed anti-virus mechanism, for example) :

- The benefit of parallelism — in task-decomposable application domains, robot teams can accomplish a given task more quickly than a single robot by dividing the task into sub-tasks and executing them concurrently. In certain domains, a single robot may simple no be able to accomplish the task on its own (e.g. carrying a large and heavy object).
- Robustness — generally speaking, a team of robots provides a more robust solution by introducing redundancy, and by eliminating any single point of failure. While considering the alternative of using a single sophisticated robot, we should note that even the most complex and reliable robot may suffer an unexpected malfunction, which will prevent it from completing its task. When using a multi agent system, on the other hand, even if a large number of the agents stop working from some reason, the entire group will often still be able to complete its task, albeit slower. For example, for exploring a hazardous region (such as a minefield or the surface of Mars), the benefit of redundancy and robustness offered by a multi agent system is highly noticeable.
- Adaptivity and Locality — the unit of a multi agents based systems has the ability of dynamically reallocating sub-tasks between the group of agents, thus adapting to unexpected changes in the environment. Furthermore, since the system is decentralized, it can respond relatively quickly to such changes, due to the benefit of locality, meaning — the ability to perform changes in the operation of a sub group of agents without the need to notify or request approval from any centralized "leader". Note that as the system comprises more agents, this advantage becomes more and more noticeable.
- Scalability — as in the previous section, as a multi agent system becomes larger, its relative performance in comparison to a centralized system becomes better. The scalability of multi agent systems is derived from the low overhead (both in communication and computation) such system possess. As the tasks assigned nowadays to multi agents based systems become increasingly complex, so does the importance of the high scalability of the systems.
- Heterogeneousness — since a group of agents may be heterogenous, it can utilize "*specialists*" — agents whose physical properties enable them to perform efficiently certain well defined tasks.
- Flexibility — as multi agent systems possess a great deal of internal complexity, such systems are capable of presenting a wide variety of behavior

patterns. As a result, many kinds of tasks can be carried out by the same multi agent system. For example, a group of simple mobile robots can form a rigid line in order to scan an area for evading target, traverse an area in order to implement a "peeling" mechanism (such as [1] or [3]), or patrol an area in order to minimize the longest time between two visits in a certain point. This flexibility allows designers of such systems to use generic components and thus design fast and cheap systems for a given problem, whereas using a single robot requires designers to produce a special robot for each task.

Following are several examples for typical applications for which multi robotics system may fit successfully :

- Covering — in order to explore an enemy terrain, or clean a pre-defined minefield. May also be used for virtual spaces, such as a distributed search engine in a computer network [1, 3].
- Patrolling — for example, guarding a museum against thieves [121, 122].
- Cooperative attack, which require the cooperative and synchronized efforts of a large number of autonomous robots [15].
- Construction of complex structures and self-assembling (for example, see work on reconfigurable robots in [63, 62, 66, 86, 94, 72, 90, 95]).
- Missions which by nature require an extremely large number of extremely small units. For example, nano-robots performing medical procedures inside a human body [123, 124].
- Mapping and localizing — one example of this work is given in [71], which takes advantage of multiple robots to improve positioning accuracy beyond what is possible with single robots. Another example for a heterogenous localization system appears in [98].
- Environment manipulation — like a distributed system of transporting heavy payloads (see [85, 87, 68, 74, 92]).

5.1.3 Swarm Intelligence — Simplicity

A key principal in the notion of swarms, or multi agent robotics is the simplicity of the agents. Since "simplicity" is a relative description by its nature, the meaning is that the agents should be "much simpler" than a "single sophisticated agent" which can be constructed.

For example, cheap unmanned air vehicles which can fly at the speed of 200 MPH and detect targets at radius of 2 miles, and are able to broadcast to a range of 1 mile are "much simpler" than an F-16 airplane which can fly at a top speed of 1500 MPH, equipped with a radar which can detect targets at a range of 50 miles and use satellite based communication system. As technology advances, the criteria for future "simple agents" may of course be changed.

As a result, the resources of such simple agents are assumed to be limited, with respect to the following aspects :

- Memory resources — basic agents should be assumed to contain only $O(1)$ memory resources. This usually impose many interesting limitation on the agents. For example, agents can remember the history of their operation to only a certain extent. Thus, protocols designed for agents with such limited memory resources are usually very simple and try to solve the given problem by defining some (necessarily local) basic patterns. The task is completed by repetition of this patterns by a large number of agents.
  In several cases, stronger agents may be assumed, whose memory size is proportional to the size on the problem (i.e. $O(n)$). Such models may be used for example for problems when assuming that the agents are UAVs.
- Sensing capabilities — defined according to the specific nature of the problem. For example, for agents moving along a 100×100 grid, the sensing radius of the agents may be assumed to be 3, but not 40.
- Computational resources — although agents are assumed to employ only limited computational resources, a formal definition of such resources is hard to define. In general, most of the polynomial algorithms may be used.

Another aspect of swarms' and swarm algorithms' simplicity is the use of communication. The issue of communication in multi agent systems has been extensively studied in recent years. Distinctions between implicit and explicit communication are usually made, in which implicit communication occurs as a side effect of other actions, or "through the world" (see, for example [81]), whereas explicit communication is a specific act designed solely to convey information to other robots on the team.

Explicit communication can be performed in several ways, such as a short range point to point communication, a global broadcast, or by using some sort of distributed shared memory. Such memory is often treated to as a *pheromone*, used to convey small amounts of information between the agents [100, 101, 102]. This approach is inspired from the coordination and communication methods used by many social insects — studies on ants (e.g. [103, 104]) show that the pheromone based search strategies used by ants in foraging for food in unknown terrains tend to be very efficient.

Generally, we aspire that the agents will have as little communication capabilities as possible. Although a certain amount of implicit communication can hardly be avoided (due to the simple fact that by changing the environment, the agents are constantly generating some kind of implicit information), explicit communication should be strongly limited or avoided altogether. However, in several cases it can be shown that by adding a limited amount of communication to the agents' capabilities, much stronger systems can be produced.

5.2 The Dynamic Cooperative Cleaners (DCC) Problem

5.2.1 The Problem's Definition

Let us assume that the time is discrete. Let G be a two dimensional grid, whose vertices have a binary property of '*contamination*'. Let $cont_t(v)$ state the contamination state of the vertex v in time t, taking either the value "*on*" or "*off*". Let F_t be the dirty sub-graph of G at time t, i.e. $F_t = \{v \in G \mid cont_t(v) = on\}$. We assume that F_0 is a single connected component. Our algorithm will preserve this property along its evolution.

Let a group of k agents that can move across the grid G (moving from a vertex to one of its *4-Connected* neighbors in one time step) be arbitrarily placed in time t_0 on the boundary of F_0 (we focus on the cleaning problem, and not on the discovery problem). *4-Connected* vertices are a pair of vertices who share a common border while *8-Connected* vertices are vertices which share either common border or a common corner.

Each agent is equipped with a sensor capable of telling the condition of the square it is currently located in, as well as the condition of the squares in the $8-Neighbors$ group of this square. An agent is also aware of other agents which are located in its square, and all the agents agree on a common "north". Each square can contain any number of agents simultaneously.

When an agent moves to a vertex v, it has the possibility of causing $cont(v)$ to become *off*. The agents do not have any prior knowledge of the shape or size of the sub-graph F_0 except that it is a single connected component.

Every d time steps the contamination spreads. That is, if $t = nd$ for some positive integer n, then $(\forall v \in F_t \ , \ \forall u \in 4{-}Neighbors(v) \ : \ cont_{t+1}(u) = on)$.

The agents' goal is to clean G by eliminating the contamination entirely, so that $(\exists t_{success} \ : \ F_{t_{success}} = \emptyset)$. In addition, it is desired that this $t_{success}$ will be minimal.

In this work we demand that there is no central control and that the system is fully 'de-centralized' (i.e. all agents are identical and no explicit communication is allowed).

5.2.2 Solving the Problem — Cleaning Protocol

For solving the *Dynamic Cooperative Cleaners* problem the **SWEEP** cleaning protocol was suggested [3]. This protocol can be described as follows. Generalizing an idea from computer graphics (which is presented in [14]), the connectivity of the *contaminated* region is preserved by preventing the agents from cleaning what is called *critical points* — points which disconnect the graph of contaminated grid points. This ensures that the agents stop only upon completing their mission. An important advantage of this approach, in addition to the simplicity of the agents, is fault-tolerance — even if almost all the agents cease to work before completion, the remaining ones will eventually complete the mission, if possible. The protocol appears in Figure 5.2.

In the spirit of [13] we consider simple robots with only a bounded amount of memory (i.e. a *finite-state-machine*). At each time step, each agent cleans its current location (assuming this is not a critical point), and moves to its *rightmost* neighbor (starting from the agent's previous location — a local movement rule, creating the effect of a clockwise traversal of the contaminated shape). As a result, the agents "peel" layers from the shape, while preserving its connectivity, until the shape is cleaned entirely.

Since we only allow agents to clean boundary points, we guarantee that no new "holes" are created. The simple-connectivity of F, if such exists, is thus kept. This however, does not hold for a certain family of initial shapes. A complete discussion regarding a criteria for F_0 which guarantees that new holes will not be created can be found in [4] and a comprehensive work discussion this issue is currently under preparation by the authors. In general, the preservation of the simple-connectivity is rather easily guaranteed for digitally convex shapes lacking areas which may turn into holes once F spreads. An example of a shape in which the preservation of the simple-connectivity of the shape cannot be guaranteed appears in Figure 5.1.

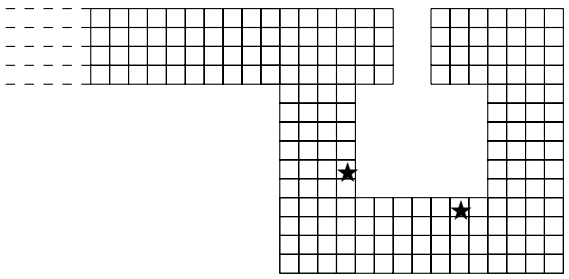


Fig. 5.1. A creation of a new hole around an agent. Notice that when the contamination spreads, it might trap one or more agents (denoted by stars) inside a "hole" that is created. In this case, the agents will continue cleaning the interior area of the hole.

5.2.3 Cleaning Protocol - Definitions and Requirements

Let $r(t) = (\tau_1(t), \tau_2(t), \ldots, \tau_k(t))$ denote the locations of the k agents at time t. The requested cleaning protocol is therefore a rule f such that for every agent i, $f(\tau_i(t), Neighborhood(\tau_i(t)), \mathcal{M}_i(t)) \in \mathcal{D}$, where for a square v, $Neighborhood(v)$ denotes the contamination states of v and its $8{-}Neighbors$, $\mathcal{M}_i$ is a finite amount of memory for agent i, containing information needed for the protocol (e.g. the last move) and $\mathcal{D} = \{`left', `right', `up', `down'\}$.

Let ∂F_t denote the boundary of F_t. A square is on the boundary if and only if at least one of its $8{-}Neighbors$ is not in F_t, meaning $\partial F = \{(x, y) \mid (x, y) \in F \wedge 8{-}Neighbors(x, y) \cap (G \setminus F) \neq \emptyset\}$.

The requested rule f should meet the following goals :

- **Full Cleanness** : $(\exists t_{success} \ : \ F_{t_{success}} = \emptyset)$. Notice that this demand is limited to cases where this is possible. Since we cannot know whether completing the mission is possible, we can only demand that when $d \to \infty$ the agents should achieve this goal.
- **Agreement on Completion** : within a finite time after completing the mission, all the agents must halt.
- **Efficiency** : in time and in agents' memory resources.

The requested rule should also be *fault tolerant*.

5.2.4 Dynamic Cooperative Cleaners — Results

Notice that analyzing the performance of the protocol is quite difficult. First, due to the preservation of the *critical points*, such points can be visited many times by the agents without being cleaned. Second, due to the dynamic nature of the problem the shape of the contaminated region can dramatically change during the cleaning process.

Another difficulty which arises is that of guaranteeing the completion of the cleaning by the agents. We must show that the agents are cleaning the contaminated region faster than the spreading of the last. Since given an instance of the problem, we know no easy way of knowing the minimal time it takes k agents to clean it, we cannot always foretell whether these k agents will be able to complete the mission successfully.

Let d denote the number of time steps between two contamination spreads. Then, for example, for any pair of values of d and k, let F_0 be a straight line of length $\lceil \frac{1}{8}d^2k^2 + dk + \frac{1}{2} \rceil$. Then, by applying the lower bound for the cleaning time (which was shown in [3]), we can see that the size of the shape is continually growing.

Thus, producing bounds for the proposed cleaning protocol is important for estimating its efficiency. Following is a summery of the bounds which were shown and proven in [3].

i. Given a contaminated shape F_0 with initial area of S_0, and k agents employing *any* cleaning protocol, following is a lower bound for S_t (the area of F in time t), and thus for the cleaning time of the agents :

$$S_{t+d} \geq S_t - d \cdot k + \sqrt{8 \cdot (S_t - d \cdot k) - 4}$$

where d is the number of time steps between spreads. Note that if $S_0 \gg d \cdot k$ then the sequence may be increasing and the agents cannot cease the fire. In addition, note that the bound is generic and applies regardless of the cleaning protocol used by the agents. The bound is based on the calculation of the maximal area which can be cleaned by k agents in d time steps, combined the minimal number of new contaminated squares which can be created ones F spreads. Additional details can be obtained in [3].

Protocol **SWEEP**(x, y) :
If (not **is-critical**(x, y)) and $((x, y) \in \partial F)$ and (there are no other agents in (x, y)) then
    Set $cont(x, y)$ to *off*; */*Clean current position*/*
If (x, y) has no contaminated neighbors then **STOP**;
If (there are no other agents in (x, y)) or (the agent has the highest **priority** among agents in (x, y)) then
    If $\neg((x, y) \in \partial F)$ and in the previous time step the agent's square was in ∂F then
        / Spread had occurred. Search for ∂F */*
        Move in 90° counterclockwise to the previous movement and skip the rest of the protocol;
    If $\neg((x, y) \in \partial F)$ and in the previous time step the agent's square was not in ∂F then
        / Keep seeking ∂F */*
        Move on the same direction as in the previous movement and skip the rest of the protocol;
    If $(x, y) \in \partial F$ then
        Go to the *rightmost* neighbor of (x, y);
End **SWEEP**;

Function **is-critical**(x, y) :
If (x, y) has two contaminated squares in its $4-Neighbors$ which are not connected via a sequence of contaminated squares from its $8-Neighbors$ then
    Return **TRUE**
Else
    Return **FALSE**;
End **is-critical**;

Function **priority**(i) :
$(x_0, y_0) = \tau_i(t-1)$;
$(x_1, y_1) = \tau_i(t)$;
Return $(2 \cdot (x_0 - x_1) + (y_0 - y_1))$;
End **priority**;

Procedure **INITIALIZE**() :
Choose a starting point on ∂F, p_0;
Put all the agents in p_0;
For $(i = 1; i \leq k; i++)$ do
    Start agent i according to the **SWEEP** protocol;
    Wait 2 time steps;
End **INITIALIZE**;

Fig. 5.2. The **SWEEP** cleaning protocol. The protocol is used by agent i which is located in square(x, y). The term *rightmost* is defined as "starting from the previous boundary point scan the neighbors of (x, y) in a clockwise order until you find another boundary point"

ii. Following is an upper bound for $t_{SUCCESS}$, the cleaning time of k agents, using the **SWEEP** cleaning protocol, for a given contaminated shape F_0 :

$$t_{quick\_clean} \triangleq \frac{8(|\partial F_0| - 1) \cdot (W(F_0) + k)}{k} + 2k$$

If $t_{quick\_clean} \leq d$ then $t_{SUCCESS} = \lceil t_{quick\_clean} \rceil$. Otherwise $(t > d)$:
If F_0 is convex***, find the minimal t for which :

$$\sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \geq \gamma + \frac{8}{k} \cdot \left\lfloor \frac{t}{d} \right\rfloor$$

where

$$\gamma \triangleq \frac{8(k + W(F_0))}{k} - \frac{d - 2k}{|\partial F_0| - 1}$$

Otherwise ($t > d$, F_0 is not convex), find the minimal t for which :

$$\sum_{i=d+1}^{t} \frac{1}{S_0 - 1 + 2\lfloor \frac{i}{d} \rfloor^2 + (c_0 + 2)\lfloor \frac{i}{d} \rfloor} \geq$$
$$\geq \alpha + \frac{8}{2k}\sqrt{\beta + 4\Big(\Big\lfloor \frac{t}{d} \Big\rfloor^2 + \Big\lfloor \frac{t}{d} \Big\rfloor\Big)}$$

where

$$\alpha \triangleq 8 + \frac{8}{2k} - \frac{d - 2k}{|\partial F_0| - 1} \quad \textit{and} \quad \beta \triangleq 2S_0 + 2c_0 - 1$$

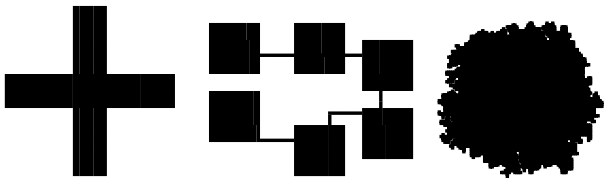
In both cases $t_{SUCCESS} = t$.
In the above, d is the number of time steps between contamination spreads, c_0 is the circumference of F_0, S_0 is the area of F_0 and $W(F_0)$ denotes the maximal of the shortest distances between an internal square of F_0 and a non-critical square of ∂F_0.
This bound is produced by defining a "depth" of a shape as the maximal shortest path between an internal point of F and a non-critical point on the boundary of F. By showing that once an agent traverses F using the **SWEEP** protocol the depth of F is decreased, and by limiting the increase the the depth of F due to contamination spreads, an upper bound over the cleaning time is produced. Additional details can be obtained in [3].

iii. For a given contaminated shape F_0, an upper bound for the number of agents needed to apply the **SWEEP** protocol, for a successful cleaning of the shape, is derived from the cleaning time bound above.

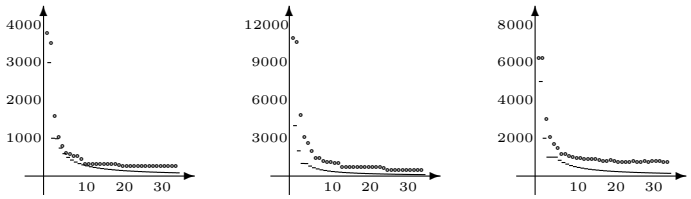
A computer simulation, implementing the **SWEEP** cleaning protocol, was constructed. The simulation examined shapes of various geometric features,

*** Meaning that for every two squares in F_0 there is a $4\text{–}Neighbors$ "shortest path" between them, which is entirely in F_0.

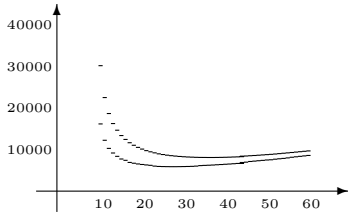
and tracked the cleaning time of k agents ($k \in [1, 100]$) which used the **SWEEP** protocol. Following is a sample of the results, including the predictions of the lower and upper bounds. The sample includes a "*cross*" of size 2960, "*rooms*" of size 3959, and a random, semi-convex, shape of size 5000, where $d = 1000$:



The lower curves represent the results predicted by the lower bound, while the upper curves represent the actual cleaning time, produced by the simulations performed (the graphs present the cleaning time as a function of the number of agents). The left graph presents the results that were produced by the "cross", etc'.



The following graph contains the upper bound for the "cross". Notice that the lower, tighter, curve was produced when taking into account that the "cross" shape is "convex":



5.2.5 Dynamic Cooperative Cleaners — Convex Transformation

While [3] presented analytic upper bounds for the cleaning time of agents using the **SWEEP** protocol, assuming no prior knowledge concerning F_0 by the agents, in many cases such agents may indeed possess such information. Should this information be available to the agents, a method of employing it in order to improve the cleaning time should be devised.

While examining the simulation results, it can easily be seen that since the circumference of F_0 is rarely convex, valuable time is lost on cleaning

the concave parts of it. Since our goal should be to minimize the size of the bounding sphere of the contaminated shape rather than the size of the contaminated shape itself, had the agents been aware of the existence of such parts, they shouldn't have bothered cleaning them. In order to satisfy this goal, when given information about F_0 the agents can calculate its convex hull (for example, using Graham scan) $C_0 \triangleq ConvexHull(F_0)$. Then, by using the **SWEEP** protocol over C_0 (instead of F_0) the agents are able to speed up the cleaning (since when C_0 is clean, so must be F_0). Using the upper bound of [3] allows us to predict this improvement.

Fig. 5.3 contains the results of the proposed transformation, for the *rooms* shape (size = 3959, circumference = 716) and the *random semi convex* shape (size = 5000, circumference = 654) of [3]. Since the *cross* which is presented in [3] is convex, the transformation will not change its cleaning time. The value of d for both shapes was chosen to be 1000. As can be seen, the use of convex transformation improved the performance of the agents by an average of approximately 35%.

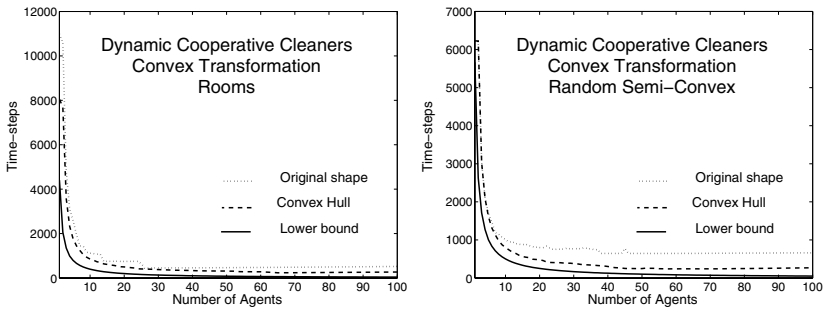


Fig. 5.3. Improvement in cleaning time of Convex-hull.

5.3 Cooperative Hunters

Significant research effort has been invested during the last few years in design, analysis and simulation of multi-agent systems design for searching areas (either known or unknown) [107, 108, 109, 110, 111]. While in most works the targets are assumed to be idle, recent works consider dynamic targets, meaning — target which by detecting the searching agents from a long distance, try to avoid detection by evading the agents.

Such problem is presented in [15], where a swarm of UAVs (unmanned aerial vehicles) is used for searching after one or more evading "smart targets". The UAVs swarm's goal is to guarantee that there exists time t in which all the escaping targets are detected, and that this t is minimal. In the current

work, the problem described above will be denoted as the *Cooperative Hunters* problem.

5.3.1 Problem

Let us assume the time is discrete, while every time step lasts c_{time} seconds. Let G be a two dimensional grid, such that every vertex corresponds to an area in the map of size $c_{size} \times c_{size}$ square meters. Let us assume that each UAV moves at speed $1 \cdot c_{size}$ meters per time step. Let us assume that the targets move at speed $v_{target} \cdot c_{size}$ meters per time step (thus, c_{time} can be adjusted accordingly).

We assume that each UAV is equipped with sensors able of detecting the targets within its current vertex of G. The targets however, can spot the searcher from a great distance (considered to be infinite, and beyond the searcher's sensors range) and subsequently, manoeuver in order to evade detection. Once the searcher detects the target, it intercepts it.

Each UAV is aware of its current location (using a GPS receiver) and while flying over vertex v, can identify whether or not vertex v and its $8\text{–}Neighbors$ are part of the area to be searched. There is no limitation over the shape of the area to be searched, although it is assumed to be simply connected.

The UAV's communicate by using a wireless channel, while the information transmitted over this channel should be kept to a minimum.

The number of hiding targets can be either known to the swarms, or alternatively, the UAVs might not know the exact number of hiding targets (in which case the swarm will continue searching until guaranteeing that there is no longer a place in which the targets can hide). The goal of the swarm is to detect all hiding targets, in a minimal amount of time.

5.3.2 Motivation

The search strategy suggested in [15] for solving this problem defines *flying patterns* that the UAVs follow, which are designed for scanning the (rectangular) area in such a way that the targets cannot re-enter sub-areas which were already scanned by the swarm, without being detected by some UAV. Note that this protocol assumes that the area in which the targets can move is known to the UAVs in advance, and must be rectangular in shape.

However, this may not always be the case — a swarm of UAVs may be launched into an area whose shape and size is unknown to the UAVs prior to the beginning of the operation. For example, a swarm of UAVs might be used in order to hunt guerrilla forces, hiding in a city. These forces can be identified by the UAVs, since they use certain vehicles, or carrying certain electronic equipment or weapons, which can be identified by the UAVs' sensors. In this example, the targets can move only in the boundaries of the city, but the operators of the system may lack information regarding the exact boundaries of the city (since they may have only old and outdated satellite images of the

area, or since large portions of the area was destroyed during recent fights). Thus, a method for searching without relying on previous knowledge of the searched area must be obtained.

The searching protocol presented in this work uses no prior knowledge of the area. The only assumption made is that the swarm's agents are capable of identifying the boundaries of the searched area, as they pass over them. In the previous example this means that once a UAV passes over the boundary of the city it can detect it. Therefore, a swarm whose agents contains no knowledge regarding the area to be searched, can still perform the searching task, by employing this protocol.

Furthermore, the presented protocol is very efficient in means of the simplicity of the agents, and the low communication between them. This makes the use of a simple protocol a notable advantage. While the protocol of [15] requires a relatively high amount of explicit cooperation between the UAVs (which can be obtained through the use of a relatively high amount of communication), the communication between the agents which is needed for a swarm using the presented protocol is extremely limited, and is bounded by $6 \cdot k$ bits per time step (k being the number of agents). Another advantage of the presented protocol is its fault tolerance, meaning — even if many UAVs malfunction or be shot down, the swarm will still be able to complete the task, albeit slower.

5.3.3 General Principle

Although the initial problem is that of searching for hiding targets within a given area, we shall consider an alternative, yet equivalent problem — the *dynamic cooperative cleaners* problem, presented in section 5.2.1.

Notice that from a cleaning protocol which is used by agents in order to solve the *DCC* problem, a protocol for the cooperative hunters problem can be derived. This is done by defining the entire area G as 'contaminated'. A 'clean' square (either a square which has not been contaminated yet, or a square which was cleaned by the UAVs) represents an area which is guaranteed not to contain any targets. By using the fact that the contamination is spreading, we simulate the fact that the targets may manoeuver around the UAVs, in order to avoid detection — if vertex v is contaminated then it may contain a target, thus, after $\frac{1}{v_{target}}$ seconds, this target could have moved from v to one of its neighbors, had it been in v. As result, after $\frac{1}{v_{target}}$ seconds all the neighbors of v become contaminated. In other words, the spreading contaminated simulates a *danger diffusion* that represents the capability of a square to contain a target.

The agents' goal is to eliminate the contaminated area — eliminate the places which the targets may be hiding in. Once there are no longer squares in which the targets may be hiding, the swarm is guaranteed to have detected all evading targets. Note that our demands regarding no prior knowledge of

the search area are met, since the cooperative cleaners problem do not assume such knowledge.

5.3.4 Search Protocol

Let each UAV i hold G_i — a bitmap of G. Let every G_i be initialized to zeros (e.g. "clean"). Let each UAV i contain a hash table of vertices — f_i which for every vertex can return on or off. The default for all the vertices is off. The list f_i represents the vertices which are known to be within the area to be searched.

Every time a UAV flying over vertex v identifies v or one of its neighbors to be a part of the area to be searched, if $f_i(v) = off$ it sets the corresponding vertices of G_i to 1, sets $f_i(v)$ to be on, and broadcasts this information to the other UAVs. Once a UAV receives a transmission that vertex v is part of the area to be searched, it sets $f_i(v)$ to on and sets the corresponding vertex in G_i to 1. Every time a UAV moves it broadcasts the direction of its movement to the rest of the UAVs (north, south, west or east).

Notice that every time step each UAV broadcasts the new squares which are parts of G (which are set to 1 in G_i), and the squares it "cleaned" by flying over them (which are set to 0). Thus, the G_i and f_i of all UAVs are kept synchronized. Since v_{target} is known to the UAVs, they can simulate the spreading contamination, by performing $(\forall v \in G_i \ , \ \forall u \in Neighbors(v) \ : state(u) = 1)$ every $\frac{1}{v_{target}}$ time steps. Thus, the bitmaps G_i always represent the correct representation of the area still to be cleaned.

The direction of movement and the decision whether or not to clean a vertex are determined using some cleaning protocol (for example, the **SWEEP** protocol of [3]). Notice that all the analytic bounds over the cleaning time of a cleaning protocol are immediately applicable for our hunting protocol. Whenever a UAV cleans a certain vertex, it sets this vertex in G_i to be 0, and broadcasts this information. Once a UAV receives such a transmission, it sets the vertex corresponding to the new location of the transmitting UAV to 0.

The UAVs are assume to be placed on the boundary of the area to be searched. Thus, each G_i immediately contains at least one vertex whose value is 1. As a result, for G_i to contains only zeros, the UAVs must have visited all the vertices of G and had made sure that no target could have escaped and "re-contaminated" a clean square. When G_i becomes all zeros UAV i knows that the targets have been found, and stops searching.

Since each time step, each UAV can move in at most 4 directions (i.e. 2 bits of information), clean at most a single vertex (i.e. 1 bit of information), and broadcast the status of 8 neighbor vertices (i.e. 3 bits of information), the communication is limited to 6 bits of information per UAV per time step.

5.3.5 Results

As mentioned in previous sections, by showing that any "cleaning protocol" may be used as the core component of a "hunting protocol" for the problem, all the analytic results concerning this cleaning protocol are immediately applicable. Specifically, by using the SWEEP cleaning protocol, all of the analytic bounds for its cleaning time (i.e. those mentioned in section 5.2.4 or in [3]) be utilized.

5.4 Physical k-Clique

5.4.1 Physical Graphs

The term *physical graph* denotes a graph $G(V, E)$ in which information regarding its vertices and edges is extracted using *I/O heads*, or *mobile agents*, instead of the "random access I/O" usually assumed in graph theory. These agents can physically move between the vertices of V along the edges of E, according to a predefined, or an on-line algorithm or protocol.

Moving along an edge e however, requires a certain amount of *travel efforts* (which may represent time, fuel, etc'). Thus, the complexity of algorithms which work on physical graphs is measured by the total travel efforts required, which equals the number of edges traveled by the agents. We assume that each edge requires exactly one unit of travel effort.

Physical graphs are conveniently used in order to represent many "real world problems", in which the most efficient algorithm is not necessarily the one whose computational complexity is the minimal, but rather one in which the agents travel along the minimal number edges. For example, the *Virtual Path Layout* problem, concerning the finding of a virtual graph of a given diameter, and its embedding in the physical graph such that the maximum load is minimized (see [36] and [37]).

Problems in physical graphs are thus variants of "regular" graph problems, such as finding a *k-clique* in a graph (description and algorithms can be found in [29]), graph and subgraph isomorphism (description and algorithms can be found in [17, 18, 19, 20, 21]), exploration problems (solved for example by *Breadth First Search* (BFS) and *Depth First Search* (DFS) algorithms [34]), etc., whose input graph is a physical graph. Thus, the complexity of these problems is measured as the number of edges an agent (or agents) solving the problem will travel along.

There is a special type of graph problems which can also be ideally described as physical graph problems. Such problems are those in which a *probabilistic*, or *real time* algorithm is used to return a solution which is not necessarily optimal. While a probabilistic algorithm returns a solution which is correct in a probability of $(1 - \epsilon)$ (for as small ϵ as we require), a real time algorithm can be asked at any stage to return a solution, whereas the quality

of the solutions provided by the algorithm improves as time passes. Using such probabilistic or real time algorithms, the computational complexity of many problems can often be reduced from exponential to polynomial (albeit we are not guaranteed of finding the optimal solution). Such algorithms can be found for example in [22, 23, 24] (graph isomorphism) and [25, 31, 26] (distributed search algorithms such as *RTA**, *PBA**, *WS_PBA**, *SSC_PBA** and *BSA**). The physical variants of such problems can be thought of as a swarm of mobile agents, traveling the graph and collecting new information during this process. As time advances, more information is gathered by the agents, causing the quality of the solutions provided by the agents to improve.

Notice that while an algorithm which assumes a random access I/O model (from now on be referred to as *random access algorithm*) may read and write to the vertices of G at any order, an algorithm which assumes a physical data extraction (referred to as a *physical algorithm*) must take into account the distance between two sequential operations. The reason for this is that the use of a random access algorithm is performed using a processing unit and random access memory, whereas the use of a physical algorithm is actually done in the physical environment (or a simulated physical environment, which maintains the information access paradigm). Thus, a random access algorithm can access any vertex of the graph in $O(1)$, while a physical algorithm is confined to the distances imposed by the physical metric.

For example, for $u, v \in V$, let us assume that the distance between v and u in G is 5. Then if after a 'read' request from u, the algorithm orders a 'write' request to v, this process will take at least 5 time steps, and will therefore consume at least 5 fuel units. Furthermore, depending on the model assumed for the mobile agents knowledge base, this operation may take even longer, if, for example, the agents are not familiar with the shortest path from u to v, but rather know of a much longer path connecting the two vertices.

As can easily be seen from the previous example, while designing physical swarm algorithms, one must take into account an entire set of considerations which are often disregarded, while designing random access swarm algorithms.

5.4.2 The Problem — Pattern Matching by a Swarm of Mobile Agents

Pattern matching in graphs is the following problem: Given a graph G on n vertices and a graph H on h vertices, find whether G has an induced subgraph isomorphic to H. Many applications of pattern matching can be found in theory and practice of computer science, see e.g. [38] for more details. It is well known that this problem is computationally hard whenever H (and also h) is not constant, but rather a part of the input (a subgraph isomorphism problem), while it has a trivial solution of polynomial complexity if H is a constant graph. Note that the subgraph isomorphism problem is reducible to the *Max-Clique* problem.

This work considers dense physical graphs (graphs which contain many subgraphs isomorphic to the desired pattern) while the goal is to find one of them within as minimal moves on the graph as possible. A graph G on n vertices is called *dense* if the average degree of a vertex in G is $\Omega(n^2)$. Alternatively, we can slightly weaken this condition by requiring that the number of edges in G is $\Omega(n)$, as long as the existence of a large number of the requested patterns can be ensured. The vertices of G are indexed from 1 to n, and the edges of G are indexed from 1 to m, where $m \leq \binom{n}{2}$.

We assume that whenever a vertex $v \in V(G)$ is visited, all edges incident with v are revealed (i.e. their indices are revealed), and naturally v's index is also revealed. Hence, if an edge $e = (u, v)$ exists in G, and some agent visited u and v (in any order), then it can deduce that v is a neighbor of u, even if the agent did not travel on e. If there is a communication between the agents, it is enough that one of the agents visited u and some other agent visited v for this information to be deduced.

One of the artificial examples of a similar model might be an unknown terrain with indexed cities and roads, where the roads are signed with their indices (say with road signs), but their end points' indices are not mentioned. However, we do not assume that the graph is planar (i.e. there might be roads, bridges and tunnels, crossing each other in any way).

Similar to ordinary navigation tasks (for example [30, 31, 32, 33]), the purpose of each agent employing the search protocol is to reach the "goal vertex" as soon as possible. However, since the goal of the agents is to detect a *k-clique*, the goal vertex is in fact the vertex which when discovered completes a *k-clique* in the agent's knowledge base. Thus, there is no specific goal vertex, but rather several *floating goal vertices*, whose identities depends on the following elements :

- The structure of G (namely, V and E).
- The information the agent had collected thus far.
- The information sharing model of the problem (be it a distributed memory, centralized memory, etc').

Note also that this problem is not an ordinary exploration problem (see [16]), where the entire graph should be explored in order for it to be mapped out. Once a requested *k-clique* is found by one of the agents, the problem is terminated successfully. This often occurs while only a small portion of the graph's vertices have been visited.

Another distinction should be made between the problem that is presented in this work and those examined in the field of multi agents routing (see [39, 40]). While multi agents routing mainly deals with the problem of finding paths in dynamic networks where the structure of the environment is dynamically changing due to load balancing and congestion, the *physical k-clique* problem considers a stable physical environment (somewhat similar to the work of [7] and [2]).

5.4.3 Motivation

It is our belief that work on swarm protocols for physical graphs is strongly required since physical graphs and networks, which have an essential role in nowadays research and industry application, are becoming more and more complex, and thus new techniques for such graphs must be composed. Several examples for such networks are the *world wide web* [35, 28, 27], the physical and logical infrastructure of the internet [28], power grids, electronic circuits [28] — all of which are complex physical environments.

The *physical k-clique* problem has several "real world" applications. For example, tracking the connectivity of a computer or telephone network, which can be utilized in order to improve routing schemes within this network. Another example may be a distributed mechanism which searches many databases containing transactions and email correspondences, in order to identify a group of people who maintain tight connections between the group's members (a possible indicator of a potential terrorists group).

Another reason the *physical k-clique* problem was selected for this research is that the *k-clique* problem, on which the *physical k-clique* problem is based, is known to be a significantly hard problem. While most of known NP-complete problems can be approximated quite well in polynomial (and sometimes even linear) time (as shown in [125]), this is not the case for the *k-clique* problem. An explanation of why there are no "semi-efficient" algorithms for the *k-clique* problem (and thus, that the best solution for it is an exhaustive search) and why the *k-clique* problem can not be approximated in *any* way, unless $P = NP$, can be found in [126]. Additional details regarding NP-Complete problems can be found in [127].

Since the *physical k-Clique* problem is in fact an instance of the *physical pattern matching* problem, solving it can serve as a first step towards a general pattern matching swarm algorithm. In future works we intend to show that the algorithm presented in this work can be applied to any pattern with few modifications.

5.4.4 Physical Clique Finding Protocol

Let $r(t) = (\tau_1(t), \tau_2(t), \ldots, \tau_n(t))$ denote the locations of the n agents at time t. The requested search protocol is therefore a movement rule f such that for every agent i, $f(\tau_i(t), N(\tau_i(t)), \mathcal{M}_i(t)) \in N(\tau_i(t))$, where for a vertex v, $N(v)$ denotes the neighbors of v, (e.g. $N(v) \triangleq \{u \in V \mid (v,u) \in E\}$).

$\mathcal{M}_i$ is the memory for agent i, containing information gathered by it through movement along G and by reading information from the shared memory. The requested rule f should meet the following goals :

- **Finding a k-Clique** : $(\exists t_{success} \mid k\text{–}clique \in \mathcal{M}_i(t_{success}))$ such that this $t_{success}$ is minimal.

- **Agreement on Completion** : within a finite time after the completion of the mission, all the mobile agents must be aware that the mission was completed, and come to a halt.
- **Efficiency** : in time, agents' memory and the size of the shared memory.

The requested rule should also be *fault tolerant*, meaning that even if some of the agents malfunction, the remaining ones will eventually find the target clique, albeit somewhat slower.

For solving the *Physical k-Clique* problem a search protocol named **PCF** is suggested. This protocol can be described as follows: each agent holds a knowledge base which contains some information regarding G. Every time an agent enters a vertex v, it performs a *data merge* process, in which the knowledge base of the agent updates and is updated by the central knowledge base.

The main idea of the search protocol is exploring the graph in directions that have the highest potential for discovering the desired pattern. Considering only cliques simplifies the arguments because of the clique's perfect symmetry.

All potential sets of vertices are sorted according to the largest clique which is contained in the set. Sets containing the same size of maximal clique are sorted according to the total number of unexplored edges which touch the vertices of the set (unexplored edges are edges $e(v,u)$ whereas at least one the identities of v and u is yet unknown) . As large the number of such edges is, the more likely it is for the set to be expandable to a *k-clique*. In addition, if a *k-clique* was found, the sort criteria places it on the top of the list, hence the agents are immediately aware of it. The algorithm uses a *job list*, containing the sets described above.

Generally, when looking for a pattern H of size h in graph G, every set of $m, m < h$ visited vertices in G that might be completed to a subgraph of G isomorphic to H (i.e. there are enough unexplored edges for every vertex) forms a potential sub-H of size m. While considering cliques as the patterns, we only need to verify that the m vertices of the set form a clique and that every vertex in the set has at least $h-(m+1)$ unexplored edges (which is the minimal requirement in order for this set to be expendable to an $h-clique$. Sets which do not meet this demand are deleted from the job list. The job list is updated (if needed) after every move of the agents.

If there are α available agents in a turn, the algorithm assigns the top α jobs in the sorted job list to the agents, where the assignments are made in a way that minimizes the total travel distance in L_∞ norm. This is done in order to minimize the travel efforts of the agents. This is an example of a difference between physical problems and "regular" problems — whereas in conventional complexity scheme all the ways of dispersing the jobs among the agents are identical, in the complexity scheme of physical problems we would like to assign jobs to the nearest agents.

Once an agent reaches its goal (the closest vertex of the job assigned to the agent), it writes the new discovered information in the adjacency matrix

and becomes available. At the beginning of each turn, if the first job in the list is associated to a list of k vertices the algorithm declares that a clique has been found, and terminates. Notice that all the agents use a common job list. In addition, the agents are assumed to broadcast every new information they encounter, creating a *common shared memory.*

We assume that all distances (i.e. graph edges) are equal to one unit, and that the agents have sufficient memory and computational power. In addition, we assume that that there are many cliques of the required size in G (otherwise, if the number of cliques is too small (e.g. $o\binom{n}{k}$), the optimal algorithm for discovering them would be the exhaustive exploration of G). The algorithm, used by each agent i appears in Figure 5.4.

```
Algorithm PCF :
While the first job in the common job list is not associated to a set of
k vertices
    Assign top job in jobs list to the agent;
    Perform the job;
    Update the agent's data structures;
    Update the agent's jobs list (or the common list);
    Update the list of available agents;
    If (all vertices explored) then
        STOP;
End PCF;
```

```
Procedure CREATE DATA STRUCTURES(i) :
Create an n × n matrix with entries of type integer;
// Represents edge indices according to vertex adjacency of G,
// as discovered by the agents.
Create a sorted list of jobs;
// Each job is associated with one potential clique.
// The job list is sorted according to the following criteria :
// (a) Size of the potential sub clique
// (b) Number of still unexplored edges
// (c) The distance of the job from the agents
// The distance is computed according to L∞ norm, that is the
// min—max on the travel distance.
End CREATE DATA STRUCTURES;
```

```
Procedure INITIALIZE() :
Initialize the agents' jobs lists;
// Note that the common job list may contain large potential
// sub-cliques, immediately upon initialization.
CREATE DATA STRUCTURES(i);
Choose random starting points on G for the agents;
Place the agents in these starting points;
Start the agents according to the PCF algorithm;
End INITIALIZE;
```

Fig. 5.4. The **PCF** search algorithm for the centralized shared memory model.

5.4.5 Results

The algorithm was tested on *Erdös-Renyi* random graphs $G \sim G(n,p)$ where G has n vertices, and each pair of vertices forms an edge in G with probability p independently of each other. In order to ensure that G has enough clique sub-graphs of size k, the value of p should be chosen wisely. Formally, the probability that k vertices of G form a clique is p^k, thus by linearity of expectation and the second moment, G will have at least $\frac{1}{4}\binom{n}{k}p^k$ cliques of size k with probability $1-o(1)$ (for further background on probabilistic arguments see e.g. [46]). Since we require that G will contain a sufficient number of k-clique (namely, $O(\sqrt{n})$ k-cliques), we choose the value of p with respect to the formula above, and specifically :

$$p = \left(\frac{\sqrt{16n} \cdot k!}{\prod_{i=0}^{k-1} n-i} \right)^{\frac{1}{k}}$$

Fig. 5.5 contains experimental results of the search algorithm for cliques of size 10 in graphs of sizes 500 and 2000. The number of agents searching the graphs is 5 through 50 agents. It can be seen that while adding more agents dramatically improves the search time, the system reach a state of saturation, after which adding more agents yields only mild contribution to the performance of the swarm, if any. This can be best observed in the smaller graph, where enlarging the number of agents from 15 to 50 decreases the search time by less than 30%, where as increasing the number of agents from 5 to 15 decreases the search time by 70%.

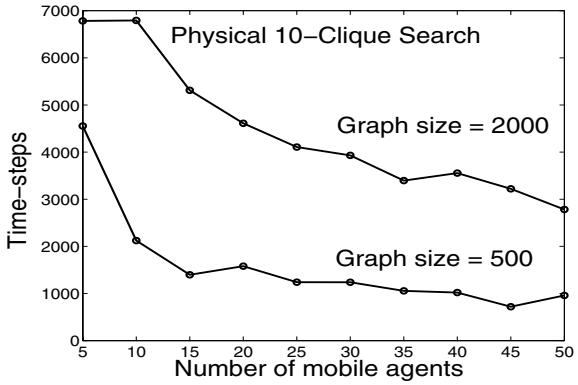


Fig. 5.5. Results of the Physical 10-Clique problem. Notice how the performance of the swarm increases while adding more agents, until it reaches a state of saturation.

Fig. 5.7 examines the increase in the swarm's search time for larger graphs. Notice that unlike the regular *k-clique* problem, in which the search time

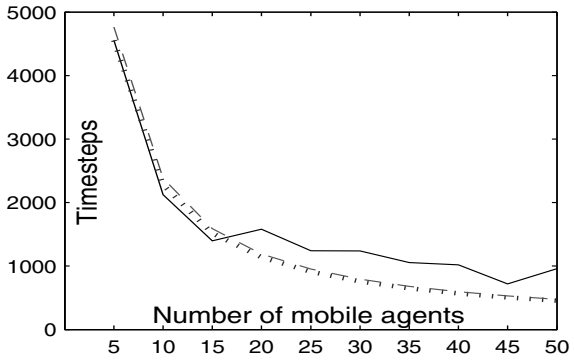


Fig. 5.6. A comparison between the results of the Physical 10-Clique problem in a graph of 500 vertices as a function of the number of agents, and the behavior of the $y = \frac{1}{x}$ function. The dotted curve presents the $y = \frac{1}{x}$ function where $y(5)$ equals the experimental solving time of 5 agents. The red dashed curve presents the instance of $y = \frac{1}{x}$ function which obtains the minimal distance from the entire set of experimental results, in L_2 norm. The proximity of the curves indicates that the system operates as expected. Observe how the later points of the experimental results are located above the $y = \frac{1}{x}$ curve, which can be expected since the utilization of the system is likely to be smaller than 100%.

equals $O(n^k)$, the results are almost linear in n. This demonstrates the basic claim that once computation resources are disregarded in comparison of travel effort, the complexity of many central problems in computer science change dramatically.

In addition, note that the search time for 20 agents is consistently smaller than for 10 agents. However, while in smaller graphs the difference is approximately 50%, for larger graphs it shrinks down to 25% for a graph of 4500 vertices. Interestingly, when increasing the number of agents to 30, the performance almost does not improve at all (meaning that the system had reached a state of saturation).

5.4.6 Exploration in Physical Environments

While considering the physical k-clique problem, an interesting discussion can take place considering the comparison between this problem and the problem of physical exploration by a swarm of mobile agents. The exploration problem for a decentralized group of mobile agents is merely the problem of achieving a complete knowledge of the graph under investigation by one of the agents. This comparison is indeed interesting since under the assumptions of a physical environment (namely, that computation resources can be disregarded in comparison to travel efforts) once an exploration of the environment

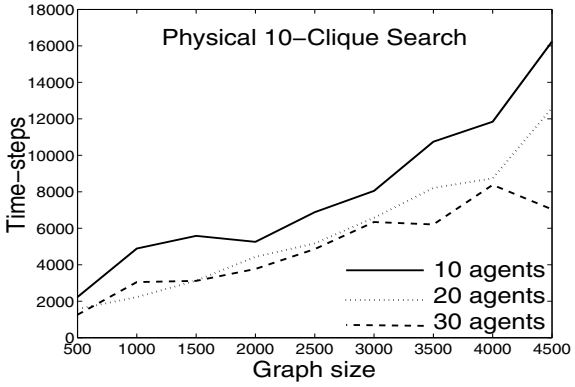


Fig. 5.7. Results of the Physical 10-Clique problem. Notice how the performance of the swarm increases while increasing the number of agents from 20 to 30, whereas this number is increased to 30, the performance almost never changed. In addition, observe the linearity of the graph, whereas the complexity of the problem in orthodox graph theory scheme is $O(n^k)$.

is completed, all cliques contained in the world are revealed. As a result, upper bounds for the exploration problem serve also as upper bounds for the physical k-clique problem.

Let G be a physical graph over n vertices. We assume that our physical environments are *Erdös-Renyi* random graphs $G \sim G(n,p)$ where G has n vertices, and each pair of vertices form an edge in G with probability p independently of each other. The edge density parameter of G, p may be either a constant or a function of n. We also assume that the agents are spread uniformly at random over the nodes of G. We shall assume without loss of generality, that the agents move according to the random walk algorithm. This type of behavior simulates the weakest model of agents, and hence time bounds here may impose similar time bounds for other essential behaviors of agents (since an agent can always act according to the random walk algorithm, and achieve its exploration time).

Under the above settings, we are interested in examining the number of required steps on the physical graph G, after which the whole graph G is known to some agent (or similarly, the complete information regarding the graph G is stored in one of its nodes). Let m denote the number of agents.

In order to obtain an upper bound for the exploration time of a random walkers swarm in G let us use the following bound of [129] :

$$E(ex_G) = O\Big(\frac{|E|^2 log^3(|V|)}{m^2}\Big)$$

Results of this bound for graphs used by the k-clique search protocol appears in Figures 5.8 and 5.9.

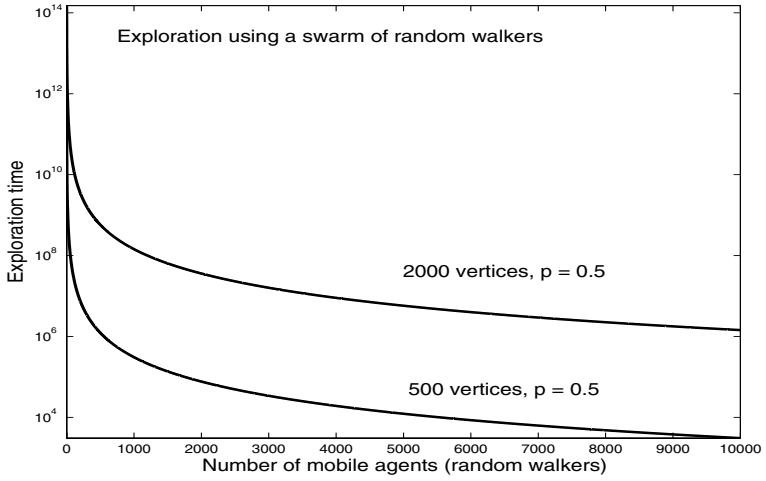


Fig. 5.8. Results predicted by the bound of [129] regarding exploration of general graphs using a swarm of mobile agents which use the random walk algorithm. The graph shows the mission completion time as a function of the number of agents.

5.4.7 Swarm Intelligence for Physical Environments — Related Work

Hitherto, there have been various works which examine problems in physical graphs, as well as the use of swarm based systems for such problems. For example, [40, 41] use mobile agents in order to find shortest paths in (dynamically changing) telecom networks, where the load balancing on the edges of the graph is unstable. Similarly, several widely used Internet routing algorithms (such as *BGP* [42], *RIP* [43] etc') propagate shortest path information around the network and cache it at local vertices by storing routing tables with information about the next hop. Another known routing system which uses "ants" and "pheromonoes" is the *AntNet* system [39]. In *AntNet*, ants randomly move around the network and modify the routing tables to be as accurate as possible.

While these approaches seem similar, there is a great difference between these works and one presented here, both concerning the environment in which the agent operate, the data that is stored at each vertex and the problem to be solved.

First, most of the works mentioned above which concern routing problems, assume that the environment is a telecom network of some sort, which changes dynamically over time. In this work, we assume the graph to be fixed and corresponds to a real physical environment (with Euclidean distances or some other metric), while the difficulty is derived from fact that the graph is not known to the agents.

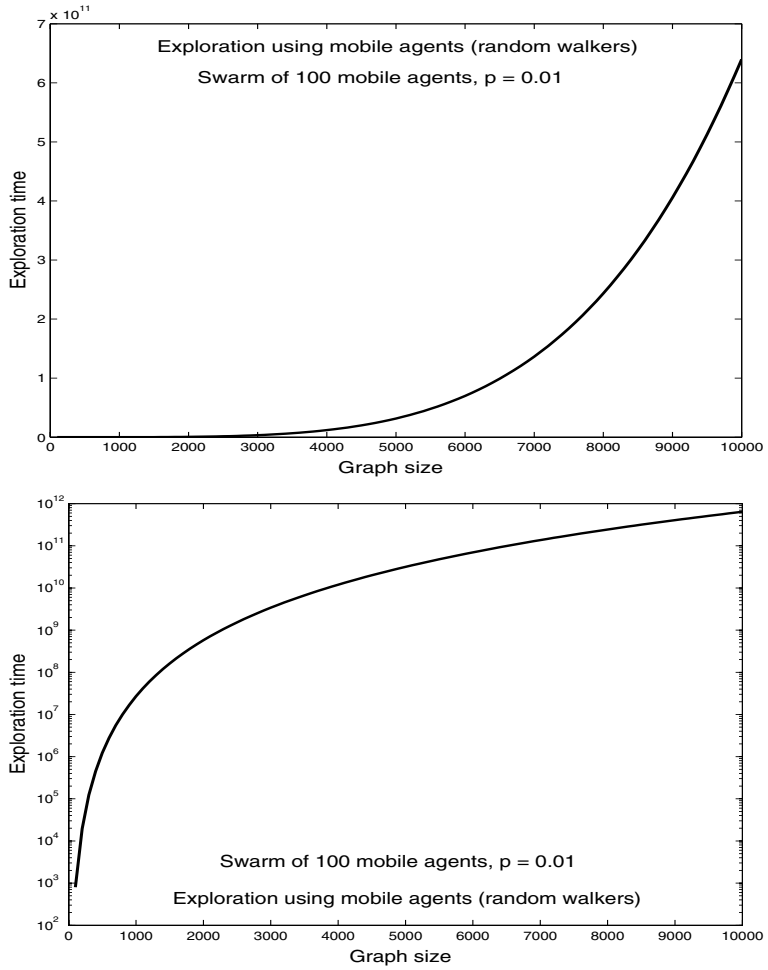


Fig. 5.9. Results predicted by the bound of [129] regarding exploration of general graphs using a swarm of mobile agents which uses the random walk algorithm. The graphs show the mission completion time as a function of the size of the graph.

Another difference is that these algorithms try to help a mobile agent (which for example, represents a phone conversation) to navigate to a certain goal. In other words, given the destination vertex (which could be any vertex of the environment) and the knowledge written in the current vertex, these algorithms try to locally decide where should the agent go to next, while the goal of each agent is merely to minimize the time it takes this particular agent to reach its destination. In the *physical k-clique* problem, on the other hand, the agents' goal is to find k vertices which form a *k-clique*, while the goal of

the agent is to minimize the time it takes the entire *swarm* to find such a clique.

Essentially, the approach presented in this work can be seen as a generalization of the *next hop lookup tables* mentioned earlier, since according to the partial knowledge of the graph, an agent decides on its next move, when this process is continually advancing, until a clique is found.

Similar to this approach, the works of [44] and [45] present a mechanism which find the shortest path within a physical graph, while assuming several communication mechanism.

5.5 Discussion and Conclusion

In this work three problems in the field of swarm intelligence were presented. These problems have several "real world" applications. While the Cooperative Hunters problem is already formulated in the form of such a problem, the Cooperative Cleaners problem may be used, for example, for a coordinating fire-fighting units, or an implementation of a distributed anti-virus system for computer networks. Additional applications are distributed search engines, and various military applications (such as a basis for UAV swarm systems, as offered in Section 5.3.3). Regarding the physical k-Clique problem, many of its applications were discussed in Section 5.4.3. Protocols for the problems were presented and analyzed, and several results were shown.

In addition, one of the major principles considering works in physical environments was shown (in Figure 5.6 of Section 5.4.5), in which a problem whose time complexity (according to orthodox complexity principles) equals $O(n^k)$ presented a physical complexity of only $O(n)$, demonstrating a basic difference between the two complexity schemes.

While examining these problems, new interesting opportunities for an extended research have emerged. We have already started investigating some of the above, producing more valuable results. Following are various aspects of this ongoing and future research :

5.5.1 Cooperative Cleaners

One of the important features of the SWEEP cleaning protocol is its simplicity. As mentioned in great length in Section 5.1.3, our paradigm assumes the use of highly simple agents, and thus derives designs of highly simple protocols. While examining the performance of this protocol, one may wonder what is the price which we pay by adopting such simple protocols instead of other more complex and resource demanding protocols. In order to answer this question, the authors have been experiencing with an $A*$ based mechanism (see [106]) which (after much centralized and highly computation demanding exhaustive processes) is able of producing *optimal solutions* for cleaning problems (meaning, solutions which are guaranteed to produce cleaning within the shortest

time possible). Surprisingly, the results of these experiments have shown that the performance of the **SWEEP** protocol are only roughly 30%–50% slower than the optimal solutions. These amazing results successfully demonstrate that a fully decentralized simple swarm protocol, assuming no communication or global knowledge, can produce results which are extremely close to the optimal solutions of the problem under investigation. This is possible thanks to the swarm behavior which emerges through the use of such protocol. The results of these experiments were published in [6] and an example appears in Figures 5.10 and 5.11.

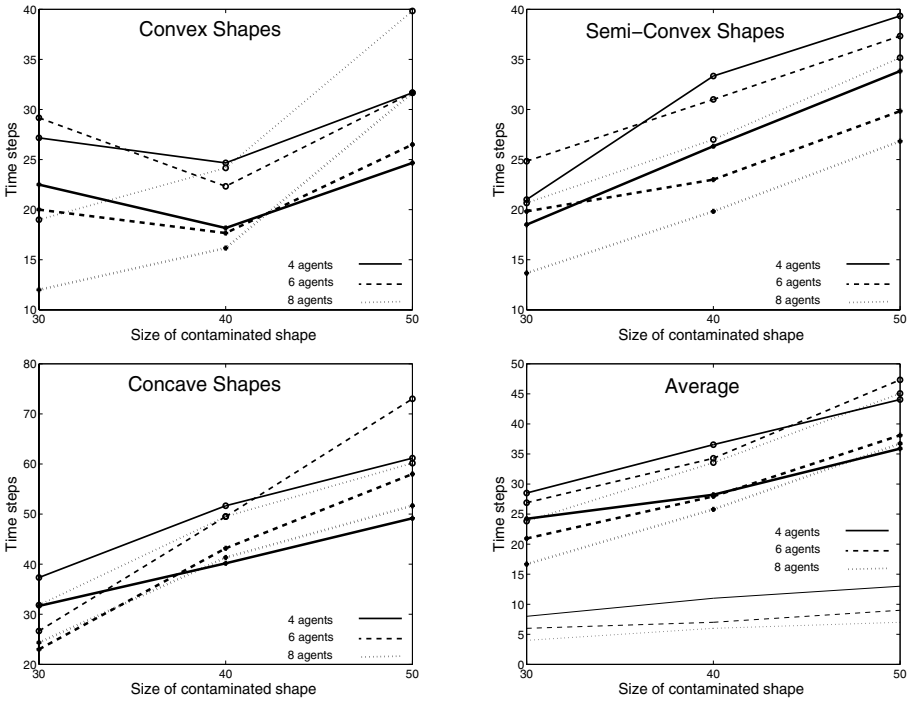


Fig. 5.10. Comparison of sub optimal and optimal algorithms. The lower and ticker three lines represent the cleaning time of the optimal algorithm whereas the upper lines represents the **SWEEP** cleaning protocol. In the right chart on the bottom, the lower three lines represent the lower bound of the optimal solution, as appears in Section 5.2.4.

Another interesting aspect is the feasibility question, i.e. foretelling the minimal number of agents required to clean a given shape (regardless of the cleaning time). In addition, developing new cleaning protocols for the problem and producing tighter bounds are also of interest to us. The authors

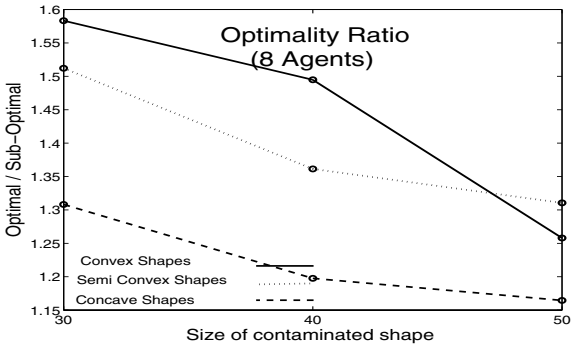


Fig. 5.11. Comparison of sub optimal and optimal algorithms. The Y-axes represents the $\frac{performance_{optimal}}{performance_{sub-optimal}}$ ratio for various sizes. Note that sa the problem is getting bigger, the sub-optimal performance of the **SWEEP** cleaning protocol get closer to those of the optimal algorithm.

are currently in the stages of developing improved cleaning protocols for the problem.

An explicit upper bound for the cleaning time of agents using the **SWEEP** protocol has been derived by the authors, using the Ψ function (see for example [105]). Another important result is the discovery of new geometric features, which are invariants with respect to the agents' activity and the spreading contamination. Their relevance to the problem was demonstrated by developing an improved upper bound for the cleaning time of the agents. Both results, including several other analytic results for the problem are presented in a paper which is currently under preparation.

Since many of the interesting "real world" applications take place in environments not easily reduced to lattices (for example, a distributed anti-virus mechanism for computer networks), additional analytic work should be performed concerning the dynamic cooperative cleaners problem in *non-planar graphs*. Although some of the existing work can immediately be applicable for such environments (for example, the generic lower bound, mentioned in Section 5.2.4), while trying to reconstruct other elements, one may face interesting new challenges.

5.5.2 Cooperative Hunters

While a flexible and fault-tolerant protocol for solving the problem, based on the dynamic cleaning problem was presented, there is still room for protocols, which will present enhanced performances. The authors are currently constructing hunting protocols which are not "cleaning protocols based" in hope of demonstrating improved results. One such protocol, which was shown to produce near-optimal results was recently constructed, and can be seen in [5].

In addition, there arises the question of robustness, meaning — given a swarm comprised of UAVs which are slightly slower than the minimal speed required by various hunting protocols, what is the average decrease in hunting performance of such swarm ? Thus, the quality of a hunting protocol is derived not only by the minimal speed limit it enforces on the UAVs or by the number of UAVs required, but also by its sensitivity to deviation from these values. A paper discussing the above is currently under preparation by the authors.

5.5.3 Physical k-Clique

While this work had assumed a *centralized shared memory* for the agents' communication, a second variant of the protocol, assuming a *distributed shared memory* model, in which the mobile agents can store and extract information using the graph's vertices is described by the authors in in[128]. This version is also much more similar to other common models of "real world" problems, since assuming storage of small portion of data in the graph nodes is considered much more easy assumption than assuming that the agents has the ability of broadcasting throughout the entire swarm. Surprisingly, our experimental results show that the performance of the special swarm search protocol designed for this problem is very close to the performance of the decentralized protocol. Again, this is a successful demonstration that the power of swarm intelligence may often compensate for shortage of sensing or communication capabilities.

In addition, since the protocol was designed to be employed by a swarm of mobile agents, it is highly dependent on the information gathered by the agents. In order to increase robustness, a special kind of *exploratory agents* were introduced to the system. A paper discussing the distributed swarm model as well as various enhancements (such as exploratory agents) and a many experimental results is currently under preparation.

As mentioned earlier, although the protocol described in this work was designed for the physical k-clique problem, we believe that by minor adjustments a generic protocol for finding a given pattern in a physical graph may be composed. The main difference between cliques and graph patterns in general is that cliques (or alternatively independent sets) have a perfect symmetry between their vertices. In other words, every pair of vertices in a clique has equivalent elements under some automorphism. The notion of "potential" is also naturally extensible to the general case. For example, given a pattern H on h vertices and a subset C of c vertices from the graph G, we say that C is potentially expandable to H if :

i. $c < h$.
ii. There exists an assignment π to the unknown pairs of vertices.
iii. There exists a set C' of $(h - c)$ vertices such that the induced subgraph on $C \cup C'$ under the assignment π is isomorphic to H.

To scale the potential of a certain subset C, a good estimation of the amount of possible assignments under which C is expandable to a set of vertices that induce a subgraph isomorphic to H must be produced.

In addition, it is our intention to devise such a protocol, as well as a protocol for the pattern matching problem in *dynamic* physical graphs, a problem which also bears a great deal of interest.

References

1. I.A. Wagner, A.M. Bruckstein: "Cooperative Cleaners: A Case of Distributed Ant-Robotics", in "Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath", Kluwer Academic Publishers, The Netherlands (1997), pp. 289–308
2. I.A. Wagner, M. Lindenbaum, A.M. Bruckstein: "Efficiently Searching a Graph by a Smell-Oriented Vertex Process", Annals of Mathematics and Artificial Intelligence, Issue 24 (1998), pp. 211–223
3. Y. Altshuler, A.M. Bruckstein, I.A. Wagner: "Swarm Robotics for a Dynamic Cleaning Problem", in "IEEE Swarm Intelligence Symposium" (SIS05), Pasadena, USA, (2005)
4. Y. Altshuler, V. Yanovski: "Dynamic Cooperative Cleaners — Various Remarks", Technical report, CS-2005-12, Technion - Israel Institute of Technology, (2005).
5. Y.Altshuler, V.Yanovsky, I.A.Wagner, A.M. Bruckstein: "The Cooperative Hunters – Efficient Cooperative Search For Smart Targets Using UAV Swarms", Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), Barcelona, Spain, (2005).
6. Y.Altshuler, I.A.Wagner, A.M. Bruckstein: "On Swarm Optimality In Dynamic And Symmetric Environments", Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), Barcelona, Spain, (2005).
7. R.C. Arkin, T. Balch: "Cooperative Multi Agent Robotic Systems", Artificial Intelligence and Mobile Robots, MIT/AAAI Press, Cambridge, MA, (1998)
8. R.A. Brooks: "Elephants Don't Play Chess", Designing Autonomous Agents, P. Maes (ed.), pp. 3–15, MIT press / Elsevier, (1990)
9. S.Levi: "Artificial Life - the Quest for a New Creation", Penguin, (1992)
10. S.Sen, M. Sekaran, J. Hale: "Learning to Coordinate Without Sharing Information", Proceedings of AAAI-94, pp. 426–431
11. L.Steels: "Cooperation Between Distributed Agents Through Self-Organization", Decentralized A.I. — Proc. of the first European Workshop on Modeling Autonomous Agents in Multi-Agents world, Y.DeMazeau, J.P.Muller (Eds.), pp. 175–196, Elsevier, (1990)
12. G.Beni, J.Wang: "Theoretical Problems for the Realization of Distributed Robotic Systems", Proc. of 1991 IEEE Internal Conference on Robotics and Automation, pp. 1914–1920, Sacramento, California April (1991)
13. V.Breitenberg: Vehicles, MIT Press (1984)
14. D.Henrich: "Space Efficient Region Filling in Raster Graphics", The Visual Computer, pp. 10:205–215, Springer-Verlag, (1994)

15. P.Vincent, I.Rubin: "A Framework and Analysis for Cooperative Search Using UAV Swarms", ACM Simposium on applied computing, 2004
16. M.A. Bender, A. Fernandez, D. Ron, A. Sahai, S.P. Vadhan: "The power of a pebble: Exploring and mapping directed graphs", In the proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computating, pp. 269–278, Dallas, Texas, May (1998).
17. L.P. Cordella, P. Foggia, C. Sansone, M. Vento: "Evaluating Performance of the VF Graph Matching Algorithm", Proc. of the 10th International Conference on Image Analysis and Processing, IEEE Computer Society Press, pp. 1172–1177, (1999).
18. J.R. Ullmann: "An Algorithm for Subgraph Isomorphism", Journal of the Association for Computing Machinery, vol. 23, pp. 31–42, (1976).
19. M.R. Garey, D.S. Johnson: "Computers and Intractability: A Guide to the Theory of NPCompleteness", Freeman & co., New York, (1979).
20. B.D. McKay: "Practical Graph Isomorphism", Congressus Numerantium, 30, pp. 45–87, (1981).
21. D.G. Corneil, C.C. Gotlieb: "An efficient algorithm for graph isomorphism", Journal of the Association for Computing Machinery, 17, pp. 51–64, (1970).
22. W.J. Christmas, J. Kittler, M. Petrou: "Structural Matching in Computer Vision Using Probabilistic Relaxation", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, no. 8, pp. 749–764, (1995).
23. P. Kuner, B. Ueberreiter: "Pattern recognition by graph matching — combinatorial versus continuous optimization", Int. J. Pattern Recognition and Artif. Intell., vol. 2, no. 3, pp. 527–542, (1988).
24. K.A. De Jong, W.M. Spears: "Using genetic algorithms to solve NP-complete problems", in Genetic Algorithms, (J.D. Schaffer, ed.), Morgan Kaufmann, Los Altus, CA., pp. 124–132, (1989).
25. A.A. Toptsis, P.C. Nelson: "Unidirectional and Bidirectional Search Algorithms", IEEE Software, 9(2), (1992).
26. J.B.H. Kwa: "BS*: An Admissible Bidirectional Staged Heuristic Search Algorithm", Artificial Intelligence, pp. 95–109, Mar., (1989).
27. D.J. Watts: "Small Worlds", Princeton University Press, Princeton NJ, (1999).
28. S.N. Dorogovtsev, J.F.F. Mendes: "Evolution of networks", Adv. Phys. 51, 1079, (2002).
29. D.S. Hochbaum, O. Goldschmidt, C. Hurken, G. Yu: "Approximation algorithms for the k-Clique Covering Problem", SIAM J. of Discrete Math, Vol 9:3, pp. 492–509, August, (1996).
30. P. Cucka, N.S. Netanyahu, A. Rosenfeld: "Learning in navigation: Goal finding in graphs", International journal of pattern recognition and artificial intelligence, 10(5):429–446, (1996).
31. R.E. Korf: "Real time heuristic search", Artificial intelligence, 42(3):189–211, (1990).
32. L. Shmoulian, E. Rimon: "Roadmap A*: an algoritm for minimizing travel effort in sensor based mobile robot navigation", In the proceedings of the IEEE International Conference on Robotics and Automation, pp. 356–362, Leuven, Belgium, May (1998).
33. A. Stentz: "Optimal and efficient path planning for partially known environments.", In the proceedings of the IEEE International Conference on Robotics and Automation, pp. 3310–3317, San Diego, CA, May (1994).

34. R.J. Wilson: "Introduction to Graph Theory", Longman, London, 2nd ed., (1979).
35. R. Albert, A.L. .Barabasi,: "Statistical Mechanics of Complex Networks", Reviews of Modern Physics, vol. 74, January, (2002).
36. O. Gerstel, S. Zaks: "The Virtual Path Layout problem in fast networks", In Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, pp. 235–243, Los Angeles, California, August, (1994).
37. S. Zaks: "Path Layout in ATM networks", Lecture Notes in Computer Science, 1338:pp. 144–177, (1997).
38. A. Apostolico, Z. Galil: "Pattern Matching Algorithms", Oxford University Press, Oxford, UK.
39. G. Di Caro, M. Dorigo: "AntNet:Distributed stigmergetic control for communiction networks", Journal of Artificial Intelligence Research, 9:317–365, (1998).
40. S. Appleby, S. Steward: "Mobile software agents for control in telecommunication networks", British Telecom Technology Journal, 12, pp. 104–113, (1994).
41. R. Schnooderwoerd, O. Holland, J. Bruten, L. Rothkrantz: "Ant-based load balancing in telecommunication networks", Adaptive Behavior 5(2), (1996).
42. Y. Rekhter, T. Li: "A Border Gateway Protocol", Request for Comments 1771, T.J Watson Research Center IBM Corporation & cisco Systems, March (1995).
43. G. Malkin: "RIPng Protocol Applicability Statement", RFC 2081, IETF Network Working Group, January, (1997).
44. A. Felner, R. Stern, A. Ben-Yair, S. Kraus, N. Netanyahu: "PHA*: Finding the Shortest Path with A* in Unknown Physical Environments", Journal of Artificial Intelligence Research, vol. 21, pp. 631–679, (2004)
45. A. Felner, Y. Shoshani, I.A.Wagner, A.M. Bruckstein: "Large Pheromones: A Case Study with Multi-agent Physical A*", Forth International Workshop on Ant Colony Optimization and Swarm Intelligence, (2004)
46. N. Alon, J. H. Spencer: "The probabilistic method", Wiley-Interscience (John Wiley & Sons), New York, (1992) (1^{st} edition) and (2000) (2^{nd} edition).
47. G. Dudek, M. Jenkin, E. Milios, D. Wilkes: "A Taxonomy for Multiagent Robotics". Autonomous Robots, 3:375397, (1996).
48. B.P.Gerkey, M.J.Mataric: "Sold! Market Methods for Multi-Robot Control", IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems, (2002).
49. M.Golfarelli, D.Maio, S. Rizzi: "A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm", Technical Report, 005-97, CSITE (Research Center For Informatics And Telecommunication Systems, associated with the University of Bologna, Italy), (1997).
50. G.Rabideau, T.Estlin, T.Chien, A.Barrett: "A Comparison of Coordinated Planning Methods for Cooperating Rovers", Proceedings of the American Institute of Aeronautics and Astronautics (AIAA) Space Technology Conference, (1999).
51. R.Smith: "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", IEEE Transactions on Computers C-29 (12), (1980).
52. S.M.Thayer, M.B.Dias, B.L.Digney, A.Stentz, B.Nabbe, M.Hebert: "Distributed Robotic Mapping of Extreme Environments", Proceedings of SPIE, Vol. 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, (2000).

53. M.P.Wellman, P.R.Wurman: "Market-Aware Agents for a Multiagent World", Robotics and Autonomous Systems, Vol. 24, pp.115–125, (1998).
54. R.Zlot, A.Stentz, M.B.Dias, S.Thayer: "Multi-Robot Exploration Controlled By A Market Economy", Proceedings of the IEEE International Conference on Robotics and Automation, (2002).
55. R.C.Arkin, T.Balch: "AuRA: Principles and Practice in Review", Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9, No. 2/3, pp.175–188, (1997).
56. D.Chevallier, S.Payandeh: "On Kinematic Geometry of Multi-Agent Manipulating System Based on the Contact Force Information", The 6^{th} International Conference on Intelligent Autonomous Systems (IAS-6), pp.188–195, (2000).
57. R.Alami, S.Fleury, M.Herrb, F.Ingrand, F.Robert: "Multi-Robot Cooperation in the Martha Project", IEEE Robotics and Automation Magazine, (1997).
58. T.Arai, H.Ogata, T.Suzuki: "Collision Avoidance Among Multiple Robots Using Virtual Impedance", In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 479-485, (1989).
59. R.C.Arkin: "Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation", Robotics and Autonomous Systems, 6:pp.105-122, (1990).
60. T.Balch, R.Arkin: "Behavior-Based Formation Control for Multi-Robot Teams", IEEE Transactions on Robotics and Automation, December (1998).
61. M.Benda, V.Jagannathan, R.Dodhiawalla: "On Optimal Cooperation of Knowledge Sources", Technical Report BCS-G2010-28, Boeing AI Center, August (1985).
62. G.Beni: "The Concept of Cellular Robot", In Proceedings of Third IEEE Symposium on Intelligent Control", pp.57-61, Arlington, Virginia, (1988).
63. H.Bojinov, A.Casal, T.Hogg: "Emergent Structures in Moduluar Self-Reconfigurable Robots", In Proceedings of the IEEE International Conference on Robotics and Automation, pp.1734-1741, (2000).
64. R.A.Brooks: "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, RA-2(1):14-23, March (1986).
65. C.Candea, H.Hu, L.Iocchi, D.Nardi, M.Piaggio: "Coordinating in Multi-Agent RoboCup Teams", Robotics and Autonomous Systems, 36(2- 3):67-86, August (2001).
66. A.Castano, R.Chokkalingam, P.Will: "Autonomous and Self-Sufficient CONRO Modules for Reconfigurable Robots", In Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000), pp. 155-164, (2000).
67. J.Deneubourg, S.Goss, G.Sandini, F.Ferrari, P.Dario: "Self-Organizing Collection and Transport of Objects in Unpredictable Environments", In Japan-U.S.A. Symposium on Flexible Automation, pp.1093-1098, Kyoto, Japan, (1990).
68. B.Donald, L.Gariepy, D.Rus: "Distributed Manipulation of Multiple Objects Using Ropes", In Proceedings of IEEE International Conference on Robotics and Automation, pp.450=457, (2000).
69. A.Drogoul J.Ferber: "From Tom Thumb to the Dockers: Some Experiments With Foraging Robots", In Proceedings of the Second International Conference on Simulation of Adaptive Behavior, pp.451-459, Honolulu, Hawaii, (1992).
70. C.Ferrari, E.Pagello, J.Ota, T.Arai: "Multirobot Motion Coordination in Space and Time", Robotics and Autonomous Systems, 25:219-229, (1998).

71. D.Fox, W.Burgard, H.Kruppa, S.Thrun: "Collaborative Multi-Robot Exploration", Autonomous Robots, 8(3):325-344, (2000).
72. T.Fukuda, S.Nakagawa: "A Dynamically Reconfigurable Robotic System (Concept of a System and Optimal Configurations)", In Proceedings of IECON, pp.588-595, (1987).
73. T.Haynes, S.Sen: "Evolving Behavioral Strategies in Predators and Prey", In Gerard Weiss and Sandip Sen, editors, Adaptation and Learning in Multi-Agent Systems, pp.113-126. Springer, (1986).
74. O.Khatib, K.Yokoi, K.Chang, D.Ruspini, R.Holmberg, A.Casal: "Vehicle/Arm Coordination and Mobile Manipulator Decentralized Cooperation", In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.546-553, (1996).
75. S.M.LaValle, D.Lin, L.J.Guibas, J.C.Latombe, R.Motwani: "Finding an Unpredictable Target in a Workspace with Obstacles", In Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA-97), pp.737-742, (1997).
76. V.J.Lumelsky, K.R.Harinarayan: "Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model", Autonomous Robots, 4(1):121-136, (1997).
77. D.MacKenzie, R.Arkin, J.Cameron: "Multiagent Mission Specification and Execution", Autonomous Robots, 4(1):29-52, (1997).
78. M.J.Mataric: "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence", In J.Meyer, H.Roitblat, and S.Wilson, editors, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, pp.432-441, Honolulu, Hawaii, MIT Press, (1992).
79. M.J.Mataric: "Interaction and Intelligent Behavior", PhD Thesis, Massachusetts Institute of Technology, (1994).
80. E.Pagello, A.DAngelo, C.Ferrari, R.Polesel, R.Rosati, A.Speranzon: "Emergent Behaviors of a Robot Team Performing Cooperative Tasks", Advanced Robotics, 2002.
81. E.Pagello, A.DAngelo, F.Montesello, F.Garelli, C.Ferrari: "Cooperative Behaviors in Multi-Robot Systems Through Implicit Communication", Robotics and Autonomous Systems, 29(1):65-77, (1999).
82. L.E.Parker: "ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation", IEEE Transactions on Robotics and Automation, 14(2):220-240, (1998).
83. L.E.Parker, C.Touzet: "Multi-Robot Learning in a Cooperative Observation Task", In Distributed Autonomous Robotic Systems 4, pp.391-401. Springer, (2000).
84. S.Premvuti, S.Yuta: "Consideration on the Cooperation of Multiple Autonomous Mobile Robots", In Proceedings of the IEEE International Workshop of Intelligent Robots and Systems, pp.59-63, Tsuchiura, Japan, (1990).
85. D.Rus, B.Donald, J.Jennings: "Moving Furniture with Teams of Autonomous Robots", In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.235-242, (1995).
86. D.Rus M.Vona: "A Physical Implementation of the Self-Reconfiguring Crystalline Robot", In Proceedings of the IEEE International Conference on Robotics and Automation, pp.1726-1733, (2000).

87. D.Stilwell, J.Bay: "Toward the Development of a Material Transport System Using Swarms of Ant-Like Robots", In Proceedings of IEEE International Conference on Robotics and Automation, pp.766-771, Atlanta, GA, (1993).
88. P.Stone, M.Veloso: "Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork", Artificial Intelligence, 110(2):241-273, June (1999).
89. P.Svestka, M.H.Overmars: "Coordinated Path Planning for Multiple Robots", Robotics and Autonomous Systems, 23(3):125-152, (1998).
90. C.Unsal, P.K.Khosla: "Mechatronic Design of a Modular self-Reconfiguring Robotic System", In Proceedings of the IEEE International Conference on Robotics and Automation, pp.1742-1747, (2000).
91. P.K.C.Wang: "Navigation Strategies for Multiple Autonomous Mobile Robots", In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.486-493, (1989).
92. Z.Wang, Y.Kimura, T.Takahashi, E.Nakano: "A Control Method of a Multiple Non-Holonomic Robot System for Cooperative Object Transportation", In Proceedings of Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000), pp.447-456, (2000).
93. A.Yamashita, M.Fukuchi, J.Ota, T.Arai, H.Asama: "Motion Planning for Cooperative Transportation of a Large Object by Multiple Mobile Robots in a 3D Environment", In Proceedings of IEEE International Conference on Robotics and Automation, pp.3144-3151, (2000).
94. M.Yim, D.G.Duff, K.D.Roufas: "Polybot: a Modular Reconfigurable Robot", In Proceedings of the IEEE International Conference on Robotics and Automation, pp.514-520, (2000).
95. E.Yoshida, S.Murata, S.Kokaji, K.Tomita, H.Kurokawa: "Micro Self-Reconfigurable Robotic System Using Shape Memory Alloy", In Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS 2000), pp.145-154, (2000).
96. J.Fredslund, M.J.Mataric: " Robot Formations Using Only Local Sensing and Control", In the proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001), pp.308–313, Banff, Alberta, Canada, (2001).
97. N.Gordon, I.A.Wagner, A.M.Bruckstein: "Discrete Bee Dance Algorithms for Pattern Formation on a Grid", In the proceedings of IEEE International Conference on Intelligent Agent Technology (IAT03), pp.545–549, October, (2003).
98. R.Madhavan, K.Fregene, L.E.Parker: "Distributed Heterogenous Outdoor Multi-Robot Localization", In the proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp.374–381, (2002).
99. M.B.Dias, A.Stentz: "A Market Approach to Multirobot Coordination": Technical Report, CMU-RI - TR-01-26, Robotics Institute, Carnegie Mellon University, (2001).
100. V. Yanovski, I.A. Wagner, A.M. Bruckstein: "A distributed ant algorithm for efficiently patrolling a network", Algorithmica, 37:165–186, (2003).
101. I.A. Wagner, A.M. Bruckstein: "ANTS: agents, networks, trees and subgraphs", Future Generation Computer Computer Systems Journal, 16(8):915–926, 2000.
102. V. Yanovski, I.A. Wagner, A.M. Bruckstein: "Vertex-ants-walk: a robust method for efficient exploration of faulty graphs. Annals of Mathematics and Artificial Intelligence, 31(1–4):99–112, (2001).

103. F.R. Adler, D.M. Gordon: "Information collection and spread by networks of partolling agents", The American Naturalist, 140(3):373–400, (1992).
104. D.M. Gordon: "The expandable network of ant exploration", Animal Behaviour, 50:372–378, (1995).
105. M. Abramowitz, I.A. Stegun: "Handbook of Mathematical Functions", National Bureau of Standards Applied Mathematics Series 55, (1964)
106. Hart, P.E., Nilsson, N.J., Raphael, B.: "A formal basis for the heuristic determination of minimum cost paths", IEEE Transactions on Systems Science and Cybernetics 4(2): 100–107, (1968).
107. Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M. and Baum, M.: "Cooperative Control for Autonomous Air Vehicles", In Cooperative Control and Optimization, R. Murphey and P. Pardalos, editors. Kluwer Academic Publishers, Boston, (2002).
108. Polycarpou, M., Yang, Y. and Passino, K.: "A Cooperative Search Framework for Distributed Agents", In Proceedings of the 2001 IEEE International Symposium on Intelligent Control (Mexico City, Mexico, September 5–7). IEEE, New Jersey, 1–6, (2001).
109. Stone, L.D: "Theory of Optimal Search", Academic Press, New York, (1975).
110. Koopman, B.O: "The Theory of Search II, Target Detection", Operations Research 4, 5, 503–531, October, (1956).
111. Koenig, S., Liu, Y.: "Terrain Coverage with Ant Robots: A Simulation Study", AGENTS'01, May 28–June 1, Montreal, Quebec, Canada, (2001).
112. Dorigo M., L.M. Gambardella: "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computation, 1(1):53-66 (1997).
113. Gambardella L. M. and M. Dorigo: "HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem", Tech. Rep. No. IDSIA 97-11, IDSIA, Lugano, Switzerland, (1997).
114. Gambardella L. M., E. Taillard and M. Dorigo: "Ant Colonies for the Quadratic Assignment Problem". Journal of the Operational Research Society, 50:167-176 (1999).
115. Bullnheimer B., R.F. Hartl and C. Strauss: "An Improved Ant system Algorithm for the Vehicle Routing Problem", Paper presented at the Sixth Viennese workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna (Austria), May 21-23, (1997), appears in: Annals of Operations Research (Dawid, Feichtinger and Hartl (eds.): Nonlinear Economic Dynamics and Control, (1999)
116. Colorni A., M. Dorigo, V. Maniezzo and M. Trubian: "Ant system for Job-shop Scheduling", JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science, 34(1):39-53 (1994).
117. Costa D. and A. Hertz: "Ants Can Colour Graphs", Journal of the Operational Research Society, 48, 295-305 (1997).
118. Kuntz P., P. Layzell and D. Snyers: "A Colony of Ant-like Agents for Partitioning in VLSI Technology", Proceedings of the Fourth European Conference on Artificial Life, P. Husbands and I. Harvey, (Eds.), 417-424, MIT Press (1997).
119. Schoonderwoerd R., O. Holland, J. Bruten and L. Rothkrantz: "Ant-based Load Balancing in Telecommunications Networks", Adaptive Behavior, 5(2):169–207 (1997).

120. Navarro Varela G. and M.C. Sinclair: "Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation", Proceedings of the Congress on Evolutionary Computation (CEC'99), Washington DC, USA, July (1999).
121. Yanowski, V. Wagner I.A., and Bruckstein A.M., "A Distributed Ant Algorithm for Efficiently Patrolling a Network", Workshop on Interdisciplinary Applications of Graph Theory and Algorithms, Haifa, Israel, April 17-18, (2001).
122. Machado A., Ramalho G., Zucker J.D., Drogoul A.: "Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures", Proceedings of MABS'02 (Multi-Agent Based Simulation, Bologna, Italy, July 2002), LNCS, Springer-Verlag (2002).
123. Rouff C., Hinchey M., Truszkowski W., Rash J.: "Verifying large numbers of cooperating adaptive agents", Parallel and Distributed Systems, Proceedings of the 11th International Conference on Volume 1, 20-22 July 2005 Page(s):391 - 397 Vol. 1 (2005).
124. P. Scerri, E. Liao, Yang. Xu, M. Lewis, G. Lai, and K. Sycara: "Coordinating very large groups of wide area search munitions", Theory and Al gorithms for Cooperative Systems, chapter. World Scientific Publishing, (2004).
125. V. Vazirani: "Approximation Algorithms", Springer-Verlag, (2001).
126. S. Arora, S. Safra: "Probabilistic checking of proofs: A new characterization of NP", Journal of the ACM, (1998).
127. M. Garey, D. Johnson: "Computers and Intractability: A Guide to the Theory of NP-Completeness", San Francisco, CA: W. H. Freeman, (1979).
128. Y. Altshuler, A. Matsliah, A. Felner: "On the Complexity of Physical Problems and a Swarm Algorithm for k-Clique Search in Physical Graphs", European Conference on Complex Systems (ECCS-05), Paris, France, November (2005).
129. A.Z. Broder, A.R. Karlin, P. Raghavan, E. Upfal: "Trading Space for Time in Undirected $s - t$ Connectivity", ACM Symposium on Theory of Computing (STOC), pp. 543–549, (1989).